DM

Extension of Action Rule Grammar and Implementation of Processing Engine of a DEMO Based Low-Code Platform

MASTER DISSERTATION

Vítor Hugo Silva Freitas
MASTER IN INFORMATICS ENGINEERING

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

June | 2023

Extension of Action Rule Grammar and Implementation of Processing Engine of a DEMO Based Low-Code Platform

MASTER DISSERTATION

Vítor Hugo Silva Freitas
MASTER IN INFORMATICS ENGINEERING

ORIENTATION
David Sardinha Andrade de Aveiro

Faculty of Exact Sciences and Engineering

Master of Science Degree in Informatics Engineering

Extension of Action Rule Grammar and Implementation of Processing Engine of a DEMO Based Low-Code Platform

Vítor Hugo Silva Freitas

Supervised by:

David Sardinha Andrade de Aveiro

Constituição do júri de provas públicas:

Nome completo (categoria), Presidente

Nome completo (categoria), Vogal

Nome completo (categoria), Vogal

Thursday 20th July, 2023

Resumo

Numerosos estudos afirmam que muitos projetos de software ficam aquém das expetativas iniciais dos utilizadores finais. Causas comuns para estas falhas são objetivos irrealistas do projeto e requisitos incompletos, entre outros.

O trabalho desenvolvido nesta tese ocorre no contexto do projeto DISME, uma plataforma low-code para modelação e execução de processos de negócio que pretende ultrapassar alguns destes problemas comuns em sistemas de informação, de modo a tornar a sua utilização para apoio à decisão mais intuitiva, personalizável e adaptável, de forma dinâmica e sem necessidade de programação.

No âmbito do DISME, estendeu-se e aprimorou-se um novo meta-modelo para o Modelo de Ação do DEMO, e desenvolveu-se o componente referente ao Executor do Sistema, cuja função é interpretar e executar as Regras de Ação. Foi depois integrado num Dashboard, que permite uma gestão de tarefas e processos de fácil utilização.

No decorrer deste desenvolvimento, notou-se ser de igual importância a extensão de outros componentes relativos ao desenho e execução de Regras de Ação, mais concretamente os componentes de gestão de Regras de Ação e de formulários do mesmo projeto, respetivamente, e a criação de um componente de parametrização para facilitar a gestão da especificação do sistema.

Para comprovar a eficácia da plataforma, foi realizada uma experiência comparando a abordagem tradicional de desenvolvimento com uma abordagem low-code utilizando a plataforma DISME. Para o caso específico utilizado, observou-se uma redução de 94,63% no esforço necessário, e uma redução de 86% relativamente à complexidade.

A usabilidade da plataforma foi também avaliada via métodos qualitativos e quantitativos. A avaliação qualitativa através do método Think Aloud deu feedback valioso sobre vários aspetos da plataforma. Os participantes consideraram a plataforma cativante, intuitiva e de fácil utilização. A avaliação quantitativa utilizando a System Usability Scale confirmou estas conclusões, com uma pontuação global de usabilidade de 89,25%.

Keywords: Engenharia Organizacional · DEMO · Regras de Ação · Sistemas de Informação · Requisitos · Fluxo de Trabalho

Abstract

Numerous studies find that many software projects fall short of end customers' initial expectations. Common causes for software project failures are unrealistic project objectives and incomplete requirements, among others.

The work developed in this thesis occurs in the context of the DISME project, a low-code platform for the modelling and execution of business processes that intends to overcome some of these common problems in information systems, in order to make their use for decision support more intuitive, customizable and adaptable, dynamically and without the need for programming.

In the scope of DISME, a new meta-model was extended and improved for DEMO's Action Model, and the component related to the System Executor was developed, whose function is to interpret and run the Action Rules. It was then integrated with a Dashboard, which allows user-friendly task and process management to the platform's users.

During this development, it was noted that it was equally important to extend other components relative to the design and execution of Action Rules, more specifically the components of Action Rule management and form management of the same project, respectively, and to create a parameterization component for easier management of the system's specification.

To prove the efficacy of the platform, an experiment was made, comparing the traditional development approach with a low-code one using DISME. For the specific case used, our findings showed a 94.63% reduction in the needed effort. Regarding complexity, a reduction of 86% was observed.

The usability of the platform was then evaluated using both qualitative and quantitative methods. The qualitative evaluation through the Think Aloud method provided valuable feedback on various aspects of the platform. Participants found the platform engaging, intuitive, visually appealing, and user-friendly. The quantitative evaluation using the System Usability Scale confirmed these findings, with an overall usability score of 89.25%.

Keywords: Enterprise Engineering · DEMO · Action Rules · Information Systems · Requirements · Workflow

Agradecimentos

This project became a reality because of the hard work and dedication of so many amazing people. I couldn't have made it through this journey without their support and motivation. I'm incredibly grateful to all of them!

First and foremost, I would like to thank my supervisor, Professor David Sardinha Andrade de Aveiro, for guiding and supporting me through this thesis project. He helped me navigate through the challenges, enabling me to overcome obstacles and approach this project with a critical mindset. I'm truly grateful for his unwavering support and encouragement. Not only that, but I also want to thank him for all the guidance, opportunities, and support he has given me since we first started working together towards the end of my Bachelor's Degree, I really am grateful for it all. If it wasn't for these opportunities, I wouldn't be doing this master's degree.

Additionally, this endeavour would not have been possible without the support from ARDITI, Regional Agency for the Development of Research, Technology and Innovation, and its research grants over the last 3 years.

I'd also like to recognize all the effort from all of Enterprise Engineering Lab's colleagues, for their unmatched work and help.

Words cannot express my gratitude towards my family, namely my mother, my father, and my sister. They have been supporting me in this academic journey that started in 2015, through all the ups and downs, and have never left my side, supporting me and encouraging me when all hope seemed lost. I really wouldn't be able to be where I am today without all their love and unconditional support. I haven't always taken the best decisions on this academic endeavour, but their hope in me always made me want to overcome that and pursue the best version of myself. Thank you. I will never be able to repay all that you have given me through these years.

A big thank you too to all my friends, especially João Pedro, Danny and Patrick, and also to Suíço, Guilherme, Leme, Tiago, Fábio, Dylan, Rosita and Lizandra. I know I haven't always been the best of friend, but reconnecting with you and having your friendship really means the world to me and makes me want to better myself for you. I am forever grateful for the moments I have spent with you, all the laughter and good moments on the good and the bad days. Hopefully many more are to come for many years.

To you all, I am really grateful.

Table of Contents

List of Figures

List of Tables

Lista de Acrónimos

AM Action Model

AOM Adaptive Object-Model

API Application Programming Interface

AR Action Rule

ARS Action Rules Specification

BPM Business Process Management

BRaNT Belief Revision applied to Neurorehabilitation Therapy

CAD Concept Attribute Diagram

CM Cooperation Model

CRD Concept and Relationships Diagram

CRUD Create, Read, Update and Delete

CSD Coordination Structure Diagram

DB Database

DBMS Database Management System

DEMO Design and Engineering Methodology for Organizations

DISME Direct Information Systems Modeller and Executer

DSML Domain Specific Modelling Language

EBNF Extended Backus–Naur Form

EE Enterprise Engineering

FDT Fact Description Table

FM Fact Model

GUI Graphical User Interface

IDE Integrated Development Environment

IS Information System

IT Information Technology

LCP Low-Code Platform

LOC Lines of Code

MDD Model-Driven Development

MVC Model-View-Controller

O-R Object-Relational

ORM Object–Relational Mapping

PM Process Model

PSI Performance in Social Interaction

RAC Rent-A-Car

REST REpresentational State Transfer

SPA Single Page Application

SUS System Usability Scale

SoI Scope of Interest

TDT Transaction Description Table

TPT Transaction Product Table

WYSIWYG What You See Is What You Get

WoM Way of Modelling

WoT Way of Thinking

WoW Way of Working

XML Extensible Markup Language

1 Introduction

This chapter presents the project's project context, objectives, implementation fundamentals and research method alongside the document's outline for a better understanding of the overall scope.

1.1 Project Context

This subsection serves as a comprehensive introduction to the project context, providing the motivation behind its inception, an overview of low-code platforms (LCPs), and a presentation of the DEMO-based low-code platform on which this thesis work was built. By exploring these key aspects, we aim to establish a strong foundation for the subsequent chapters, leading to a comprehensive understanding of the research topic and its implications.

1.1.1 Motivation

Numerous studies find that many software projects fall short of end customers' initial expectations. From [1], where certain case studies were conducted, a survey of 800 IT (Information Technology) managers [2, 3] revealed that 63% of software development projects failed, 49% went over budget, 47% cost more to maintain than anticipated, and 41% fell short of meeting user and business requirements. Ibraigheeth et al. examined several project failure-related reports that have been published and built a list of failure factors responsible for this high failure rate. Unrealistic project objectives, incomplete requirements, a lack of stakeholder and user involvement, issues with project management and control, an inadequate budget, changing or inconsistent requirements and specifications, a lack of planning, poor communication, and the use of new technologies for which software developers lacked the necessary experience and expertise are common causes [4].

1.1.2 Low-Code Platforms

On the other end, information systems (IS) and software in general tend to be inflexible and challenging to modify, requiring a lot of work to keep up with continuously changing requirements, laws, etc. Numerous tools, including IDEs (Integrated Development Environments), modelling tools, DBMS (Database Management System), O-R (Object-Relational) mapping frameworks, GUI (Graphical User Interface) editors, deployment and compilation aids, and so on, are required when using a traditional software development infrastructure. Plus, there is an expanding global trend toward the development of dynamic and adaptive software that may directly execute models or automatically generate code from them. These tools are combined into one system in LCPs, a novel and innovative method to automate business operations, reducing the need to move between them and, more critically, the necessity to integrate and maintain consistency among the implementation artefacts generated by various technologies [5–7].

An LCP is a platform for rapid application development, deployment, execution, and management using declarative, high-level programming abstractions including model-driven and metadata-based programming languages, as well as one-step deployments. User interfaces, business processes, and data services are offered and supported by LCPs. They can drastically cut down on the time and cost associated with establishing, implementing, and sustaining processes. Additionally, because of the increased opportunity to use the company's human resources for research and development tasks, the work's results would more closely match the actual needs of the business [7, 8].

As this project focuses on the usage of LCPs, before presenting, in an upcoming subsection, our own platform used for the purpose of this project, we provide an overview of the main concepts in the LCP area, alongside the drivers and inhibitors for the adoption of LCPs and an overview of a few platforms, that are representative of this emerging trend, so the reader can get acquainted if it's not the case.

Challenges in integrating with other systems and limited functionality are some of the biggest inhibitors from the *Compatibility* dimension [9].

Regarding the *Complexity* dimension, productivity gains are a driver of LCPs, resulting mainly from reducing the efforts of routine tasks in software development projects of low to moderate complexity. This reduction in entry barriers for application development and ease to develop applications applies in several ways. First, the pre-defined, integrated environment provided by LCPs makes synchronizing artefacts produced by previously separate development components easier. Second, productivity is enhanced by the reference implementations of generic tasks such as GUI design, O-R mapping, MVC implementation, and deployment in different environments [5,9]. When it comes to inhibitors in this dimension, the primary ones to the adoption of LCPs are their inflexibility, and limited customization options, as well as their suboptimal usability and insufficient documentation [9].

It is important to note that LCPs are not built on radically new or innovative technologies, but rather well-known tools and components that have been used, in varying forms, for decades. Many LCP products have, in fact, existed for many years and are now simply rebranded [5].

One of the main ways that LCPs are able to reduce the need for conventional coding is through their conceptual modelling components, which are among the most crucial parts of the platform. This strategy is consistent with Model-Driven Development (MDD), which employs conceptual models to bridge the gap between problem and software implementation domains by employing tools that facilitate the methodical conversion of problem-level abstractions to software implementations [5].

This brings us to the conclusion that improved performance metrics of the software development process are one of the main drivers for LCPs when it comes to the *Relative Advantage* dimension. LCPs can lead to cost savings during development, testing, and maintenance, as well as productivity increase, component reuse, and development time reduction. Other referenced drivers are the reduced dependency on IT developers, as it overcomes the lack of hard-to-hire skilled IT developers and eases the burden on IT departments by allowing citizen developers to perform development tasks independently, and quicker reaction to market demand, as the development time is reduced, and it is possible to have a quicker reaction to requirements adjustment, increasing the business' responsiveness [9].

Although LCPs have been influenced by research on MDD, they have not taken advantage of other research-based methodologies such as reference models and DSMLs (Domain Specific Modelling Language). These approaches can substantially improve the efficiency of system design by incorporating domain knowledge. Therefore, it is important to acknowledge the potential benefits of reference models and DSMLs in the development of LCPs [5].

One of the inhibitors regarding the *Observability* dimension is the difficulty in estimating total cost. The pricing models of LCPs vendors typically involve a combination of factors such as the number of users developing with the platform, the applications to be deployed, and the storage

location of the data. However, determining the total cost of using an LCP for an entire organization or department can be challenging [9].

Finally, the usage of LCPs comes with the risk of lock-in effects, with this being the sole inhibitor of the *Trialability* dimension of LCPs [9]. This implies that the models and representations created in one particular LCP may not be compatible with another, leading to a potential dead end in case a vendor decides to discontinue support for their platform. This issue poses a significant threat to investment protection and should be taken into account before committing to a specific LCP [5].

The most popular low-code platform for creating websites is WordPress[1]. This open-source project originally intended to fill the market gap where there was a demand for a classy, well-designed personal publishing system that a user could administer through a drag-and-drop manner. More than 41% of the web today, from basic blogs to business websites, is powered by WordPress. WordPress's easy, extensible framework and ecosystem of plugins and templates are the main reasons for its appeal. With over 54,000 plugins, the amount of customization one can have without having the need to write code is remarkable.

OutSystems[2] is a high-performance, low-code platform that produces significant productivity while continuously innovating to produce serious applications. OutSystems is built from the ground up with the goal of assisting enterprises in creating, delivering, and evolving innovative applications at the speed that business demands. Customers can overcome strategic issues like application modernization, workplace innovation, business process automation, and customer experience transformation with the use of visual development tools and automation powered by Artificial Intelligence. Solutions are engineered to scale, are safe, resilient, and ready for the cloud thanks to OutSystems.

The low-code platform Appian[3] was created for companies of all sizes. Users can create Business Process Management (BPM) applications with it. Case management, BPM, three-step app development, and application integration are important aspects. Users can automate complicated procedures and develop unique apps on Appian's low-code platform, which can be installed on any device. Users can communicate with other team members for project discussions thanks to the social collaboration function. Without writing any code, Appian provides pre-built connectors for connectivity with other programs. Data and analytics are also provided for interactive reports and document management. In addition to on-premises and hybrid deployment choices, Appian also offers cloud-based deployment alternatives. Online case management, phone support, real-time screen sharing, a knowledge base, phone support, and email are all forms of help.

With Budibase[4], customers may quickly create internal tools or business apps by connecting to external data sources like Postgres, MySQL, Oracle, Google Sheets, or Airtable. These apps can be anything from admin panels, portals, and forms to application tracking systems, inventory management systems, and customer service applications. One of Budibase's main selling points is that it is open source and free, giving businesses more flexibility and extensibility while also enabling them to host everything on their own servers. This is crucial for businesses that may want to safeguard sensitive data from third-party infrastructure.

[1] https://wordpress.com
[2] https://www.outsystems.com/
[3] https://appian.com/
[4] https://budibase.com/

1.1.3 DEMO-based Low-Code Platform

An Enterprise Engineering (EE) method called DEMO (Design and Engineering Methodology for Organizations) [10] is linked to a strong body of theories which intend to address the software failure challenges highlighted above. Despite how sound DEMO is in theory, there are still many legitimate concerns regarding its utilization. DEMO's Action Model (AM), which is hardly ever employed in projects, is one of the fundamental components and one of the theoretical foundations that is frequently overlooked in current practice [11]. This occurs even though the methodology's creator himself regards the AM as the most significant model and where all model information is contained in detail [10, 12]. It is regarded as the organization's differentiator model, or what makes it special. From this model one can elicit all other three aspect models of DEMO (Cooperation Model, Process Model and Fact Model).

With this project, we propose a new Action Meta-Model and Grammar for a DEMO-based low-code platform's rules processing engine by evolving the DEMO AM with a new meta-model in the form of an EBNF (Extended Backus–Naur Form) syntax which is currently being implemented in our DEMO-based low-code platform, DISME (Direct Information Systems Modeller and Executer).

We claim that the way Action Rules (ARs) are currently specified in DEMO results in incomplete specifications that maintain ambiguity and do not contain enough ontological information for direct generation of information systems, as claimed by DEMO's propounder. Within our project, we can describe, still on an ontological level, a wider range of crucial details and information, enabling a nearly direct execution of models as an information system. As a result, we help close the enormous gap between DEMO models and the significant implementation issues that surface during the software development process and which should be described right away along with ontological elements. By applying this in DISME, through the execution of models directly, we drastically shorten the time it takes to produce information systems. And thanks to the use of DEMO as our core conceptual foundation, we have, as a starting point, a more complete elicitation of requirements, one of the main reasons Information Systems projects fail. Throughout the document, we demonstrate and validate our contribution using the Rent-A-Car (RAC) case [13], which simulates the operation of a car rental company. Additionally, we further validate the project's efficacy through an experiment and usability tests conducted with the NexusBRaNT case [14].

1.2 Objectives

The core objectives for this project revolve around DISME. As a result, the main goals were to extend the new Action Meta-Model and Grammar for AR specification and develop its DEMO-based low-code platform rules processing engine - the Execution Engine - that controls the flow of information according to the execution of Action Rules, previously defined in their management component through visual programming.

Through the extension of the new action meta-model and grammar for AR specification and development of the Execution Engine, it was also noted that extensions and enhancements to existing DISME components were required. Such is the case of the existing Action Rule design component, since it enforces the AR's grammar and syntax through a visual programming editor based on the Blockly plugin. As such, this objective is of the highest relevance, as the Execution Engine depends on the information generated by this component to work. Another component that required extension and enhancements was the Forms management component, responsible for form creating, editing and translating. This work was of elevated importance as the Execution Engine

relies on these components to provide a way for its users to input data, which is indispensable to every IS.

Another objective that came to be is the data logging on the Execution Engine. To be more exact, this logging happens on every evaluation, whether that is an automatic or user evaluation, or an artefact that has the ability to influence the action flow taken by the engine so that we can always have traceable decisions from this component. Furthermore, it also serves as a valuable resource for auditing and debugging purposes, providing insights and information that can be utilized to address any potential issues or challenges that may arise during the Execution Engine's operation.

With these components developed, extensions and enhancements to the Execution Engine-powered Dashboard component, which serves as the user interface for users to interact with when performing organizational tasks, were made.

When conducting implementation tests with real systems, it was also noted that there was a need for creating a parameterization component to replace the manual insertion of the platform's implemented system artefacts in DISME.

The final objectives of this project concern the validation of the DISME platform by conducting an experiment implementing a traditionally developed system using a low-code approach - to compare the needed effort for both approaches - and a validation process to assess, qualitatively and quantitatively, the usability of the developed platform.

1.3 Implementation

As this project is integrated into a larger research project that aims at the development of an open-source software platform with DEMO methodology as its solid base to battle the IS problems previously enumerated, it is important to understand the fundamentals of DISME.

1.3.1 Architecture

Architectures developed to adapt at runtime to meet changing user requirements by retrieving descriptive information that is interpreted at runtime are typically referred to as *reflective architectures* [15]. These distinguish between the program's base level, which remains constant, and its meta-level, which changes. DISME is based on a particular kind of architectural reflection known as *Adaptive Object-Model architecture* (AOM), accentuating flexibility and run-time configuration.

AOMs are designed differently than the majority of object-oriented architectures. In a typical object-oriented design, classes would be used to describe the various sorts of business entities and would be paired with attributes and methods. A change in the business results in a change to the code, which results in a new version of the program. An AOM system uses metadata to express classes, properties, relationships, and behaviour. Instead of being built on classes, the system is an instance-based paradigm. In other words, the system interprets and stores its Object-Model in a database (DB). Therefore, anytime a business change is required, these descriptions are adjusted, and the updated version of the program is then immediately available [15].

One of the fundamental patterns in developing an AOM is Type Square, which results from the use of the TypeObject and Property patterns. Its name is derived from the class diagram representation of the final layout, as can be seen in Figure 1, with the classes Entity, Entity Type,

Property, and Property Type. Instead of requiring programming for a new class that introduces a new object kind, as is done in classic object-oriented systems, TypeObject turns the unknown subclasses into straightforward instances of a generic class, allowing new *classes* to be created dynamically at runtime. The Property pattern addresses the class attributes differently than the classical approach. Instead of being different instance variables, a single class variable is used to store a collection of attributes. Using this method eliminates the need to modify the source code in order to add (or remove) more information than was originally intended from existing entities. [15, 16].

As can be seen in Figure 1, this approach supports defining model elements and their properties (respectively, the Entity Types and Property Types) as well as allowing for their instantiation (as Entities and Properties) [17].

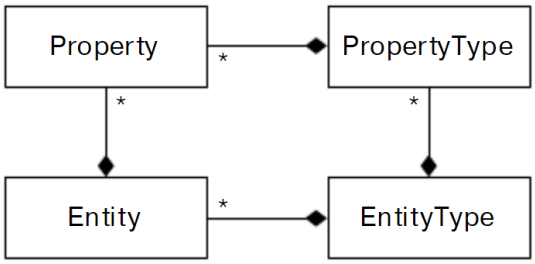


Fig. 1: The Type Square Pattern [17]

The DISME platform is a web application, which means its architecture comprises three core components: the client-side, the server-side and the database server. The client-side, also known as frontend, is the integral part that engages with the user, takes input, controls user interactions with the application, and handles display logic. By allocating the user requests to the appropriate component and overseeing all application operations, the server-side, or backend, comprises a RESTful (Representational State Transfer) API (Application Programming Interface) that manages the business logic and processes the user requests. The data that the program needs is provided by the database server, which manages tasks involving data.

1.3.2 Technologies

As this project is involved in a larger research project that was already being developed before the initialization of this project's scope, the technologies to be used in the different components were already chosen with the general architecture in mind and considering the platform's requirements. Either way, it is important to present them and summarize the reasons why they were selected.

On the frontend, the AngularJS framework was initially selected. It has since then been upgraded to Angular, a ground-up rewrite of AngularJS. Angular is a TypeScript-based, free and open-source web application framework created by Google for the building of SPAs (Single Page Applications). The main reasons for picking this framework were its data binding capabilities, services and dependency injection, custom component usage, extensive community and integration with the selected backend framework [18].

Concerning the backend, the Laravel framework was picked. Laravel is an open-sourced PHP web framework known for its server-side handling of routing, HTML authentication, templating,

and more. The reasons boasting this choice were its stability, authentication and authorization systems, multilanguage functions, security and vulnerability technics, extensive number of contributors and the fact that it is a stable and constantly developing technology [18].

Given that Laravel was selected as the server-side framework, the option for persistent storage would have to be a database type that Laravel effectively supports from scratch using an Object–Relational Mapping (ORM), without relying on potentially out-of-date third-party ORMs. The MySQL database was then chosen, with Laravel having its ORM, Eloquent [18].

1.3.3 Main Components

With the project's architecture and technologies understood, it is important to understand DISME's fundamental components. Three main components primarily make up DISME: 1) a Diagram Editor to create the higher level DEMO models graphically; 2) the System Manager to precisely detail and parametrize all DEMO Models, with special attention to the Action Model, so that a complete information system can be specified according to an organization's demands; and 3) The System Executor to directly run the modelled information system in production mode.

In the System Manager, one or more users assume the administrator role and have the ability to modify each organizational process by creating and editing transactions, their relations, Action Rules and input forms that are associated with these transactions, in specific transactions steps, as well as by specifying the main business objects and their attributes, or, in other words, the database of the information system. Users who model the system just need a basic understanding of Enterprise Engineering modelling, which is similar to the "language/representation" used within businesses, rather than requiring specific programming skills.

Users who have been granted authorization to participate in transactions in the System Executor do so in accordance with their roles and following DEMO's transaction pattern. The System Executor can be broken down into two main components: 1) the Dashboard, which serves as the user interface for users to interact with when performing organizational tasks, and 2) the Execution Engine, which controls the information and process flow following the full specification of the system.

The interface of the Dashboard comprises four main sections, which can be observed in Section 3.5 through Figures 28, 29, and 30. Figure 28 shows where the user can start new processes based on the process types available in the system and their access permissions. Additionally, this figure contains another section that displays the tally of pending and completed tasks, as well as active delegations made by the user.

Moving on to Figure 29, it showcases the Dashboard segments where users can view data related to their pending organizational tasks, such as the creation date, process type, associated transaction type, and state. The tasks are divided into direct and delegated authorization categories, depending on the user's access rights. Furthermore, this is where the Execution Engine will be incorporated, enabling users to execute the tasks exhibited in the table rows.

Finally, Figure 30 exhibits the third and last section of the Dashboard, where users can initiate tasks that belong to existing processes, based on the available tasks in the system and their access permissions.

The development of the database behind the platform's solution was heavily influenced by the DEMO way of thinking, trying to capture the essence of an organization's workflow, but without abstracting from their infological and datalogical implementations. One of the goals of DISME is to keep the platform as flexible as possible in terms of the editing possibilities available [19].

1.3.4 Addressing Inhibitors to Low-Code Platform Adoption in DISME

In relation to the drivers for the adoption of LCPs that have been mentioned earlier, it is worth noting that these drivers are universally applicable to all LCPs, and DISME is certainly no exception to this rule.

DISME has taken into consideration a majority of the inhibitors that hinder the adoption of LCPs, aiming to address most of them and thus facilitate a smoother and more effective adoption of LCPs in the current software development process.

In relation to the integration of DISME with other systems, a component that is specifically designed for efficient RESTful API management is currently in the process of being implemented. This component aims to facilitate the retrieval and provision of information from and to external systems, regardless of whether it requires simple CRUD (Create, Read, Update and Delete) operations or querying for specific results. To further enhance its usability, DISME has developed a highly intuitive graphical interface that simplifies the process of constructing these queries.

In terms of the inhibition of limited functionality, it is observable that DISME has a considerable amount of integrated features, as can be inferred from the previously described details.

It can be said that DISME does not suffer from the drawback of lack of flexibility and customization, as it offers a wide range of options for customization and flexibility. For instance, the Forms design feature allows for customization to meet specific requirements. Additionally, DISME provides the capacity for the detailed design of Action Rules to make customizations at the operational level. Besides, users can design pages by defining queries to ensure that the information presented is up-to-date, making the platform totally flexible and customizable to meet user requirements.

With regard to the inhibitor of usability, DISME offers a solution by employing graphical components to manage everything, including forms, drag-and-drop components, and the diagram editor. The Dashboard, which enables users to perform their daily organizational tasks, was also designed using recent usability patterns and technologies that prioritize usability, such as Bootstrap. As such, DISME provides a user-friendly interface that enables users to easily navigate and interact with the platform's features and functionalities, thus mitigating the usability inhibitor.

As stated later, DISME is not currently available, and therefore, its documentation is not yet provided. However, documentation is a crucial aspect of the platform, and it will receive considerable attention. As an open-source platform, providing thorough documentation will be significant for the platform's adoption as a credible LCP. By being entirely open-source, both DISME and the existing plugins it uses, organizations using DISME also don't face the risk of becoming locked into a specific vendor, mitigating the vendor lock-in inhibitor.

Finally, with respect to the inhibitor of difficulties in estimating the total cost, it should be noted that DISME's usage of the platform will be entirely free of direct costs. Any costs incurred would only be associated with the training required to effectively utilize the platform, if necessary.

1.4 Research Method

According to *Design Science Research* by A. R. Hevner [20,21], the Information Systems Research paradigm used in this study should be viewed as a collection of three closely related cycles of activities.

In Figure 2, these activities are depicted. Hevner argues that these three activities should not be used separately because only together do they provide a solid design science research and can produce a reliable result. Our research, with regard to the first cycle, Relevance, which is depicted in Figure 2, revealed a glaring issue of ambiguity and a lack of concise and crucial information regarding the current syntax of DEMO Action Rules. As a result, an opportunity to design a more comprehensive syntax was at hand. We devised a new grammar for DEMO's Action Rules in relation to the second design cycle. This grammar was developed following numerous iterations of exhaustive and thorough design, implementation, and evaluation of various language elements, as well as testing them in the Execution Engine in our platform using both the Rent-A-Car case and a real-world project being developed. We proposed a new Action Meta Model for DEMO that, in our opinion, will allow the development of Action Rule Specifications in a more thorough and complete manner. Finally, the theoretical underpinnings of DEMO itself provide support for the studies about the final third cycle, Rigor.

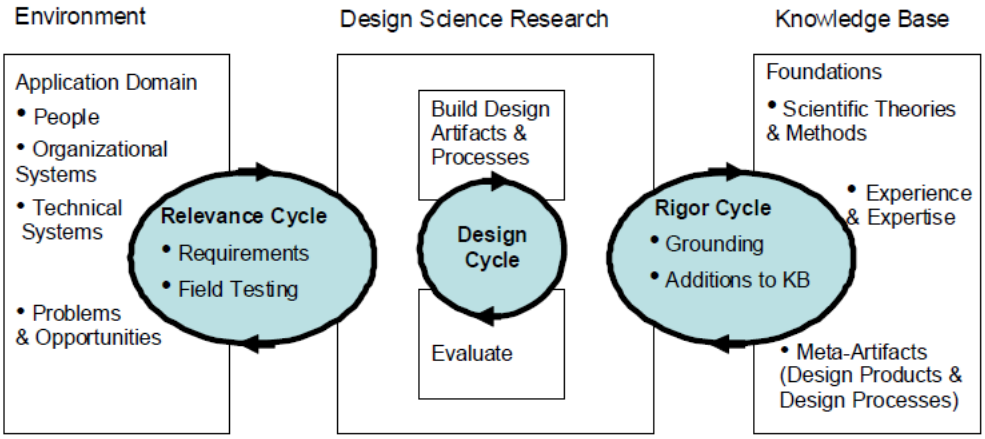


Fig. 2: Design science research cycles [21]

These iterative cycles were not only applied to the development of the meta-model of the Action Rules, but also to the refinement and enhancement of the Execution Engine itself, including its algorithm. It is important to highlight that one of the key contributions of this project lies in the development of this algorithm, which underwent multiple iterations and refinements throughout the implementation process.

Finally, in regards to the Design and Rigor Cycles, three scientific papers were submitted to renowned academic conferences, two of them accepted. The first was focused on the extended grammar, the second on evaluating the needed effort in both development approaches and the third on usability evaluation. We, thus, made concrete contributions to the Knowledge Base and scientific community.

1.5 Outline

This document is organized into 6 chapters. In the current introductory Chapter 1, it is firstly provided this project's context, including its motivation, an explanation of low-code platforms and an overview of the DEMO-based low-code platform being developed. Then, the objectives and implementation details of the project are discussed, providing an overview of this project's architecture, technologies, main components and how it adresses the aforementioned inhibitors to the adoption of low-code platforms, along with the research method employed.

In Chapter 2, context is provided into the domains of Enterprise Engineering and the DEMO methodology. The key concepts of DEMO, such as the Operation Axiom, Transaction Axiom, and Distinction Axiom, are explained, along with DEMO's ontological modelling and action model.

Chapter 3 provides a comprehensive account of the implementation process within the DISME platform. The evolution of the proposed Action Rule Syntax is explained in Section 3.1, while Sections 3.2 and 3.3 elaborate on the necessary adaptations made to the Action Rules Management Component and Forms Component, respectively. The main focus of this project, discussed in Section 3.4, centres around the restructuring, algorithm, and logging of the Execution Engine. Detailed insights into the work accomplished in the Dashboard and Parameterization Components of DISME can be found in Sections 3.5 and 3.6, respectively.

Then, Chapter 4 focuses on a conducted experiment where the NexusBRaNT system was implemented on the DISME platform. The method used is described, and results and discussion surrounding the effort and complexity required to implement an information system using both traditional and low-code approaches are presented.

Having done that, Chapter 5 presents the validation process carried out to assess the usability of the DISME platform. The methods and procedures used for qualitative and quantitative validation are explained, and the results obtained from these validation efforts are discussed.

Finally, Chapter 6 summarises the key findings and contributions of the research. It also outlines potential future work and areas for further exploration in the context of DISME and low-code platforms.

Furthermore, in this document, several chapters are dedicated to presenting the research findings and outcomes in the form of submitted, accepted, and published papers. Specifically, Chapter 3.1 encompasses the contents of a previously submitted, accepted, and published paper named "A New Action Meta-model and Grammar for a DEMO Based Low-Code Platform Rules Processing Engine" in the 2022 Enterprise Engineering Working Conference [22], Chapter 4 encompasses the contents of a previously submitted, accepted, and in the process of publication paper named "Traditional vs. low-code development: comparing needed effort and system complexity in the NexusBRaNT experiment" in the 25th IEEE Conference on Business Informatics [23], while Chapter 5 incorporates the contents of a paper named "Evaluating the Perceived Quality and Functionality of DEMO models' representations and the Usability of a System Implemented on a DEMO based Low-Code Platform" that has been submitted for publication in the 42nd International Conference on Conceptual Modelling.

By having dedicated chapters for submitted, accepted, and published papers, the thesis not only showcases the research contributions of this project but also demonstrates the commitment to academic rigour and active participation in scientific dissemination.

2 Background And Theoretical Foundations

DISME uses DEMO, the principal methodology in Enterprise Engineering, as a solid foundation for the production of collaborative-based organizational models and diagrams for the specification of its processes, information flow, responsibilities of both human and software, procedures and other kinds of organizational artefacts [24].

2.1 Enterprise Engineering [10]

Enterprise Engineering, as can be seen in Figure 3, is an emerging discipline that takes an engineering perspective on enterprises. It seeks to deal with organizational issues that conventional organizational sciences cannot resolve. Changes inside an organization are seen as situations requiring (re)design and (re)implementation.

The overarching objective of EE is to improve businesses, or to improve enterprises, in every sense of the word, by developing new, appropriate theories, models, methods, and other artefacts for the analysis, design, implementation, and governance of enterprises by combining (relevant parts of) management and organization science, information systems science, and computer science. We refer to all organized human activity by "enterprise" (like companies, agencies, institutions, supply chains, etc.).

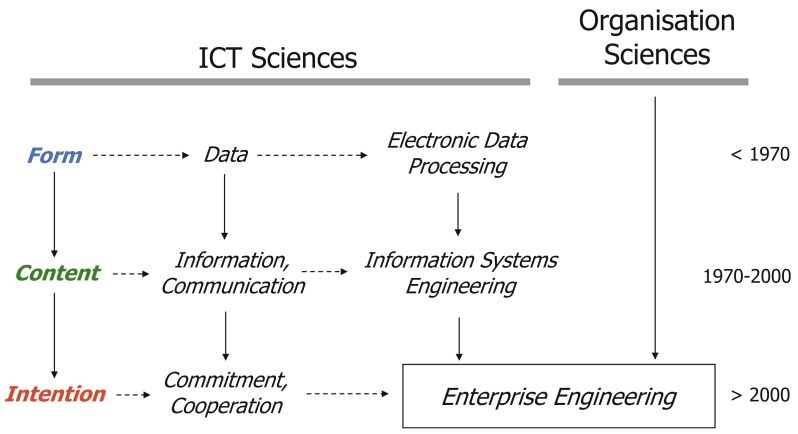


Fig. 3: Enterprise Engineering's roots [10]

The pressing need for an EE discipline in society stems from the fact that contemporary societies are becoming more and more complex networks of organizations with intricate interrelationships, where standard organizational and management sciences are ineffective at offering assistance. Deming's persuasive argument can be summed up as follows: practically all (94%) symptoms of subpar enterprise performance are inescapably caused by how organizations are structured. The implication of this study is that just 6% of operational failures can legitimately be attributed to personnel [25].

The three main objectives for the field of EE are intellectual manageability, organisational concinnity, and social devotion. Intellectual manageability is about understanding complexity and maintaining perspective on it. Designing, engineering, and implementing an enterprise so that the

resulting operational organization is always a cohesive and consistent whole is known as organizational concinnity. Recognizing that an organization's operation is driven by its operational personnel, not by its managers, is what social devotion entails [26].

2.2 DEMO - Design and Engineering Methodology for Organizations [10]

The approach to an organization's ontology presented subsequently is called DEMO, which is supported by the Ψ-Theory. DEMO is an organization modelling methodology for the analysis and representation of organizational processes through transaction modelling. The PSI (Performance in Social Interaction) theory is about the construction and operation of organisations. The word "organisation" indicates that one takes the construction perspective on enterprises. Organisations are systems in the category of social systems, which means that the system elements are social individuals, called actors. The guiding idea is that actors enter into and comply with commitments towards each other. It explains how and why people cooperate, and in doing so bring about the enterprise's business.

2.2.1 Operation Axiom

According to the operation axiom of the Ψ-Theory [13], subjects in organizations execute two different types of acts: production acts that have an impact on the P-world, or production world, and coordination acts that have an effect on the C-world, or coordination world. Subjects are actors performing an actor role responsible for the execution of these acts. These worlds are always in a particular state, indicated by the C-facts and P-facts that have transpired up to that point in time.

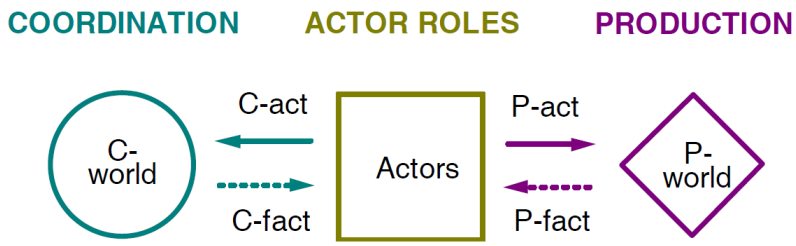


Fig. 4: Interaction of the Actor with the Production and Coordination Worlds [10]

An actor is a subject (human being) in an actor role. The authority and responsibility that the actor may exercise are determined by their role. Coordination acts, which are communication acts, raise commitments. The related coordination fact is produced when a coordination act is completed. For instance, conducting a request act regarding some product results in the fact that the product is requested.

When active, actors consider the status of the P-world and the C-world. Actors continually strive to fulfil the agenda provided by C-facts. In other words, actors engage in interaction through the creation and management of C-facts. Figure 4 depicts this connection between the actors and the worlds. It illustrates the guiding principle of organizations whose members are dedicated to effectively accomplishing their agenda. The coordination actions are how actors enter into and uphold commitments towards reaching a given production fact. In contrast, the production acts

contribute to the organization's objectives by bringing about or delivering products and/or services to the organization's environment [27].

2.2.2 Transaction Axiom

Coordination acts/facts are the atomic building blocks of organisational (but commonly called: business) processes. They always occur in particular patterns of interaction between subjects who play either the initiator role or the executor role in the transaction. These patterns are instances of one generic pattern, called the (business) transaction, in accordance with the transaction axiom of the Ψ-Theory [13]. Examples of process kinds and their respective transaction kinds, in the context of the RAC case, can be seen in Table 1.

Table 1: Process Kinds and associated Transaction Kinds of the RAC case

Process Kind	**Transaction Kind**
Rental	Rental Completing
	Car Taking
	Car Returning
	Deposit Paying
	Invoice Paying
Branch Manage	Branch Creating
Car Manage	Car Creating
Car Type Manage	Car Type Creating
Transport	Transport Managing
	Transport Executing

The terms "initiator" and "executor" replace the colloquial terms "client" and "producer". In addition, rather than subjects, these terms refer to actor roles. An actor role is described as having the authority and responsibility to carry out a particular type of transaction. Subjects play actor roles in such a way that multiple subjects can play one actor role and one subject can play multiple actor roles. In general, actor roles cannot and should not correspond directly in name or content to organizational positions or functions. As a result, whenever possible, the names of these roles shall be modest variations of the names of the executed transactions. For example, in the RAC scenario, the transaction type "Rental Completing" will be executed by the "rental completer". An actor role should, in theory, be played by the same subject for all actions that fall under that actor's purview, namely the initiator's request and acceptance and the executer's promise and declaration. By having a precise description of responsibilities, it is possible to have a better and more informed conversation about what the actual requirements for organizational roles or positions should be, as well as to map actor roles to these roles and, ultimately, to map actual people to the stated roles [18].

Three phases make up the transaction pattern: (1) the order phase, where the initiating actor role of the transaction expresses his wishes in the form of a request and the executing actor role promises to produce the desired result; (2) the execution phase, where the executing actor role

produces the desired result; and (3) the result phase, where the executing actor role states the produced result and the initiating actor role accepts that result, effectively closing the transaction. This succession, which can be seen in Figure 5, is referred to as the "basic transaction pattern", and only takes into account the "happy case", in which everything proceeds as predicted. To realize a new production fact, all five of these steps are essential.

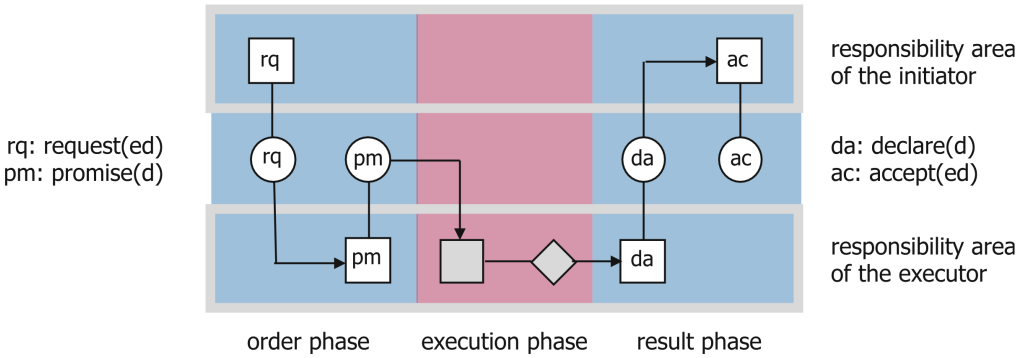


Fig. 5: The basic transaction pattern [10]

Figure 6 represents an extension of the basic pattern shown above, where the basic transaction process is represented by the green path. The yellow paths depicted here represent cases where the pattern diverges to discussion states (dc and rj). These are represented by a double disk because ending up there is most likely not what the initiator had in mind. Therefore, in the left yellow path's case, the executor is challenged to explain why the request was not accepted, and the initiator gets the chance to counter the executor's arguments and talk about potential adjustments to the product's properties. In the RAC case, the rental completer could have declined the request for a Rental Completing because the provided driver's licence isn't valid. In this case, the client performing the request can give up the rental or perform an adapted request with a new submission of a driver's licence. This is shown in Figure 6 by the left yellow path. The executor can then perform the promise of the adapted request, thus following the basic transaction pattern in the green path. It should be noted that the cardinality range 0..1 indicates that executing a new request is optional. The procedure continues in the declined state, and the process is possibly cancelled if the initiator chooses to do nothing in its place. The initiator may also reject the declare state, which brings the transaction process to the rejected state, where the logic discussed for the left-side yellow path is applied in the same way.

The second extension of the basic transaction pattern consists in adding four revoke patterns, one for each of the main steps of the transaction (request, promise, declaration, and acceptance). In other words, from any state within the main transaction process, both the initiator and the executor can undo any fundamental step they have already taken. They can also repeatedly revoke a step inside a single transaction. Revoking a step indicates that one wants to reverse a deliberate action taken previously because one has changed one's mind. It is possible to cancel a mistaken action (as long as the addressee has not responded).

For a better understanding, let's resort to the RAC case to explain the process of revoking a request act, that can be seen in Figure 7. After submitting a request for a rental completing containing a car of the economic type, the renter realizes that a convertible car is a better choice

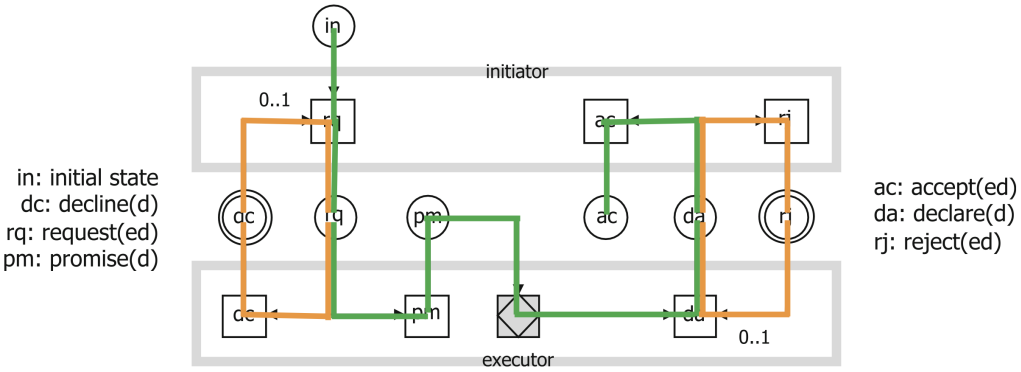


Fig. 6: The standard transaction pattern [10]

because of the weather. Therefore, he expresses his wish to the rental completer. He will then evaluate the renter's desire, which will result in the rental completer refusing (yellow path) or allowing this revocation of the request (blue path). If it is allowed, a new request will be submitted with the updated car type and the process will continue (green and white path). If refused, the transaction will move to the refused state and the process will be possibly cancelled. Similar thoughts can be conceived for the revoking of the other three main steps of a transaction.

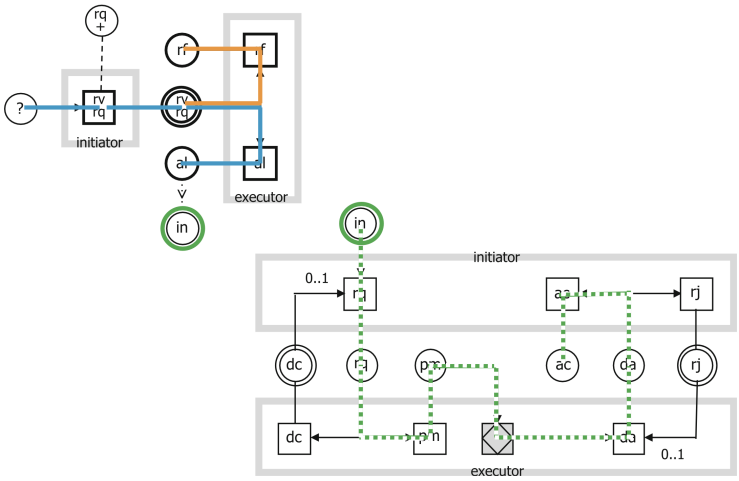


Fig. 7: The process of revoking a request act [10]

The basic transaction pattern with these two extensions results in the complete transaction pattern, shown in Figure 8. This pattern is considered to be a socio-economic law: all transactions in every type of organization follow paths of acts along this pattern.

Every transaction (instance) is of a particular transaction kind. A transaction kind concerns one specific product kind and has one specific actor role as its executer role. All transactions go through the four social commitment coordination acts of request, promise, state, and accept; however, these steps might be taken tacitly, that is, without any kind of explicit communication taking place. Particularly, nodding or similar non-verbal behaviour is frequently used to realize the promise and acceptance. This could occur as a result of the adage "no news is good news" or just plain forgetfulness, both of which can seriously damage a business. In situations of failure, this

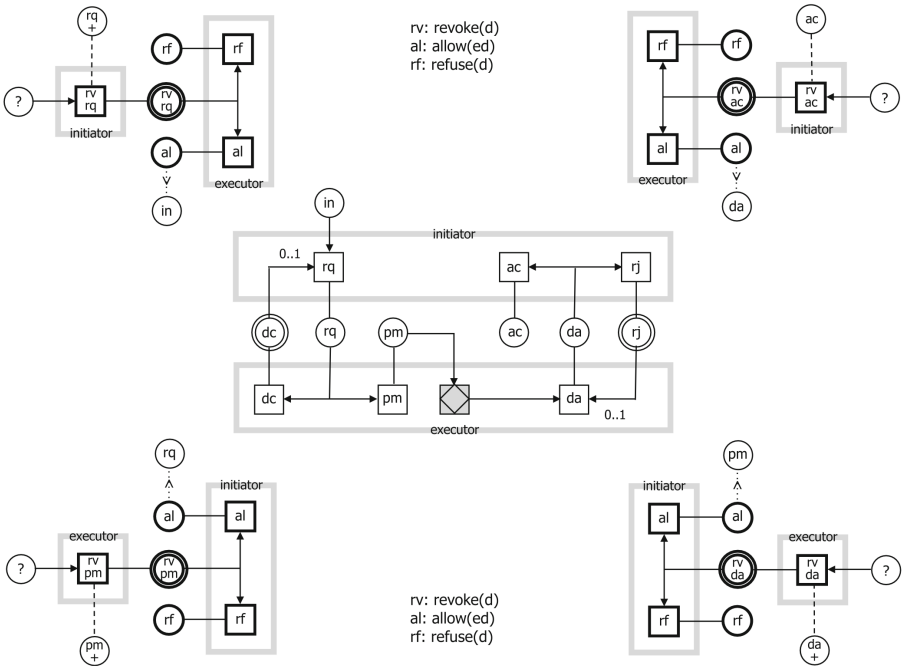


Fig. 8: The complete transaction pattern [10]

becomes more important. All coordinating actions should be carried out explicitly to minimize misunderstandings. Therefore, it's crucial to always take the complete transaction pattern into account while designing organizations. The responsible actor may also not carry out these steps because the relevant subjects may delegate one or more of the transaction steps that fall under their purview to another subject, even if they are still ultimately liable for such acts [18, 27].

The abstraction made by the Ψ-theory to arrive at the ontological model of an organization is the application of the transaction axiom, resulting in a huge reduction in complexity of about 70%, at the documentation level [18].

2.2.3 Distinction Axiom

To fully understand an organization's operation essence, one must understand the three human abilities that are distinguished in performing coordination acts: forma, informa, and performa. This distinction gives rise to three levels of correspondence in the communication between subjects: the forma level (notational correspondence), the informa level (cognitive correspondence), and the performa level (social correspondence), as shown in Figure 9. To be successful, all three conditions of correspondence must be satisfied, that is, the communication must be free of distortion. The medium level, which is below the forma level, is where forms are encoded in tangible materials and transferred between people. Although equally necessary for effective communication, this level is seen as being outside the purview of EE.

As can be seen in Figure 10, in the forma level, we have the ability to act at the formative level of coordination, like uttering (speaking, writing) and perceiving (listening, reading) sentences, as well as to perform documental production, like storing, retrieving, transmitting, and copying sentences. The ability to act at the informative level of coordination, like expressing facts (formulating) and educing facts (interpreting), as well as to perform informational production, like remembering and

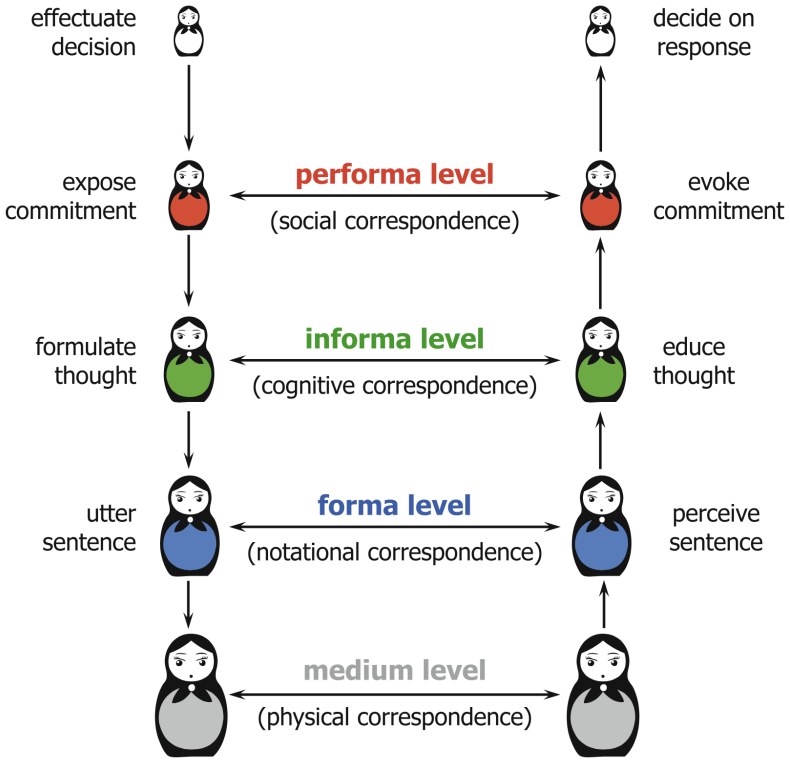


Fig. 9: The process of a communicative act [10]

recalling facts, and computing new facts from existing ones refers to the informa level. The final upper performa level is where one has the ability to act at the performative level of coordination, like exposing commitment (by the performer) and evoking commitment (by the addressee), as well as performing original production, i.e. creating new facts by deciding, judging, manufacturing or observing things in the physical world.

When it comes to production, the forma ability concerns the documental or datalogical production (store, copy), the informa ability involves the informational or infological production (deduce, compute, calculate) and the performa human ability concerns the essential production in an organization (create, decide, judge), therefore also called ontological production [18].

The abstraction made in the Ψ-theory with the axiom of distinction makes it so that to arrive at the ontological model of an organization only the performa skill with respect to production is considered, thus abstracting from production acts at the datalogical and infological level. This results in a second major reduction in complexity, estimated also at about 70% in terms of documentation [18].

2.2.4 DEMO Ontological Modeling

Like every proper methodology, DEMO comprises a Way of Thinking (WoT), a Way of Modelling (WoM), and a Way of Working (WoW). The WoT consists of a couple theories, with the main one being the Ψ-Theory. The WoM consists of an integrated whole of four aspect models: the Cooperation Model (CM), the Action Model (AM), the Process Model (PM), and the Fact Model (FM). These models, depicted in Figure 11, constitute the complete ontological model of the business organization and subsequently represent the corresponding entity's ontological model.

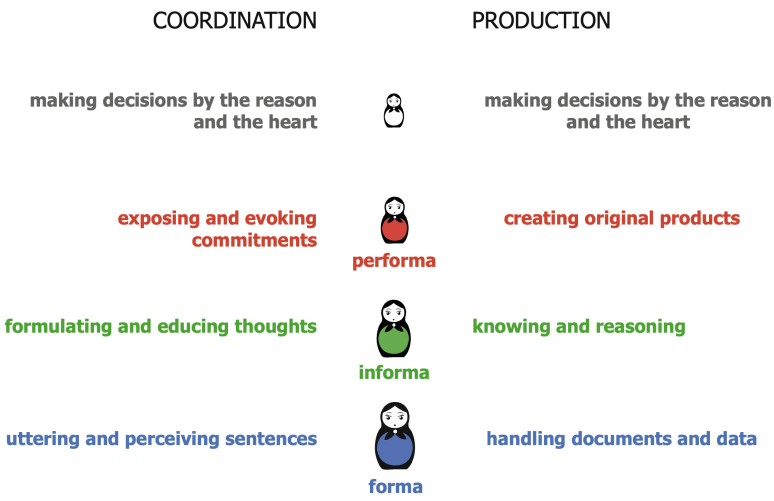


Fig. 10: Human abilities in coordination and production [10]

Although a brief explanation is given below for each one, we will be focusing mainly on the Action Model, since it is the one that this project has a direct impact on.

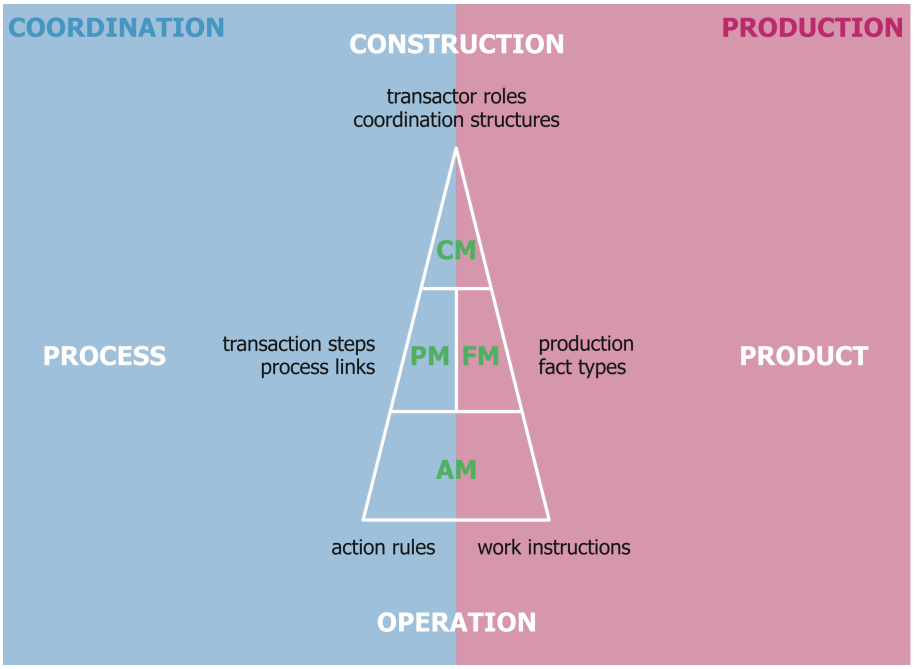
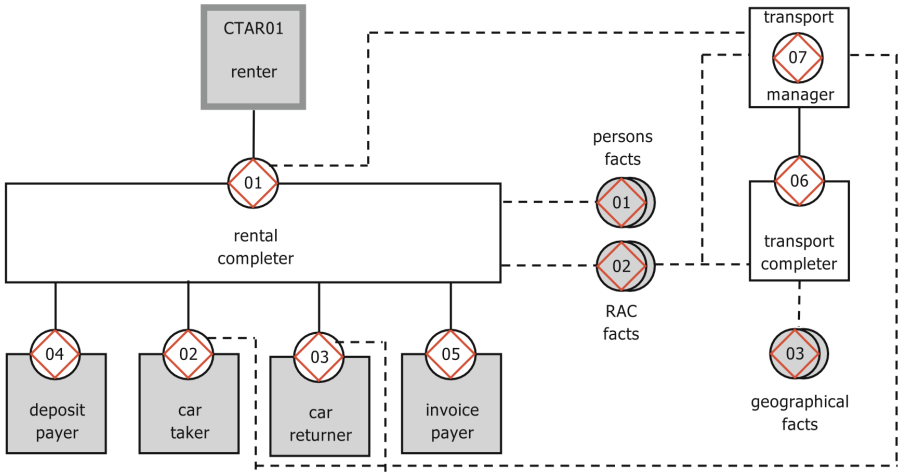


Fig. 11: The integrated DEMO aspect models [10]

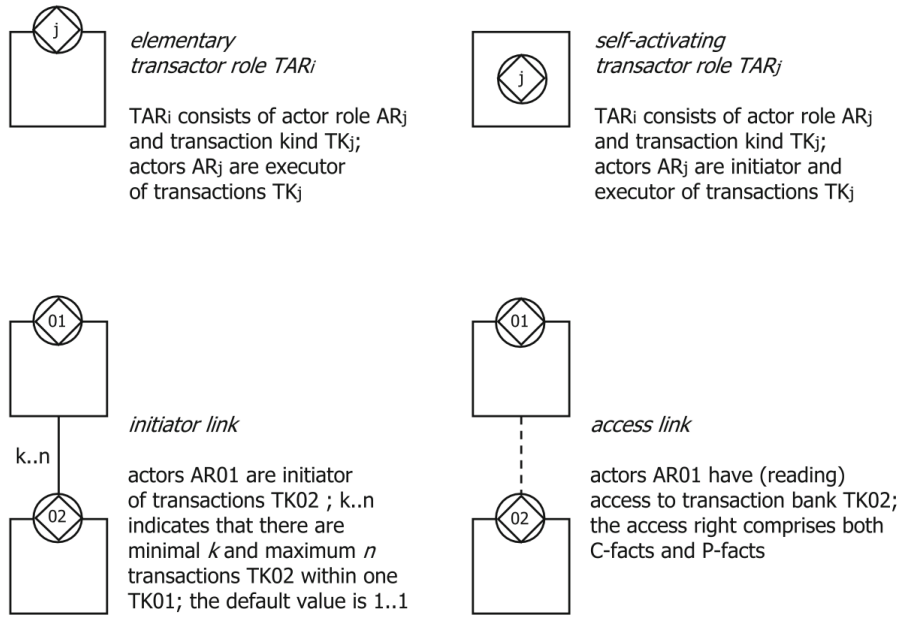
Every time we refer to an organization, a specific Scope of Interest (SoI) is intended. An SoI may include (a portion of) a business or (a portion of) a network of businesses.

The CM of an organization is the ontological model of its construction, thus of the identified transactor roles and the coordination structures among them. A CM is expressed in a Coordination Structure Diagram (CSD) and a Transaction Product Table (TPT), possibly supplemented by a Bank Contents Table and a Bank Access Table. The CM of the RAC case is depicted in Figure 12 and Table 2. To enhance comprehension, Figure 12b presents the legend corresponding to the

CSD represented in Figure 12a. Within Table 2, the abbreviations TK, PK, and AR stand for Transaction Kind, Product Kind, and Actor Role, respectively.



(a) Coordination Structure Diagram of the RAC case



(b) Legend of the Coordination Structure Diagram

Fig. 12: Coordination Structure Diagram of the RAC case [13]

The PM of an organization is the ontological model of the state space and the transition space of its coordination world. It contains the existence laws and occurrence laws for all internal and border transactor roles. The PM connects the CM and the AM of an SoI as far as coordination is concerned. A PM is expressed in a Process Structure Diagram, optionally supplemented by a number of Transaction Process Diagrams and a Create Use Table. The Process Model of the Rental process in the RAC case can be seen in Figure 13. To enhance comprehension, Figure 13b presents the legend corresponding to the CSD represented in Figure 13a. The wait link between (T04/ac) and [T02/rq] is a policy requirement set by Rent-A-Car. It specifies that the rental car can only be taken after the deposit has been paid. The wait links between (T02/ac) and [T03/rq], as well

Table 2: Transaction Product Table of the Rental Process in the RAC case [13]

Transaction Kind	**Product Kind**	**Executer Role**
TK01 Rental Completing	**PK01** [rental] **is** completed	**AR01** rental completer
TK02 Car Taking	**PK02** **the** car **of** [rental] **is** taken	**AR02** car taker
TK03 Car Returning	**PK03** **the** car **of** [rental] **is** returned	**AR03** car returner
TK04 Deposit Paying	**PK04** **the** deposit **of** [rental] **is** paid	**AR04** deposit payer
TK05 Invoice Paying	**PK05** **the** invoice **of** [rental] **is** paid	**AR05** invoice payer
TK06 Transport Executing	**PK06** [transport] is executed	**AR06** transport executer
TK07 Transport Managing	**PK07** transport managing **for** [day] **is** done	**AR07** transport manager

as between (T03/ac) and [T05/rq], are necessary for logistical reasons. The first link states that a car must be taken before it can be returned, while the second waiting link indicates that the rental car must be returned before the final invoice can be prepared and payment can be requested [13].

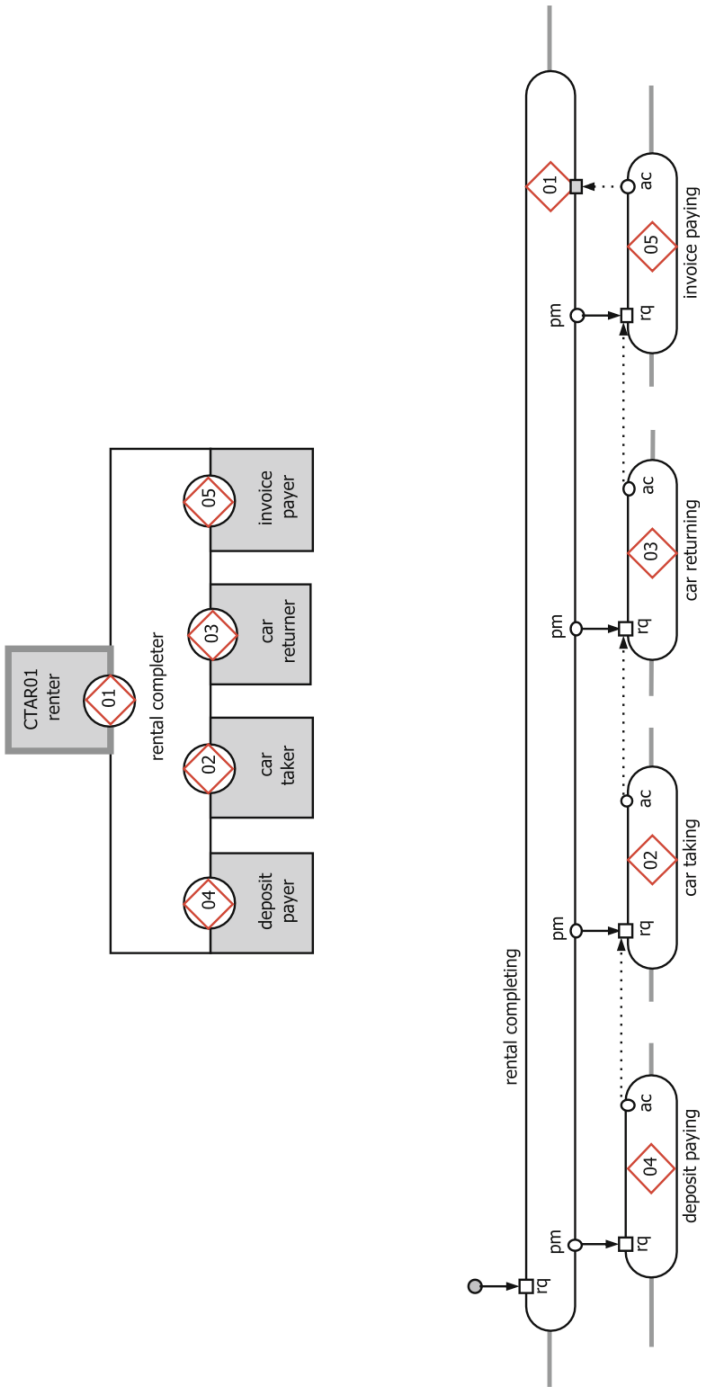
The FM of an organization is the ontological model of the state space and the transition space of its production world. It contains the existence laws and occurrence laws for all identified entity types, value types, property types, attribute types, and event types. An FM is expressed in an Object Fact Diagram, supplemented by Derived Fact Specifications, and optionally supplemented by Existence Law Specifications.

2.2.5 Action Model

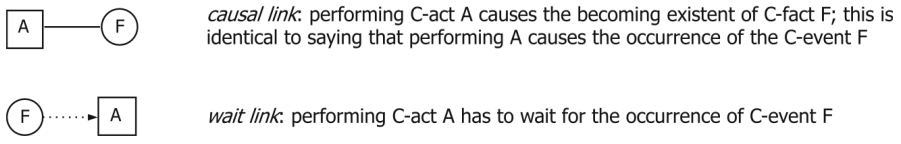
The AM of an SoI is the ontological model of its operation. For every internal actor role, it provides the rules that guide the role fillers in doing their work. The guidelines for responding to coordination events are called Action Rules (similar to business rules), the ones for performing production acts are called work instructions. An AM is represented by Action Rule Specifications and Work Instruction Specifications. The first ones guide actors in performing coordination acts, the second ones guide them in performing production acts. The AM is the solid foundation on which the other three models are standing. In a sense, they are already "contained" in the AM, they only need to be "extracted". Lastly, there is nothing "above" the CM.

DEMO Action Rules are the guidelines for managing events to which actors must react, or business rules. The Action Model of DEMO is not comprised by this set of rules alone, but also contains work instructions regarding the execution of production acts, both represented in the Action Rules Specification (ARS) [12]. Because work instructions are usually enterprise-specific, we will not elaborate on them. One should consider them as detailed instructions for accomplishing a certain production task, like the concluding of a rental contract. The Action ARS standard has evolved through time, starting with a pseudo-algorithmic language and culminating, in DEMO's specification language 4.5, in a definition which adheres to the EBNF, the international standard syntactic meta language, defined in ISO/IEC 14977 [28].

These rules are sometimes referred to as business rules in modern usage. Every form of agendum that actors in a particular actor role must deal with when looping around their actor cycle has, in theory, a corresponding Action Rule. The "exception" states (declined, rejected, and the states in the revocation patterns) are mostly ignored because the response is much too dependent on the



(a) Process Structure Diagram of the Rental Process in the RAC case



(b) Legend of the links in the Process Structure Diagram

Fig. 13: Process Structure Diagram of the Rental Process in the RAC case [13]

topical situation. One must at least produce the Action Rules that correspond with the coordination events in the basic transaction pattern, as seen on Figure 5.

The general form to represent an Action Rule is <event part> <assess part> <response part>. Figure 14 shows an example of this Action Rule specification form. What event (or collection of concurrent events) is reacted to is specified by the event part. An Action Rule's assess portion is divided into three sections that correspond to the three validity claims: the claims to rightness, sincerity, and truth. The final section, the response, is broken down into an if-clause that outlines what must be done if the actor believes complying with the event is justifiable and, potentially, what must be done if it is not. This method of developing Action Rules enables the performer to stray from the "rule" if they believe it is acceptable, while also being held accountable for it [12].

when	membership starting **for** [membership] **is** requested		(TK01/rq)
	with	**the** member **of** [membership] **is some** person	
		the payer **of** [membership] **is some** person	
		the starting day **of** [membership] **is some** day	
assess	*rightness:*	**the** performer **of the** request **is the** member **of** [membership]	
		the addressee **of the** request **is a** membership starter	
	sincerity:	* the member complies with the Volley Regulations *	
	truth:	**the** starting day **of** [membership] **is the** first day **of some** month;	
		the age **of the** member **of** [membership] **on the** starting day **of** [membership]	
		is equal to or greater than the minimal age	
		in the year **of the** starting day **of** [membership];	
		the number of members **on the** starting day **of** [membership] **is less than**	
		the max members **in the** year **of the** starting day **of** [membership]	
if	*performing the action after* ***then*** *is considered justifiable*		
then	promise	membership starting **for** [membership]	[TK01/pm]
	to	**the** performer **of the** request	
else	decline	membership starting **for** [membership]	[TK01/dc]
	to	**the** performer **of the** request	

Fig. 14: Example of an Action Rule specification [10]

We consider this way of ARS to be ambiguous because, despite using a structured English syntax akin to that found in Semantics of Business Vocabulary and Rules [29], it does so in an imprecise manner that lacks some necessary ontological details to be used as the basis for the implementation of an information system. For instance, it lacks a method to deal with sets of actions or operators. Additionally, the current standard brings unneeded complexity since it includes a lot of extraneous details about three different forms of evaluation: fairness, sincerity, and truth.

The following section, in which we go into more detail about our proposal, will develop these claims.

3 Implementation

This section focuses on the practical realization of various components within the low-code platform. It encompasses the development and update of essential elements that contribute to the platform's functionality and effectiveness. The following subsections discuss each component in detail, highlighting their significance and contributions to the overall goal of the platform.

3.1 New Action Rule Syntax Specification and Implementation

In Table 3, we present, in EBNF, the current result of our iterations of development of a syntax and constructs specification of DEMO Action Rules which are runnable.

We begin, then, with the definition of EBNF, which tells us that in computer science, the Extended Backus-Naur Form is a metasyntax notation, which can be used to express a context-free grammar. EBNF is used to make a formal description of a formal language [28]. All EBNF rules can be found in [28], but in order to understand Table 3, let's introduce the most commonly used rules in this grammar. "( )" means grouping. "{ }" is for optional elements (zero or more times). "|" means alternative. "-" means Syntactic Exception. Separates a rule which must be used (on the left) from the rule describing what is not allowed to be used (on the right) ("Consonant = Letter - Vowel;"), but if there's nothing on the right, it can be interpreted only as something that must be used ("OneOrMore = {Something}-;"). Elements with uppercase or between quotes mean a terminal symbol with specific behaviour assigned to the system. Furthermore, in the case of Table 3, column separation equals the EBNF symbol "=".

We next introduce how this grammar corresponds to a set of requirements for the respective implementation of DISME's engine that runs the Action Rules, and consequently, all the logic used for the implementation of their visual programming editor. In this specification presented in the table below, in relation to its previous version, new concepts are highlighted in bold and updated ones are in italic.

Extension and improvements of the already existing EBNF for Action Rules syntax resulted in greater compliance with the EBNF syntax rules, greater modularity in the present components, an overhaul, and expansion of past concepts like the *causal link* action type, and the introduction and definition of new concepts such as the *edit entity instance* action type. In comparison with its first iteration [19], only about twenty percent of concepts remain similar, like the *property*, *if*, *while* and *for* actions' definition. This grammar is an ongoing process and is deemed to keep expanding.

An Action Rule occurs in the context of a transaction step, that is, a transaction type, among those specified in the system, in the activation of a particular transaction state. An Action Rule can lead to the execution of one or more actions of a specific type. For example, an action may imply a *causal link* - generating a new transaction instance or changing the transaction state of the current transaction - or it may be of type *assign expression* and simply designate a value to a property in the system. We can have a sequence of one or more actions. For each action, one needs to specify the action type that will imply what concrete operations/instructions will be executed by the Execution Engine and then define its parameters, specific to the corresponding action type, required for its execution.

Table 3: DISME's Complete Action Rule's EBNF Syntax

when	WHEN transaction_type IS | **HAS_BEEN** transaction_state { action }-
transaction_type	STRING NOTE: has to be a transaction_type specified in the system.
transaction_state	INITIAL | REQUESTED | PROMISED | EXECUTED | DECLARED | ACCEPTED | DECLINED | REJECTED | STOP | QUIT | REVOKE_REQUEST_REQUESTED | REVOKE_REQUEST_ALLOWED | REVOKE_REQUEST_REFUSED | REVOKE_PROMISE_REQUESTED | REVOKE_PROMISE_ALLOWED | REVOKE_PROMISE_REFUSED | REVOKE_DECLARE_REQUESTED | REVOKE_DECLARE_ALLOWED | REVOKE_DECLARE_REFUSED | REVOKE_ACCEPTANCE_REQUESTED | REVOKE_ACCEPTANCE_ALLOWED
action	causal_link | assign_expression | user_input | **edit_entity_instance** | user_output | produce_doc | if | while | foreach | *API_CALL*
*user_* output	STRING NOTE: special HTML code defined in the template manager component, dashboard will output this to the user interface.
produce_doc	static_template | form_template
static_template	STRING NOTE: special rendered HTML code annotated with custom directives from which a PDF is generated.
form_template	STRING NOTE: special rendered HTML code annotated with custom directives from which a PDF is generated. The interpretation of some directives prompts the end-user at runtime for additional input (e.g. an observation to be placed in the PDF).
assign_expression	property "=" term | property_value
property	STRING NOTE: has to be an existent property specified in the system.

causal_ link	transaction_type MUST BE transaction_state [min [max]] **[CANCEL_PROC] [CONTINUE_IF_SAME_USER]** NOTE: min max are optional and by default come with 1 as prefilled; if min doesn't exist, by default = 1. If max doesn't exist, by default = min. Cancel_proc refers to whether the causal_link cancels the current process. Continue_if_same_user marks whether the execution engine should take the user directly to the execution of the causal_link task when it reaches this action, in case the user is an executor of the created task.
min	Integer
max	Integer | *
user_ input	{ form_property }-
edit_entity_instance	{ entity_detail } { form_property | form_ent_type }- NOTE: properties inserted here must have the flag 'editable' as true.
form_property	property [form_calculation] [enable_condition] { validation_condition }- [MANDATORY] NOTE: mandatory is a checkbox in the block that specifies if the filling of the property should be mandatory in its respective form.
form_ent_type	LINKING_PROPERTY { form_property }- NOTE: one_to_many or many_to_many ent_types only. Linking_property refers to which property is assigned automatically by the engine (ex.: property Car in Car has Feature'). Form_property will be all of the remaining properties from this ent_type.
entity_detail	property NOTE: properties specified here are to be shown in the entity selection modal select box in 'edit entity instance' actions.
form_calculation	compute_expression NOTE: used when we want that the value assigned to a property is automatically computed based on some expression which will fetch data either from values in properties on the current form itself, or properties from the database (currently only supports properties included in the current form).
enable_condition	ENABLE condition NOTE: this is used when we want that a property is "hidden/disabled" from the form unless the specified condition is true, which in that case the property will be shown.
validation_ condition	[NOT] validation_condition_type [EXTRA_FIELD_1] [EXTRA_FIELD_2] [user_output] NOTE: The not, extrafield1 and extrafield2 fields will appear depending on the validation_condition_type chosen. EXTRA_FIELD_1 and EXTRA_FIELD_2 examples are the min and max fields if the validation_condition_type is "Belongs Range".

validation_condition_type	REQUIRED | IS_NUMBER | IS_INTEGER | EQUAL_TO | MAX_WORD_LENGTH | LESS_EQUAL | HIGHER_EQUAL | HIGHER_THAN | LESS_THAN | MIN_LENGTH | BELONGS_RANGE | MAX_LENGTH | MIN_WORD_LENGTH | MAX_WORD_LENGTH | HAS_CHARACTER | HAS_WORD | IS_EMAIL | IS_URL | CUSTOM_VALIDATION | REG_EXPRESSION
if	IF condition THEN { action }- [ELSE { action }-]
condition	(ISTRUE | NOT evaluated_expression | condition) | (AND | OR { evaluated_expression | condition }-) NOTE: if condition is of type ISTRUE engine will evaluate the expression; if condition is of type NOT, engine will evaluate either the expression or condition; if it is of type AND or OR it will accordingly evaluate the specified expressions/conditions
evaluated_expression	comp_evaluated_expression | user_evaluated_expression
comp_evaluated_expression	term logical_operator term | property_value
user_evaluated_expression	STRING NOTE: this is a simple text input and dashboard shows this "textual informal expression" that has to be evaluated by the user who will decide on a result of true or false.
logical_ operator	"<" | ">" | "==" | "!="
property_value	STRING NOTE: must be a possible value of a property with 2 cases: 1) value_type is enum and one can select all allowed values for the selected property 2) value_type is prop_ref where possible values to select will be the values associated with the property fk_property_id (e.g., property name of entity type location has many values for the different locations one can pick-up or drop-off rentals).
term	constant | value | property | query | compute_expression | produce_doc
constant	**value_type** STRING NOTE: Can be a constant defined on the system, or can be a new specification of a constant. In the latter case, the name, value and value type of the constant to be created must also be specified.
value	**value_type** STRING NOTE: free value inserted when editing the Action Rule. Must also specify the value type when specifying the new value.
value_ type	*TEXT* | INTEGER_NUMBER | REAL_NUMBER | BOOLEAN | **ENUM** | **DATE** | **TIME**

query	STRING { term } NOTE: has to be an existing query specified in the query table. It will have terms if it's a dynamic query where the user can specify query parameters.
compute_expression	term { compute_operator term }-
compute_operator	"+" | "-" | "×"|"/"|"^"
while	WHILE condition { action }-
foreach	FOREACH set { action }-
set	SET_OF_ELEMENTS

An action of type *user input* can be specified that will prompt the user for input through a form, that is, for the user to input some data for a certain process instance. This form will be designed in the form management component of DISME, shown in Figure 15, according to the properties associated with the respective action. It is also possible to specify, for each property in the form, *enabling conditions*, *validation conditions* and *form computing*. *Enable conditions* are used when we want that a property is hidden/disabled from the form unless the specified condition is true, which in that case the property will be shown. *Validation conditions* have to be satisfied/validated so that the user can submit the form data, being that if the condition is not satisfied, an error message is presented back to him. *Form computing* enables us to define computations regarding data in the current form for a specific field, with that property being filled automatically based on the given expression instead of a manual fill by the user.

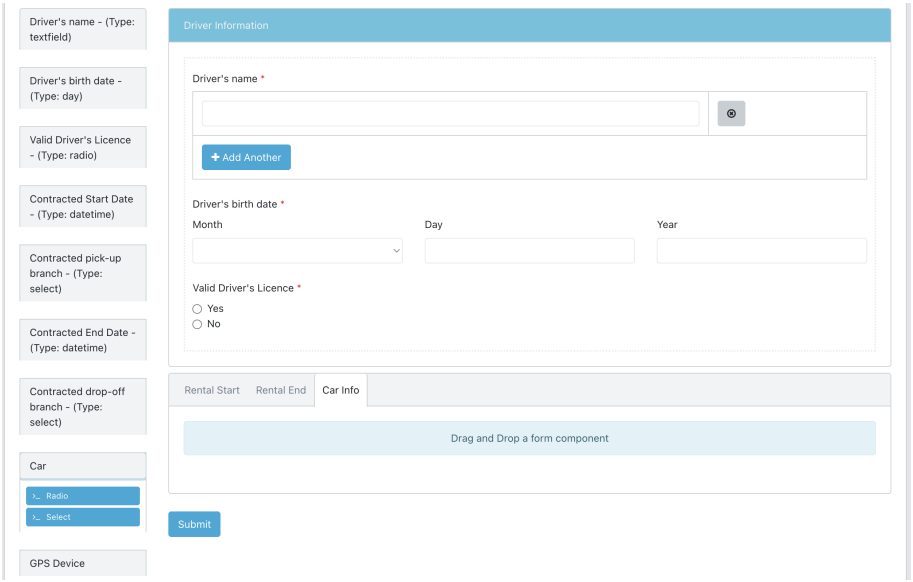


Fig. 15: DISME's Form Editor

As opposed to the last action type mentioned, one can also define actions of type *user output* that will output information to the user. Using a WYSIWYG (What You See Is What You Get) editor to create a new template or selecting an already saved template from the system's database, we can output a custom notification or dialogue box directly to the user when the Action Rule is run. The possibility to add properties to this editor, whose value is filled in the running of the

Action Rule, thus making this a dynamic template, is currently being implemented in another feature of DISME and is set to be reused in these ARs at a later date.

It is also possible to specify *if then else* flows, as can be seen in the AR example provided in Figure 16, and in the condition one can specify complex rules containing logical conditions evaluated automatically by the Execution Engine or informal expressions evaluated by the human user responsible for the transaction step as true or false, or a combination of both.

While kind of flows are already supported but aren't fully specified, lacking specification in our grammar to prevent infinite loops from happening in their execution. *For each* kinds of flows are not yet implemented and also have this same problem, but both are planned to be included in a future iteration of DISME's Action Rule's Syntax. The *set of elements* can be a group/array of elements that can be obtained from a customized query that returns a set of elements from the internal and/or external information system.

The terminal symbols presented as *STRING* are automatically parsed and interpreted by the Execution Engine of DISME.

Actions of type "produce doc" and "api call", although being interconnected with the definition and execution of Action Rules, have its definition and development inserted in other projects at DISME by other colleagues, and therefore fall out of this project's scope and will not be further extended.

In the following subsections, the most relevant innovations in the new Action Rule syntax specification will be discussed, followed by the presentation of an example regarding an Action Rule definition.

3.1.1 Relevant Innovations

An important innovation in the Action Rules syntax is the realization that one needs to decompose a "normal" Action Rule into two Action Rules for each transaction step, one regarding the act itself and another for the respective fact created. This duality is achieved with the usage of the **IS** and **HAS_BEEN** terms when defining the root of the Action Rule, as can be seen in the first row of the EBNF table. We came to this realization while noticing that an actor, while executing a certain c-act or the p-act itself, will need to create some original fact(s) (e.g. input in a form while executing a request); and while dealing with a c-fact/p-fact having been executed, it might be needed to have complex conditions evaluation and also new facts creation or computation. It is also worth noting that the responsible role for the *HAS_BEEN* Action Rule is the opposite to the one responsible for the *IS* Action Rule while in the same transaction state, that is, if it is the initiating role of the transaction that is responsible for the *IS* Action Rule, it will be the executing role that will be responsible for the *HAS_BEEN* Action Rule of the same transaction state, and vice versa. This allows an even more clear separation of responsibilities in DEMO models.

Another relevant addition to this syntax is the inclusion of the new **edit entity instance** action type. By carrying out demonstrations and trials of DISME's usage on information systems in a real scenario, it became apparent that an action for editing previously filled data, more specifically entity instances, was needed, especially in a data intensive, and not so process intensive, information system. With this action, the user can specify Action Rules that comprise the modification of editable properties, that is, properties that have the *editable* flag active, belonging to entity instances created formerly in the current process instance. Also, properties, or *entity details*, can

be specified to be shown in the entity selection modal's select box that will appear on the execution of this action type, in order to give context and facilitate the selection. The transaction type *Edit Car Information* is an illustration of this. It allows one to change properties like a car's rental pricing and mileage. When creating a transaction instance of this type, the vehicle being edited has to be chosen from a dropdown list. Here, the entity details are the chosen characteristics that would be listed underneath each option, such as its colour, to further specify which Car it belongs to, allowing the user to choose the appropriate option, for instance, if there were two identical cars.

```
01  WHEN 'Car returning' HAS_BEEN declared
02  IF ['car is damaged']
03  THEN
04    ASSIGN_EXPRESSION 'car damage' = true
05    CAUSAL_LINK 'Damage Handling' [must be] requested
06  ELSE
07    ASSIGN_EXPRESSION 'car damage' = false
08  IF ['current date' > 'contracted drop-off date']
09  THEN
10    ASSIGN_EXPRESSION 'late return penalty' = true
11    ASSIGN_EXPRESSION 'late return penalty charge' = EXPRESSION
12  ELSE
13    ASSIGN_EXPRESSION 'late return penalty' = false
14  IF ['Actual drop off branch' == 'Contracted drop-off branch]
15  THEN
16    ASSIGN_EXPRESSION 'location penalty' = false
17  ELSE
18    ASSIGN_EXPRESSION 'location penalty' = true
19    ASSIGN_EXPRESSION 'location penalty charge' = EXPRESSION
```

Fig. 16: Action Rule to handle the transaction step "Car returning has been declared"

Some flags were also added to the *causal link* action specification that handle how the Execution Engine should behave when running this type of actions, namely the **cancel process** and the **continue if same user** flags, that refer to whether the *causal link* action cancels the current process, for example on the passage of a transaction to the "quit" state, and whether the Execution Engine should take the user directly to the execution of the transaction step specified in the causal link, when it reaches this action, in case the current user in the engine's task is also responsible for that step. The latter flag could be applied, for example, to the first causal link depicted in Figure 16. In this scenario, due to this causal link action, if the car was deemed to be damaged, the Execution Engine would automatically execute the Action Rule related to the task *Damage Handling is Requested* as soon as it was generated, instead of continuing the execution of this Action Rule with the evaluation of the next *if statement*'s condition. This is very important in terms of usability, since the process can flow naturally between different transactions without the user needing to go back to his main Dashboard and search for the new Action Rule that needs their input. In this type of action, minimum and maximum are optional and by default and come pre-filled as 1. This cardinality indicates how many transactions should result from the current action, whereas if minimum doesn't exist, by default is equalled to 1 and if maximum doesn't exist, by default is equalled to minimum.

3.1.2 Action Rule Example Application and Explanation

An example of an Action Rule definition, adapted to the last iteration of our Action Rule's Syntax from [24], can be seen in Figure 16.

After the first *if* statement, an *informal expression* that needs to be evaluated by a human user physically inspecting the car and comparing it to the damage sheet signed at pickup can be found. In the event that there is freshly observed damage on the vehicle, a boolean property in the rental instance gets the value *true* written to it before a transaction to handle the issue is requested. This property serves as a flag in the rental entity and can then later be used to make general queries about rentals with or without damage. We then have a couple *if* actions that automatically evaluate whether penalties should be applied. In case penalties are to be applied, mathematical expressions can be specified to calculate them automatically that take into consideration properties from the current process. In determining whether there is a location penalty, one can see the usefulness of having our *dual specification of Action Rules* for each transaction state. In this case we can be sure to have the organizational facts that originated from the *Car returning **IS** declared* transaction step which are then needed in this **HAS_BEEN** Action Rule. More specifically, the *Actual drop-off branch* property would be a fact produced in the **IS** Action Rule and not in the **HAS_BEEN** rule. This need of having actions and facts in both "parts" of a transaction state was disguised with the term **WITH** in the current and limited ARS of DEMO.

3.2 Action Rules Management Component

In order to define an *Action Rules Management Component* component that allowed the visual programming of these Action Rules, the Blockly library was used. Blockly is a library that adds a visual code editor to web and mobile applications. The Blockly editor uses interlocking, graphical blocks to represent code concepts like variables, logical expressions, loops, and more. It allows users to apply programming principles without having to worry about syntax or the intimidation of a blinking cursor on the command line [30].

It thus allows, as is the goal of this component in DISME, managers, or individuals in a comparable position in organizations to develop Action Rules that are then saved and used in the Execution Engine through the Dashboard component, even if they have little or no prior programming experience. It is responsible for the creation and consequent storage of Action Rules for a transaction step, that is, a transaction type in a specific transaction state. Another important feature available on this component allows us to see all previously created Action Rules and, if needed, load them onto the visual programming editor for editing. The choice of this library was also due to the fact that it is compatible with all the main browsers, i.e. Chrome, Firefox, Safari, Opera, and Microsoft Edge and that it is highly customizable and extensible [31].

Blockly comes with numerous predefined blocks. Everything from mathematical functions to loop structures. However, in order to interact with an external application, one has to create custom blocks that implement their business logic and syntax. Table 3 ultimately represents the logic used for the implementation of DISME's custom Blockly blocks, whose current iteration can be seen in Figure 17.

The Blockly library allows the implementation of every syntax rule displayed in Table 3 through a high-customizable workspace that, for example, allows the definition of numerous field types, such as labels, dropdowns, checkboxes, and text or number inputs. It also allows multiple input types,

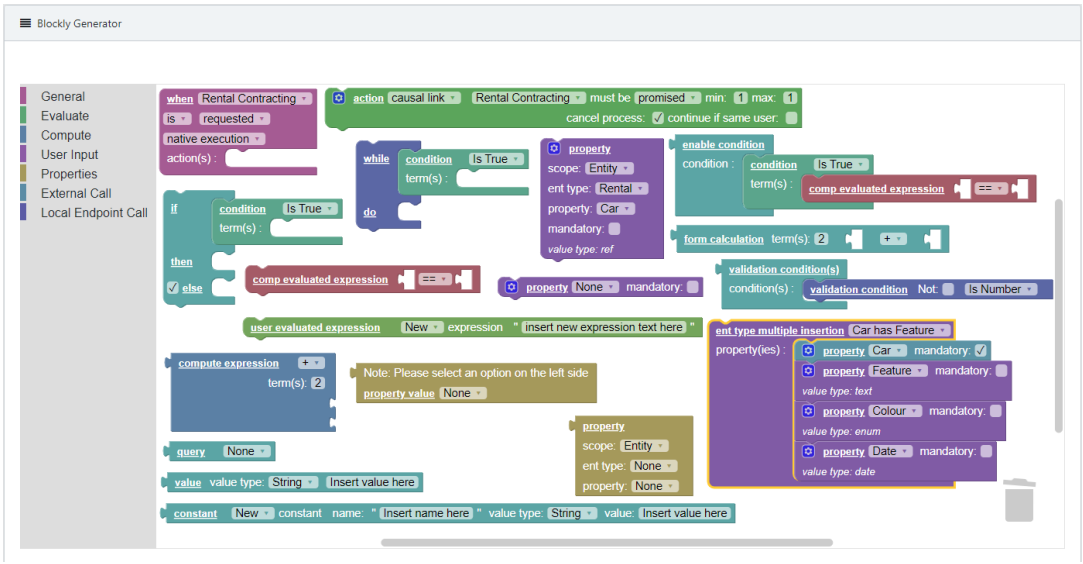


Fig. 17: Action Rules Management Component Blocks

which differ from the fields of these blocks in that, while fields will serve for the interaction of the block with the user, inputs will serve to establish interaction between blocks. These input types are value inputs, such as the ones in the *comp evaluated expression* block that allow us to insert one block inside each input presented in a puzzle-like fashion, statement inputs, like the ones in the *while* block that allow us the introduction of one or more blocks and are represented through indentations on the block and, finally, dummy inputs, for the displaying of labels, text inputs, checkboxes and such. An example of the difference between fields and inputs can be seen in the *while* block, where the fields are where the title of the block is presented, while the inputs are the indentations in it that will allow the connection to other blocks.

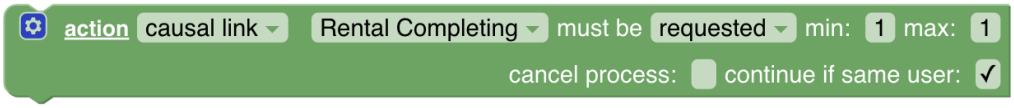
Blockly also allows syntax validation through provided methods that lets one specify which blocks interact between themselves, that is, which blocks one particular block can connect to, through its upper and bottom connection, and which ones it accepts in its inputs, while also allowing the reconfiguration of a block depending on a field's selected option, as can be seen in the *action* block presented in Figure 18, that reconfigures the block's fields depending, for example, on the chosen action type.

In the following subsections, the most relevant innovations in the Action Rules Management Component will be discussed, followed by an exploration of its storage and then a presentation of an example regarding its design.

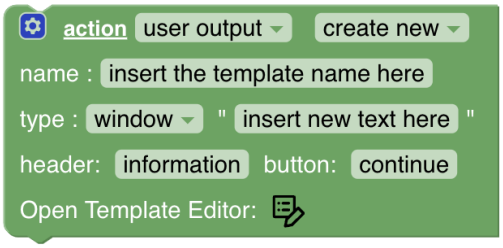
3.2.1 New Functionalities

To accommodate the most recent iteration of the Action Rule's Syntax Specification, represented in subsection 3.1.1, an update to DISME's custom blocks was necessary.

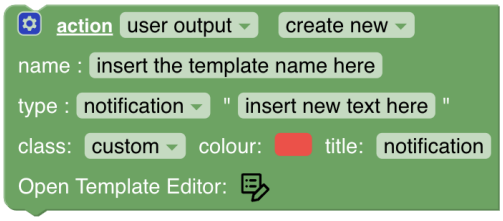
A significant upgrade was undertaken to enhance the functionality and efficiency of our Blockly installation. The existing setup, which relied on local files of an outdated version, was replaced with a more streamlined and standardized approach using an NPM package installation. This upgrade aimed to leverage the benefits offered by the NPM package ecosystem, including simplified package management, version control, and seamless integration with other modules and libraries,



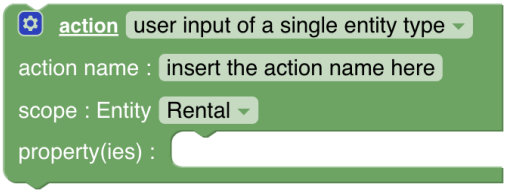
(a) Causal link



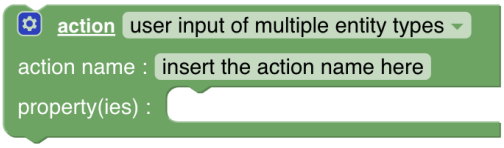
(b) User output - window



(c) User output - notification



(d) User input - single entity type



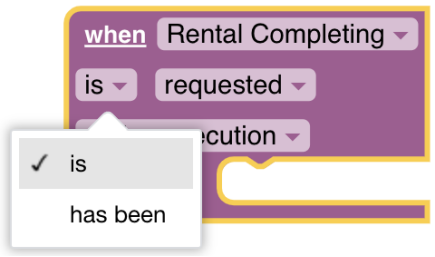
(e) User input - multiple entity types

Fig. 18: Action Block's different configurations

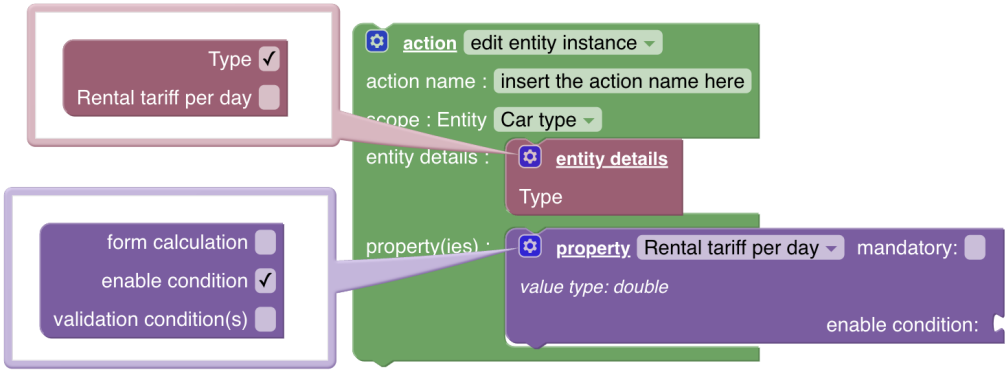
and involved careful planning, implementation, and testing to ensure a smooth migration process. By adopting the NPM package installation for Blockly, this upgrade not only improved the maintainability and scalability of our platform, but also aligned our system with industry best practices and facilitated future enhancements and updates. Also, during the upgrade process from local files to an NPM package installation for Blockly, certain code modifications were necessary to ensure compatibility with the chosen version of the package.

For the dual specification of Action Rules, a dropdown was introduced in the *When Is* block to replace the former *IS* text label that was displayed in the block. With this dropdown, depicted in Figure 19a, one can now specify whether the Action Rule being designed belongs to the dealing of the transaction step's act or fact, through the *is* and *has been* options, respectively.

The introduction of the new *Edit Entity Instance* action type directly impacts the *action* block. Some of this block's already existing configurations, depending on the action type selected, can be seen in Figure 18, but a new configuration needed to be set up. Therefore, the new action type was added to the block's action type dropdown. When selected, it mutates the block so that its necessary fields can be filled, according to Table 3. This means introducing inputs for the definition of *entity details* and *properties* or *entity types*. The first one was implemented through a new *entity details* block that opens up a dialogue menu with the selected *entity type*'s properties so that the user can select which ones are desired to be *entity details*. When the dropdown option for this action type is selected, this block inserts itself in the *entity details* input and can't be moved or deleted. The block is disposed off automatically by Blockly, however, when the action type is changed or the action block is deleted. When a property is selected in its dialogue box, it is automatically added to the *entity details* block as a text field, so that one can know which properties are selected even after closing the prompt. The *property(ies)* input lets us add *property* or *entity type* blocks. When a property block is added, it will check the selected *entity type* and present its **editable**



(a) When Is block



(b) Edit Entity Instance block

Fig. 19: Relevant innovations in DISME's custom blocks

properties only. As can be seen in the Syntax Specification and in Figure 19b, one can add inputs for *form calculation*, *enable condition* and *validation conditions* for each property by checking the respective checkbox in the *property* block's dialogue menu.

The new *causal link* flags added to the action type's specification that handle how the Execution Engine should behave when running this type of actions, were added to the action block's causal link configuration as checkboxes, as can be seen in Figure 18a. Namely, one was added for the **cancel process** and one for the **continue if same user** flags. Therefore, the specification of these flags' value is now only one click away.

3.2.2 Action Rules Storage

Custom Blockly applications need to turn blocks into code for execution. In DISME's case, as the main goal of the *Action Rules Management Component* is to store the designed Action Rules, custom blocks are transformed into structured data so that their storage is feasible. For that, Blockly code generators were configured for XML (Extensible Markup Language) generation, as serialization, or the process of converting a data structure or object state into a format that can be stored, transferred, and later reconstructed, is the primary goal of XML. The XML generated from the aforementioned component is then run through a parser designed to construct the server-side desired structured data so that its storage into the designated database tables, seen on Figure 20, can be done after submission to the project's backend.

The Action Rules Syntax is constantly updated according to the progress of the platform, and that has a direct influence on the project's database. When updating it to accommodate the most recent changes, the main principle of DISME's database, modularity is key, was followed to the letter. As such, it was deemed unnecessary to introduce a new table for the new *edit entity*

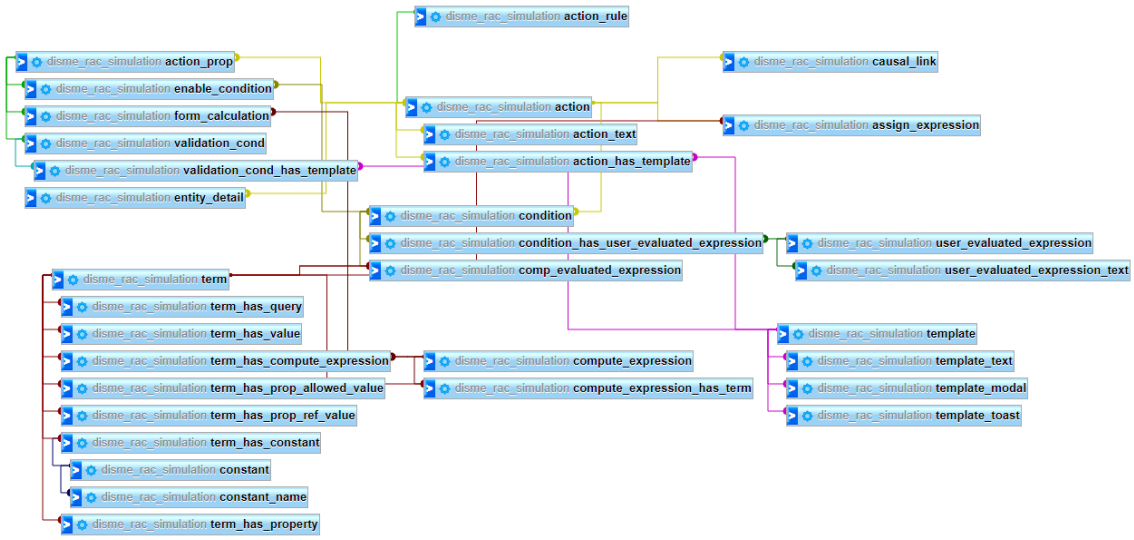


Fig. 20: Action Rules Storage Database Tables

instance action type, as it has the same principles as the *user input* action type, as can be seen in Table 3. So, the *action prop* table, which stores properties belonging to an action, is now used for both action types, as are its complementing tables, such as *enable condition*, *form calculation* and *validation condition*. The distinction between these two action types is made on the *action* table through the *type* attribute. A new table was introduced, however, for the respective *entity details*, as this concept wasn't applied before in the project.

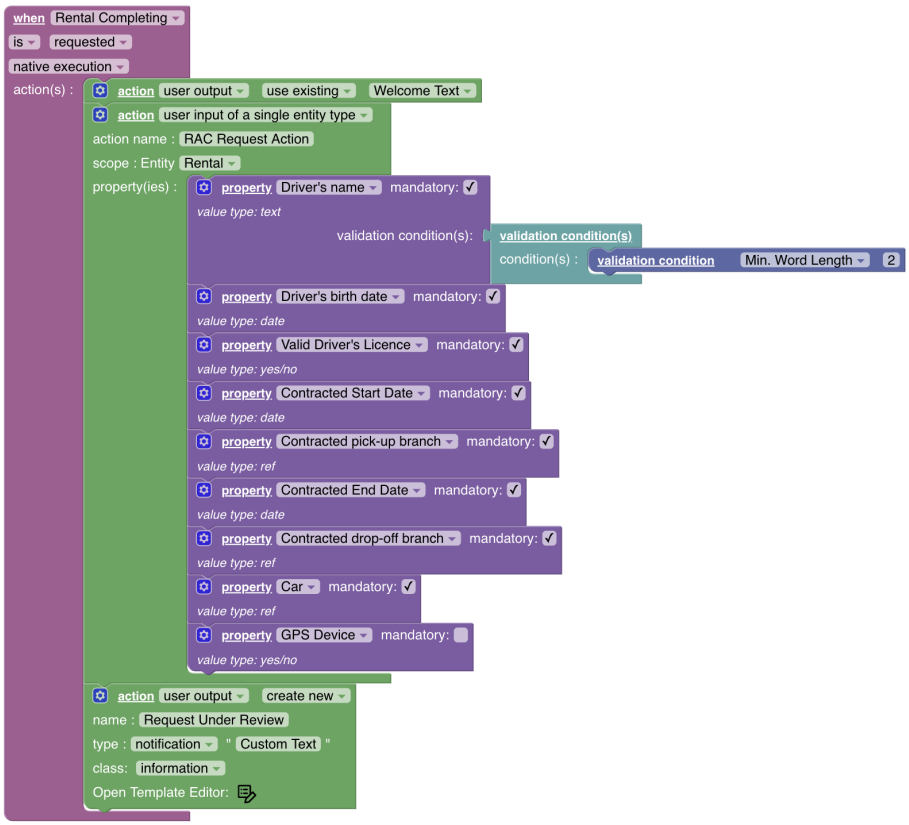
The remaining relevant innovations, as the dual specification for Action Rules and *causal link*'s new flags, were provided through the introduction of new attributes in the *Action Rule* and *causal link* tables, respectively.

3.2.3 Action Rule Design Example

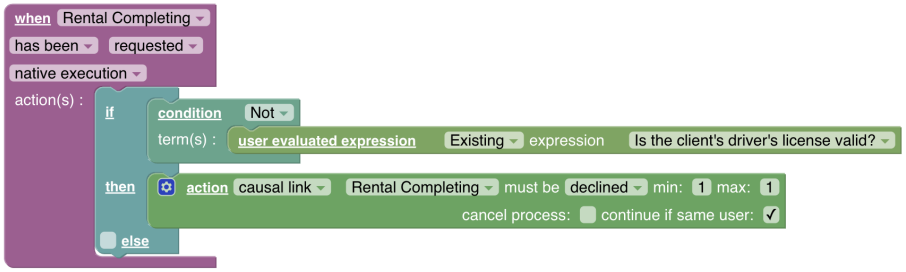
An example of the definition of an Action Rule using this component can be seen in Figure 21. This example represents the dual Action Rule specification of the first transaction type on a Rental process, that is, the *Rental Completing*.

In the Action Rule regarding the act, in Figure 21a, we have the presentation of a predefined welcoming template to the *renter*, through a *user output* action and then a *user input* action for the displaying of a form containing the main information regarding the rental and the renter. The final action in this Action Rule is again a *user output* action, where a window is used to display a message to the *renter* informing them that the filed request is now under official review. However, in this second *user output* action, a new template style is to be used, and so it is created directly through Blockly. For that, one can define the template's information and text directly on the block, or open the **Template Editor** through this block's corresponding button, as is depicted in Figure 22, and then specify the template's information and text. The *Template Editor*'s main advantage over direct specification on the block is its usage of a WYSIWYG text editor, through the integration of the open-source TinyMCE's [32] component, allowing users to create formatted content within a user-friendly interface.

Regarding the fact's Action Rule displayed in Figure 21b, the *rental completer* is displayed a *user evaluated expression* for assessing if the *renter*'s driver's licence is a valid one. In case of a



(a) Rental Completing IS requested



(b) Rental Completing HAS BEEN requested

Fig. 21: Dual Action Rule specification for the request state of Rental Completing

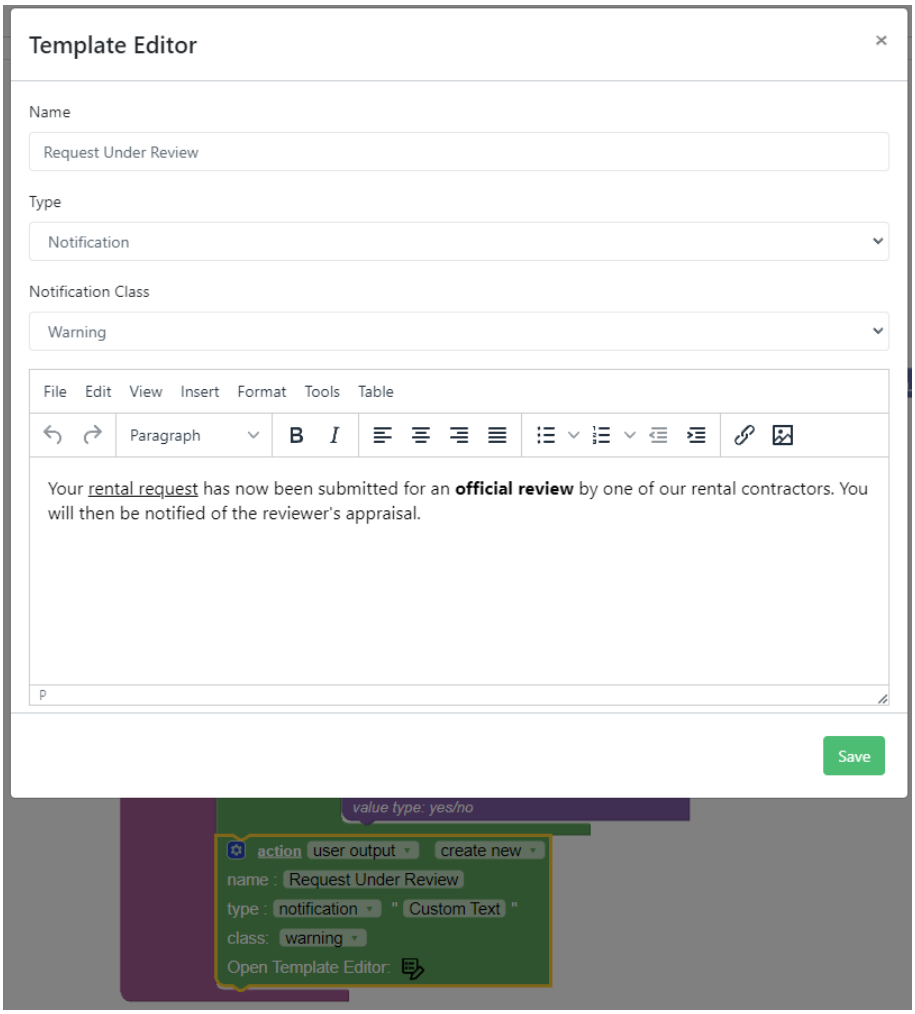


Fig. 22: DISME's Template Editor

negative appraisal, a *causal link* action is triggered to transition the ongoing *Rental Completing* transaction to the *declined* state. Otherwise, in the absence of any specified actions in the *else* part of the block, the current *Rental Completing* transaction will follow the basic transaction pattern, as can be seen in Figure 5, and advance to the *promise* transaction state.

3.3 Forms Management Component

If some actions specified in Action Rules involve user input, as is the case showcased in Figure 21a, the user must design the respective form using the Forms Management Component provided, which was previously developed by a colleague from DISME. This component allows developers to edit, render and translate highly customized forms that can be used to collect data from users and drive the overall flow of the task. An example of a form designed with this component, from the aforementioned Action Rule, can be seen in Figure 23.

Form builders provide an efficient and convenient way of creating forms without writing code. They allow users to create forms using drag-and-drop features, saving time and effort. In order to develop the Form Editor component, research was made about several plugins that enabled a form editor, that offered a wide range of features to help create dynamic and responsive forms, to be integrated into an Angular frontend, the one currently being used in DISME's architecture. The plugin ultimately chosen was Form.io [33], an open-source form builder that was completely integrated into DISME, with the customization of several functionalities to satisfy the concrete specifications and needs for its use in DISME.

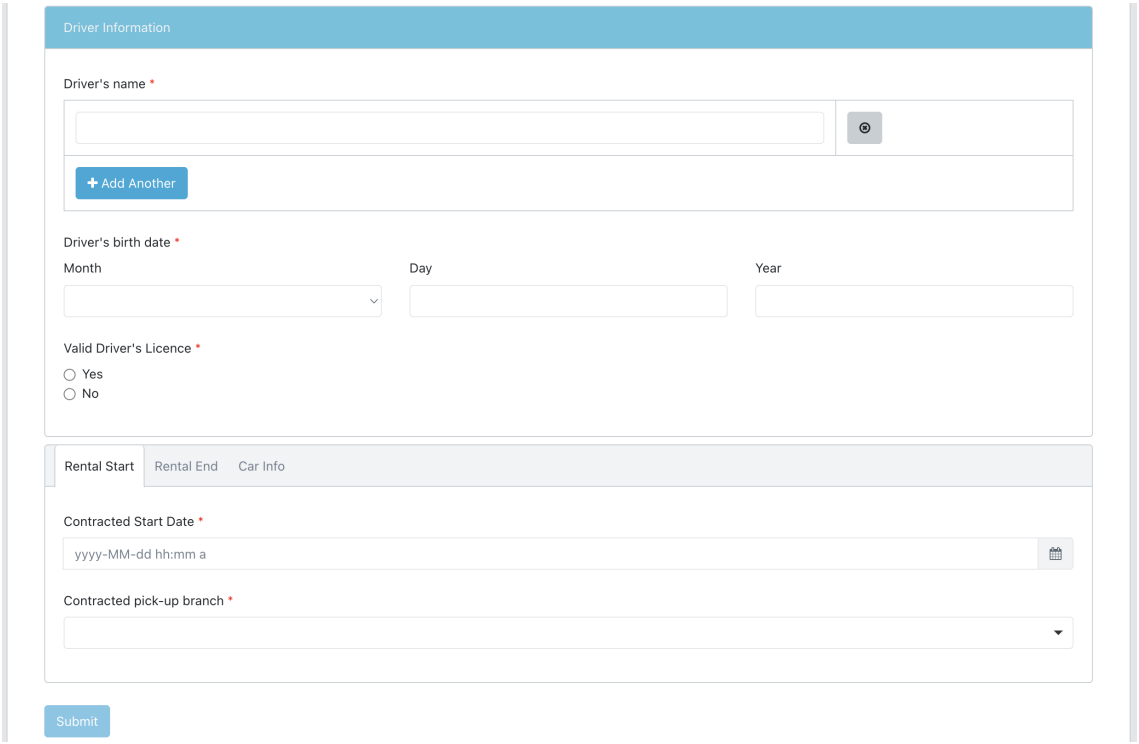
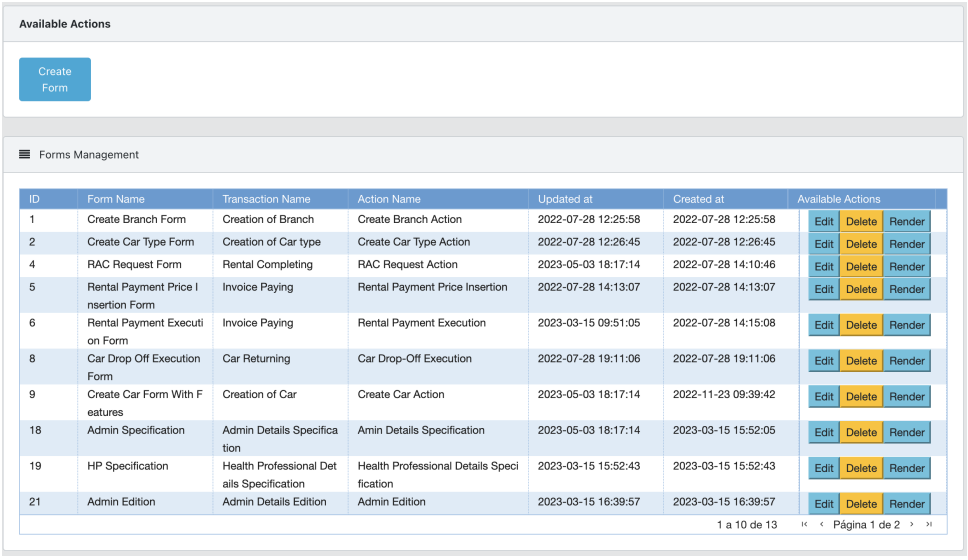


Fig. 23: Rendered form for the *user input* action of the *Rental Completing is requested* Action Rule.

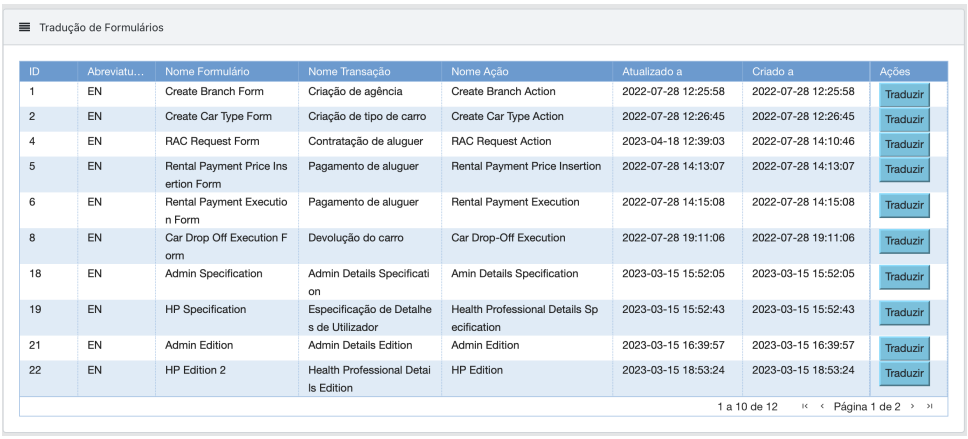
The two main components referring to DISME's forms are, then, responsible for editing, rendering and translating them within the platform. They were previously developed by another colleague, and have two different functions:

– Forms Management, which can be seen in Figure 24a, where the user can create forms from *user input* actions defined in the Action Rules Management Component, mostly defining the structure of the form to be presented to the user. It also provides facilities for editing, deleting and rendering an existing form. The form editing environment can be seen in Figure 15.

– Forms Translation, visible in Figure 24b, where users can translate forms defined in other languages in the system into their language. The translation is done automatically using resources from the system's database.



(a) Forms Management Component



(b) Forms Translator Component

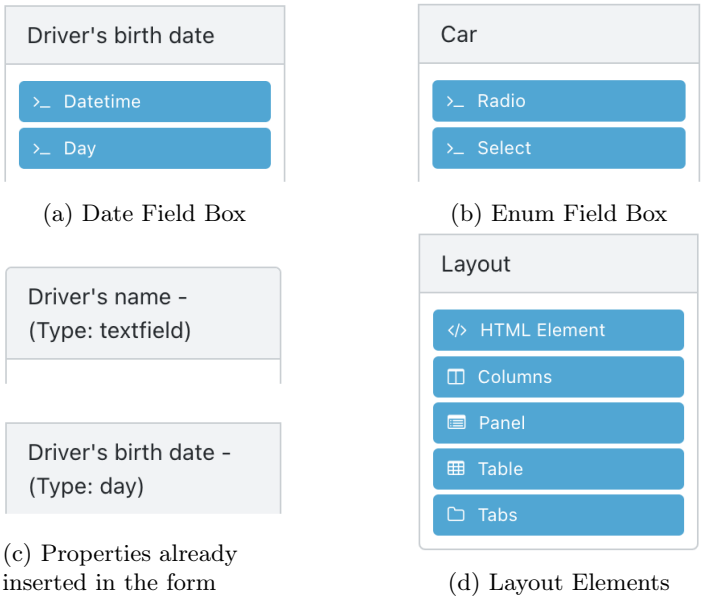
Fig. 24: DISME's Forms Components

The Forms Management Component is the bridge that connects the user to the Form Editor, through its create and edit buttons, used to define a form's layout. After having defined the properties inside a form and its characteristics, through Blockly in the Action Rules Manage-

ment Component, one has to define the layout of these properties within the form through this component, so that it can be correctly presented to the user at run-time.

When accessing the Form Editor for creating/editing a form, all properties are loaded to Form.io's interface from the database selected action. Users can easily add and modify these properties into the form, as well as arrange them in a logical and intuitive manner. For each property, an array of field types are automatically picked depending on its type, as can be seen in Figure 25. For example, if a property is of type *date*, *date* and *datetime* options are presented in the property's field box, as can be seen in Figure 25a. In case it's an *enumerate* or *property reference* property, meaning that its values reference another property's values in the system, *radio* and *select box* field types are loaded into its form editor's property box, as can be seen in Figure 25b. This is done so that the user can choose the most appropriate field type to insert into the form, depending on the respective property and its intended behaviour.

When a field type is selected and a property is inserted into the form, its field box is updated immediately to reflect the changes made. This means that the user can see exactly what they are adding to the form, and can easily edit or delete properties as needed. In addition, the inserted field type is attached to the property's name and the respective field box is emptied, as can be seen in Figure 25c, which makes it easy to identify and manage each property. This feature is particularly useful when working with large and complex forms that contain many different properties. If a property is then deleted from the form, the field box is reinstated to its previous state, with all the field types available once again. This allows the user to make changes and adjustments to the form as needed, without losing any of the work that has already been done.



(a) Date Field Box

(b) Enum Field Box

(c) Properties already inserted in the form

(d) Layout Elements

Fig. 25: Form Editor Field and Layout Boxes

In the context of form building, layout elements are essential components that allow developers to create visually appealing and easy-to-use forms. Form.io offers various layout elements that can be added to forms, such as panels, tabs, and grids, as can be seen in Figure 25d. These elements provide a higher level of design customization to users and allow them to create forms that meet

their specific needs. Panels and tabs are two of the most commonly used layout elements in Form.io. Panels provide a container for grouping related fields in a form. They allow developers to create a logical flow of information by grouping related fields together. This not only improves the visual appearance of the form, but also enhances the user's understanding of the form's content. Tabs, on the other hand, allow developers to organize fields into different sections within a form. They provide an intuitive way of presenting complex information to users by breaking down the form into smaller sections. An example of layout elements usage, more specifically panels and tabs, can be seen in Figure 15.

Given the significant impact that the proper functioning of this component has on the performance of the Execution Engine, the main concept of this project, as it is where the Forms' designing, rendering, storage and overall functioning are defined, it was essential to continually expand and enhance the component's existing functionalities to maintain its robustness and flexibility in light of DISME's new functionalities. As such, errors were identified and corrected promptly, and new innovations in DISME's forms context were implemented. The main innovations will be described in the following subsection, and the forms submissions storage restructuring will be expressed afterwards.

3.3.1 New Functionalities

To accommodate the most recent functionalities of DISME's forms, several updates were necessary.

During the system's parametrization, users can now specify whether a property should support *multiple values* definition. To enable this functionality, Form.io's internal flag for multiple selections/specifications was utilized. This internal flag is a powerful tool that enables users to specify different values for the same property in a form. The flag is automatically activated or deactivated based on the property's specifications, ensuring that Form.io's fields containing the property reflect the property's parametrization accurately. This allows a user to select/specify different values for the same property in a form. An example of this can be seen in Figure 15 through the *Driver's Name text* property, representing its usage in the Form Editor, and Figure 23, which shows the field in a rendered form. Another example of this functionality's usage is the one depicted in Figure 26 through the *Colour enumerate* property.

The addition of support for *has-many entity types*, this is, *entity types* that represent one-to-many or many-to-many relationships, is a significant improvement that allows for the creation of more complex forms with greater ease. When creating a user input action, inserting one of these entity types will automatically include all of its properties in the action. These properties are then grouped into a datagrid field type in the Form Editor, allowing the user to specify multiple instances of the same entity type. In addition, the Execution Engine automatically computes the linking property, making it unnecessary to include it in the form's rendering. An example of this functionality is the *Car Has Feature* entity type shown in Figure 26. This entity type includes the *Car* linking property, the *Feature*(name) property, and the feature's *Colour* property. Since the *Car* property is the entity being created, it is automatically assigned. The user can then specify the *Feature*(name) and the feature's *Colour* in duos, and add or delete rows inside the datagrid as needed.

Another relevant innovation is the loading of already-stored values when in presence of an *edit entity instance* action. This innovation significantly improves the user experience and reduces the chance of errors or omissions when editing existing entity instances. When the form's action is of

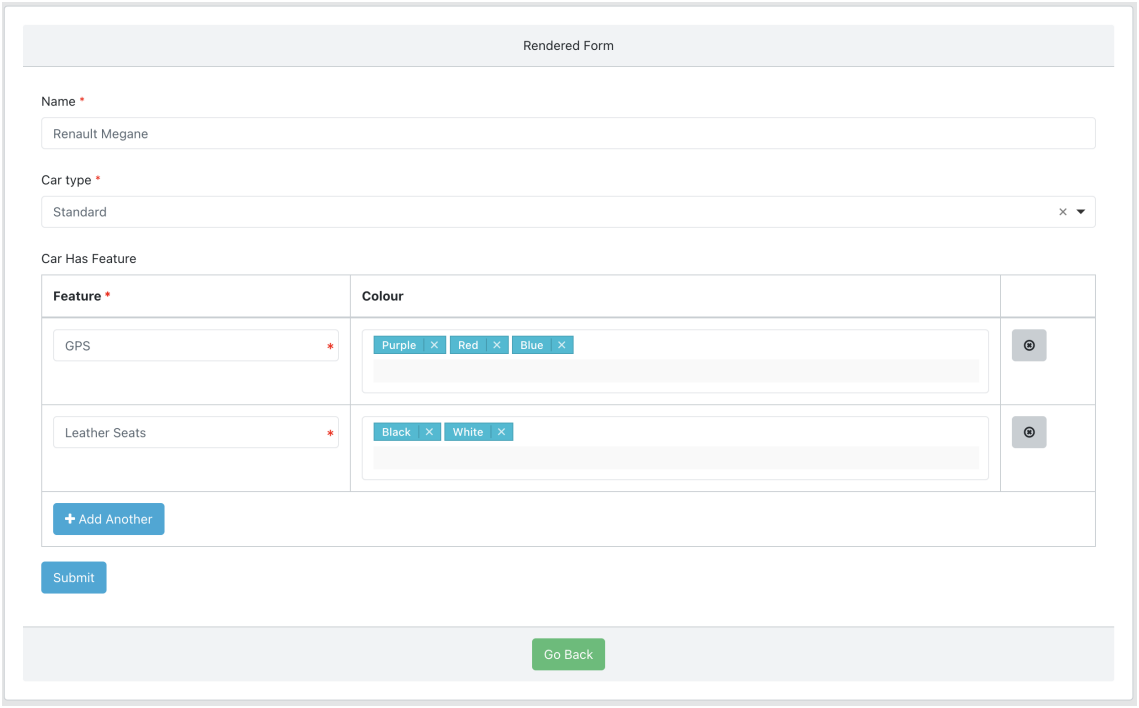


Fig. 26: Rendered form for the *user input* action of the *Car Creation is executed* Action Rule.

this type, it will fill the form's fields with the possible previous values submitted before presenting it to the user. The loading of already-stored values provides a quick and convenient way for the user to see what information has already been entered, and easily make any necessary changes. This feature is particularly useful for forms with numerous fields or complex data structures, where it can be difficult to remember all the details of a previous submission.

3.3.2 Form Submissions Storage

As software applications evolve, updates are required to ensure they remain relevant and efficient. One such update that was necessary was in the storage of form submissions and the handling of existing property flags. These flags, such as the *requires translation* flag, are vital in ensuring that the application functions correctly when used in multilingual environments. In addition to these existing flags, new flags have been introduced, including the *multiple values* flag, which allows for the storage of multiple values for a single form field, and the *soft delete* flag, which enables the deletion of data without permanently erasing it. Alongside these flags, support was added for the aforementioned *has many entity types* usage, so that the user can specify several instances of the same *entity type* in a single form filling.

When processing form submissions, it is important to ensure that the data is valid and can be stored correctly. To achieve this, a crucial step was the creation of server-side validation to ensure that all the field's specified validation conditions are fulfilled, even if Form.io's client-side validation fails. Then, it is important to verify the field type for every field in the form submission. If the field type is an *entity type* field, which refers to a *has many entity type*, a specific process is followed to ensure the data is stored correctly.

In this process, an entity is created for each row filled in the form, and the linking property for the entity type is automatically filled. Then, for each row, the filled property values are stored in the database as they would in normal property fields.

When storing property values, it is essential to take into account all the relevant aforementioned flags to ensure that the data is stored correctly and efficiently. To ensure that the data is stored correctly, the storage of property values follows a specific process. Firstly, previous values are checked for existence. If they exist, it means that we're in an *edit entity instance* action, and the *soft delete* flag must be consulted. If the flag's value is positive, previous value(s) are marked as deleted without actual erasure from the database, except if a previous value is still present in the newly submitted values, and the property's new value(s) are stored. If the flag's value is negative, the new value(s) are inserted as updates on the old values' database records, and excess previous values are hard deleted if needed.

This process ensures that the data is stored correctly and that the risk of errors or omissions is minimized. The use of the soft delete flag is particularly useful in ensuring that data is not lost accidentally, making it easier to recover it if necessary. The process also ensures that excess previous values are hard deleted if needed, reducing the amount of data stored, and improving performance while ensuring data privacy is followed.

Another important aspect of property values storage that is worth discussing is the handling of properties with the *multiple values* flag. When this flag is present, the form submission will result in several values being inserted into the same entity, meaning that the property will have all of those values in the context of the entity created. This allows for more flexibility and accuracy when dealing with properties that can have multiple values.

The storage of multiple values for a property can be particularly useful in situations where the property is related to a list of options. By allowing multiple values to be selected and stored within the same entity, the system can better represent the complexity of real-world scenarios and provide more accurate data for analysis.

Additionally, if the property requires translation, its value is stored in a table that supports multilingual definition. If it doesn't require translation, its value(s) are stored in a general usage table. This ensures that the data is stored efficiently and can be retrieved easily when needed. The use of separate tables for properties that require translation also makes it easier to manage multilingual data, reducing the risk of errors and improving the user experience.

When a property value is stored, the system checks if a dependency must be established between processes. The process of establishing a dependency between processes is straightforward. When a property of type *prop ref* is encountered, meaning that its values are derived from another property in the system, the system checks if the selected values are stored in a different process than the current property values being stored. If they are, a dependency is saved so that the system can maintain a history of dependency between processes. This allows us to know where to search if additional information is needed about the current property value being stored, as we have the dependency that saves the corresponding value's process. By considering any dependencies that may exist and maintaining a history of the dependency between the processes, the system can also ensure that its data is consistent and up-to-date, reducing the risk of errors and improving the user experience.

The final innovation in the property values storage that is worth mentioning is the automatic update of *prop ref* type properties when the referring property has a new value inserted. This is a highly useful feature that ensures that forms data is always accurate and up-to-date, improving the user experience and reducing the likelihood of errors.

For instance, when inserting a new car into the system using the form presented in Figure 26, the *Rental Completing - Car* property from the *Car Info* tab displayed in Figure 23 will be updated automatically to reflect the new option in its select box. This means that any changes made to the referring property will be immediately reflected in all the *prop ref* type properties that depend on it.

3.4 Execution Engine

The Execution Engine, the main concept of this project, operates as a controller located on the server-side. It carries out the execution of Action Rules that have been defined beforehand in the aforementioned visual programming component. The restructuring of the Execution Engine will be elaborated on in the upcoming subsection, providing an in-depth understanding of the rationale behind its changes. Subsequently, a detailed specification of the algorithm employed by the Execution Engine will be provided to facilitate comprehension. Finally, in the last subsection, the significance of logging in the context of the Execution Engine will be explored.

3.4.1 Restructuring

This component was originally developed in a previous iteration of the DISME project under the name of *Expression Engine*, with a focus on evaluating formal expressions. However, due to the system's evolving requirements and database structure, a restructuring of this component became necessary to simplify and optimize its operationalization.

Given the substantial changes needed for this component, it was decided to rename it to *Execution Engine* to better reflect its functions and to rebuild it from scratch. The previous version of this component had a confusing logic for its operationalization and lacked documentation, making the rebuilding process all the more critical.

It is worth noting that there was no code reuse from the previous iteration. Instead, the reasoning responsible for evaluating formal conditions present in the actions was leveraged, which involves the use of the *Expression Language* [34] component. This component provides an engine that can compile and evaluate expressions and logical conditions, making it a valuable asset in the rebuilding of the *Execution Engine*.

3.4.2 Algorithm

When a user intends to carry out an organizational task, which is a transaction in a specific transaction state, the Execution Engine is triggered. Its algorithm is expressed in Figure 27, and will be detailed now.

At first, it checks whether the task execution is in its starting phase, and if so, it retrieves the first action of the Action Rule associated with it. If the task execution has already started, it retrieves the action that was pending in the last execution or the action next to the last one performed. After that, it examines the type of action to be evaluated and proceeds accordingly. The procedures for each type of action are described below.

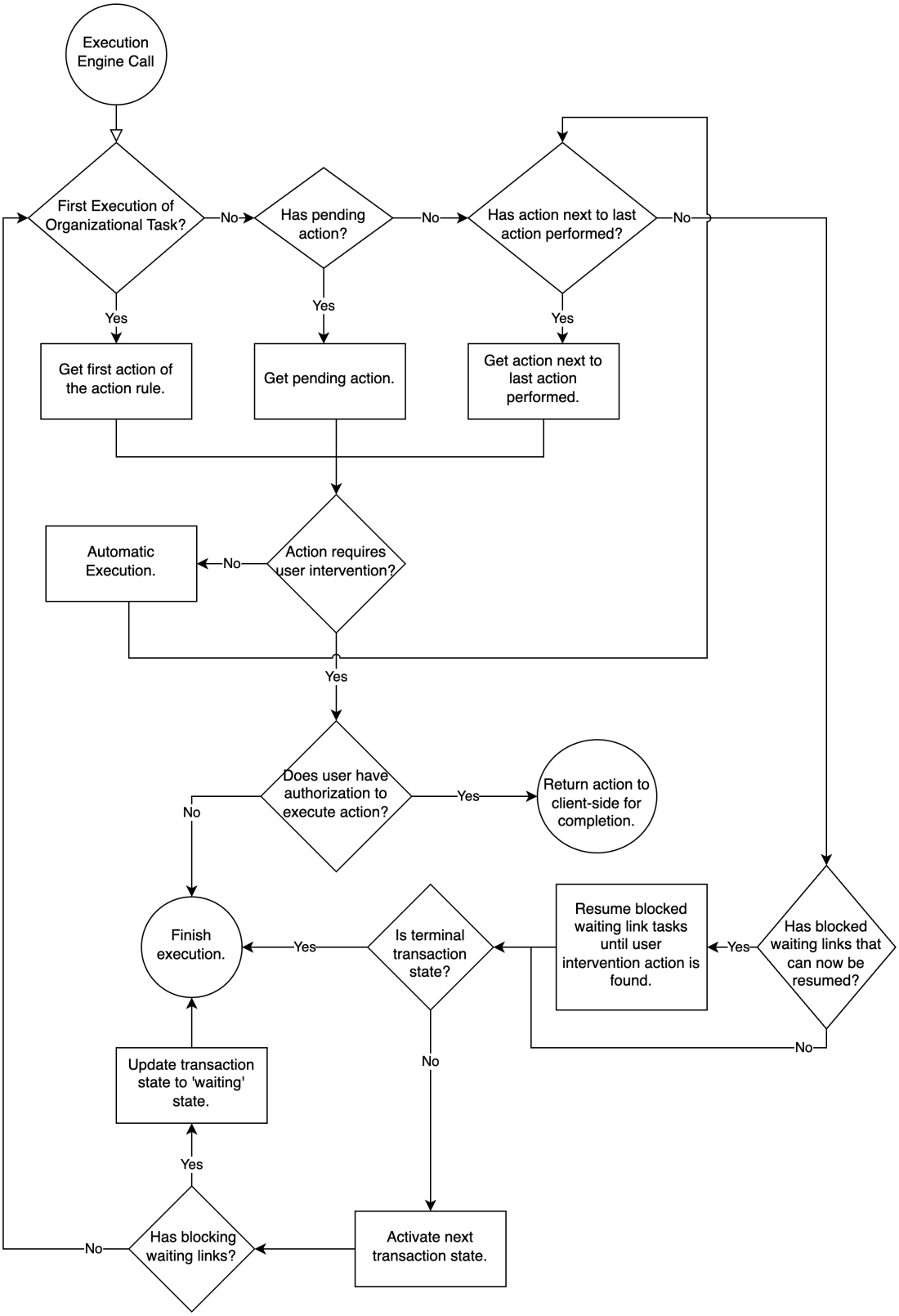


Fig. 27: Flow Diagram of the *Execution Engine*'s Algorithm

When an action of type **causal link** is encountered, the *Execution Engine* initiates a sequence of checks to determine the appropriate course of action. Firstly, it verifies whether the transaction that is being triggered exists within the ongoing process. If the transaction is present, the current transaction step is marked as performed, and the state of the transaction is updated correspondingly. In case the causal link pertains to the current transaction type, it must be the last action in the Action Rule to avoid any interference with its supplementary actions. The Action Rule management component ensures this by prohibiting any actions from following the current one if it's a causal link action referring to the same transaction type as the Action Rule's transaction type. On the contrary, if the triggered transaction doesn't exist, a fresh transaction is created and assigned the state specified in the causal link action.

Either way, after creating the corresponding action's transaction step, the Engine then conducts a search for any blocking waiting links that may be present. Waiting links are specifications about transactions that are only allowed to be executed if, and only if, the transaction step of another transaction has already been performed by the corresponding user. In these links, the transaction that *waits* for the other necessarily needs information from the previous one to proceed. Therefore, if the newly created transaction step contains any blocking waiting links, it remains in a waiting state until those links are performed, and cannot be resumed until then.

If the action has the *continue if same user* flag active, the *Execution Engine* checks if the user has the necessary permissions to execute the triggered transaction in the designated state. If the user is authorized to execute the transaction, it is forwarded to the execution of its Action Rule, leaving the execution of the original Action Rule, i.e., the Action Rule of the task from where the user called the Execution Engine, pending. If the execution of the triggered transaction eventually ends, the execution of the original Action Rule is resumed. Otherwise, the current Action Rule's execution is resumed. It is important to mention that if the causal link has the *cancel process* flag set, which is usually associated with changes to the *quit* state, the process to which the transaction belongs is cancelled. An example of the application of a causal link action can be seen in line 5 of Figure 16, where the *Damage Handling is requested* transaction step is originated if deemed necessary.

If the action being executed is of the **assign expression** type, the *Execution Engine* will evaluate the type of term to be matched to the property specified in this action. Once the term type is identified, the engine will retrieve the corresponding value from the corresponding type's system database table, and assign it to the specified property within the context of the current process. Examples of this type of action being applied in an Action Rule are shown in lines 10 and 11 of Figure 16, where the *late return policy* property is assigned the boolean value true and the *late return penalty charge* property is calculated based on the provided *compute expression*.

Compute Expressions are used when we want to compute the value of a mathematical expression that may contain constants, values, properties, queries or compute expressions themselves as its variables. Its main objective is to calculate a value based on a mathematical evaluation of these options' value at run time. During runtime, the construction of these expressions takes place, taking into account their terms' value at that specific moment. Subsequently, they are evaluated through the utilization of the Expression Language [34].

When the *Execution Engine* encounters an action of the **user input** type, it checks the action's associated form whose structure has been previously defined in the Forms Management component.

The form is then passed to the client side in order for it to be presented to the user for input. Client-side validation is then performed by the form rendering component, and server-side validation occurs when the user attempts to save the form data in the database.

Similarly, **edit entity instance** actions check for the action's associated form alongside the action's referring *entity type*. This information is subsequently relayed to the client-side, where a modal is initially presented to the user, providing them with the available instances of the referenced *entity type* in the current process for selection with their specified details. This enables the user to select which instance of the action's *entity type* in the current process they would like to modify. Once the user has made their selection, the procedures for *user input* actions are replicated, with the user being presented with an input form. Notably, this form also loads any pre-existing values, which allows the user to conveniently and quickly view the previously entered information, and make any necessary adjustments accordingly.

When an action of the **user output** type occurs, the *Execution Engine* searches for the corresponding template and displays it to the user. This type of action has two possible forms, namely a modal (window), and a toast (notification). If it is a notification, the Execution Engine automatically proceeds with the execution of the Action Rule after displaying it to the user. However, if it is a window, the task is left pending until the user presses the window's confirm button.

The *Execution Engine* also performs condition evaluation in the context of **if** actions, which may contain logical conditions evaluated automatically by the *Execution Engine*, informal expressions evaluated by the user, or both. Figure 16 presents several cases of conditions in the context of an *if* action, where the conditions are enclosed within the respective square brackets. In the first *if* condition, there is an informal expression to be evaluated by the user, while the remaining ones contain logical conditions to be automatically evaluated by the Execution Engine. Initially, the *Execution Engine* checks if the condition includes informal expressions that are yet to be evaluated by the user. If so, it passes them to the client-side so that it is presented to the user for evaluation. After the user evaluation, the action's logical evaluation resumes. Otherwise, it proceeds to evaluate the condition specified in the action. After there are no more informal expressions to be evaluated and the expression is constructed, the condition is evaluated through the usage of the Expression Language [34], and the result determines the subsequent flow in the Action Rule, either following the associated **then** or **else** block, if present.

Now that we've gone over the processing involved with each type of action, we can classify them into two more types that aid in comprehending the Execution Engine's workflow. These types are enumerated below.

- **Automatic Actions**: These actions are executed automatically by the *Execution Engine* and include the *assign expression*, *causal link*, and *if - logical condition evaluation* types.
- **User-Intervention Actions**: These actions require user intervention and are returned to the client-side for this purpose. They include the *user input*, *edit entity instance*, *user output*, and *if - evaluation of informal expressions* types.

While executing an Action Rule, the *Execution Engine* will automatically execute all of its associated actions until it encounters an action that requires user intervention. Upon encountering such an action, the *Execution Engine* will send it back to the client-side for the user to complete. If the user completes the action, the automatic execution resumes until the *Execution Engine* finds another user intervention action or completes the corresponding Action Rule.

It should be emphasized that the *Execution Engine* always searches for the subsequent action in the flow of the Action Rule whenever an action is executed. Additionally, if the user interrupts the execution of the current Action Rule, its state is saved, and the execution resumes from the last pending action when it is resumed.

As the last step of this Execution Engine, whenever it reaches the last action of an Action Rule, it searches for and releases any blocked waiting links, before passing the transaction to the next state, following the basic transaction pattern represented in Figure 5.

Waiting links, as explained before, are specifications about transactions that are only allowed to be executed if, and only if, the transaction step of another transaction has already been performed by the user. Transaction steps containing blocking waiting links will remain in a *waiting* state until their waiting links have been executed. Once a transaction step is executed, it will search for any *waiting* transaction steps that are dependent on its execution for the continuation of their execution. If blocked transaction steps are discovered and no additional blocking waiting links exist, their state will be changed to *pending*, and execution will resume until an action requiring user intervention is encountered. However, the engine will not return the task to the user; rather, the task is simply added to the *Pending Tasks* section of the responsible users' Dashboard.

After creating the new transaction step for the transaction's next state, once more, as happens with causal link actions, the Engine will proceed to search for any blocking waiting links that may be present. If the newly created transaction step contains any blocking waiting links, it will remain in a *waiting* state until those tasks are executed. During this time, the engine will refrain from automatic execution. Otherwise, though, once it passes to the next state, it will execute the state's Action Rule automatically until it finds an action that requires the intervention of the users with the role responsible for that state. If the Action Rule is entirely composed of automatic actions, it will complete it and move on to the next state. This procedure continues until it finds a state that contains a user intervention action or until it reaches the completion of a terminal state of the transaction.

In addition, if the Execution Engine encounters an action that requires user intervention, it will proceed to verify whether the user who initiated the call to the engine is authorized to execute that particular action. The reason for this verification process is that different roles execute different transaction states within a transaction, as demonstrated by the complete transaction pattern of the DEMO Theory shown in Figure 8. Specifically, the initiator role participates in different states compared to the executer role, highlighting the need for proper authorization checks during the execution process. Should the user be authorized, the action will be returned to the client-side for further processing. Otherwise, if the user is not authorized to execute such an organizational task, the Execution Engine will come to a halt and cease its execution.

The reason for this behaviour of automatically executing states after completing the current state is to ensure that there are no entries in the list of pending tasks in the user's Dashboard, as shown in Figure 29, that do not require intervention from the user. This helps to eliminate unnecessary clicks that would produce no action for intervention. Note that the task that has just been performed always disappears from the user's Dashboard.

3.4.3 Logging

Logging is an essential aspect of any software system, and it plays a crucial role in the system's development, debugging, and maintenance. In the context of the Execution Engine, logging is particularly important as it provides insights into the behaviour of the engine and the tasks it performs. The logging information generated by the engine can be used to analyse and optimize the system's performance, diagnose and fix errors, and monitor its health and stability. In this subsection, we will explore the significance of logging in the context of the Execution Engine and the importance for its efficient and reliable operation. We will discuss the various types of logs that can be generated by the engine and the information they contain.

The Execution Engine's primary function is to execute actions according to predefined Action Rules. As such, it is imperative to maintain a detailed record of the status of each action during its execution, so we know what actions have been executed and whether there are actions whose execution is pending inside an Action Rule. To accomplish this, the Execution Engine creates a history of every action's execution beginning and ending. This history includes logging information about the action's current status, such as whether it is in progress or has finished, as well as a timestamp indicating when the action began or ended. Additionally, the engine records the identity of the user who initiated the action to aid in tracking, monitoring and auditing. Furthermore, to enable traceability, the engine captures the organizational task in which the action is/was executed.

Assign expression actions are a type of action that execute automatically in the Execution Engine. As a result, it is essential to maintain a detailed record of the actions executed by the engine to ensure accurate and efficient operation. To achieve this, the engine logs essential information related to each *assign expression* action as it is executed. This includes details such as the specific *assign expression* that was executed, the organizational task that triggered the action, the value record that was created or modified in the system's database, the user responsible for executing the task, and the action's execution timetable. All of this information is saved in the system's database to provide a comprehensive and reliable history of the system's operation. By capturing this information, developers and administrators can perform various functions such as monitoring and troubleshooting. It also allows for better traceability and enables efficient system management, thus ensuring that it operates *assign expression* actions correctly and can effectively serve its intended purpose.

The Execution Engine utilizes the *Expression Language* to automatically calculate *compute expressions*. In doing so, the engine generates a detailed log of each *compute expression*'s evaluation, including information about the expression analysed, the resulting expression constructed by the engine, and the *Expression Language*'s calculation result. Furthermore, the engine captures essential metadata about the execution, such as the identity of the user who initiated the action, the execution's timestamp, and the organizational task in which the *compute expression* was executed. Maintaining a detailed record of the system's operation on *computed expressions* enables traceability and facilitates issue diagnosis and performance optimization for developers and administrators. If a value is questionable at any moment during transaction processing, it's possible to revisit what originated it and to correct the system's behaviour by adjusting the expression to be calculated.

In DISME, *if then else* type actions play an important role in enabling the controlling of the execution flow of Action Rules based on conditional statements. To ensure traceability, the Execution Engine logs information about conditions and their sub-expressions, whether they are evaluated automatically by the engine itself or by a user. For conditions, the logged information

links the condition itself with the organizational task and user that triggered its execution, the complete expression constructed by the engine, the expression's result and its timestamp. For sub-conditions, information about the parent condition's log is also saved. When dealing with *computer evaluated expressions*, the Execution Engine links the parent condition's log with the evaluated expression and its result obtained using the *Expression Language*, alongside the responsible user and execution timestamp. For *user evaluated expressions*, the Execution Engine links the parent condition's log with the *user evaluated expression* evaluated, the user's evaluation (true or false), the user responsible for the evaluation and its timestamp.

The Execution Engine-powered Dashboard is also equipped with a logging system that aims to store information related to the acknowledgement and opening of tasks by users. This logging system serves two purposes. Firstly, it facilitates the display of the *new* indicator in the Pending Tasks tables of the Dashboard, as can be seen in Figure 29, which indicates tasks that are yet to be acknowledged by the user. Secondly, it records the time when the user first opened the execution of a respective task, thereby enabling data analysis of the time taken by users to execute different tasks. By detecting patterns that suggest too much time between the first opening and eventual execution of organizational tasks, this information can be used to adjust the handling of tasks.

3.5 Dashboard

As stated previously, the Dashboard is an essential component that allows users to interact with organizational tasks through an intuitive interface. It is worth noting that this particular component had been developed earlier by one of our colleagues. For the purpose of illustration, the interface of this component can be observed in Figures 28, 29, and 30.

Within the Dashboard, users have the capability to initiate processes and track the count of completed, pending, and delegated tasks, which are illustrated in Figure 28. Additionally, users can perform organizational tasks, as displayed in Figure 29, and launch tasks of existing processes, as observed in Figure 30. Notably, each task is colour-coded to represent the type of process to which it belongs, as determined during the creation of the process type in the corresponding component, as can be seen in Figure 41b.

The focus on this component was integrating the Execution Engine functionality into the pending tasks section, as depicted in Figure 29, but the effort was also put into redefining server-side functions to enhance efficiency, updating the remaining buttons' functionalities in the Dashboard, the insertion of pagination in its first and last sections and the correction of each section's filter behaviour. As a means to achieve these goals, we first conducted a refactoring of the server-side functions that were responsible for retrieving the required data for display across the different sections.

3.5.1 Improving Accessibility and Functionality of Process Initiation in the Dashboard Component

Within the **Start a Process** section of Figure 28, the transaction types that start processes, such as the *Rental Completing* transaction type of the *Rental* process type from Table 1, should be visible to users as long as they have access to it with the role of initiator, thus allowing them to start a process. However, the component was previously limited to only displaying the process types that the user would have started in the past, if he had already started any process, in an

attempt to only display the most started processes. This resulted in an omission of processes that had never been started before, but to which the user had access and might need access to perform that operation.

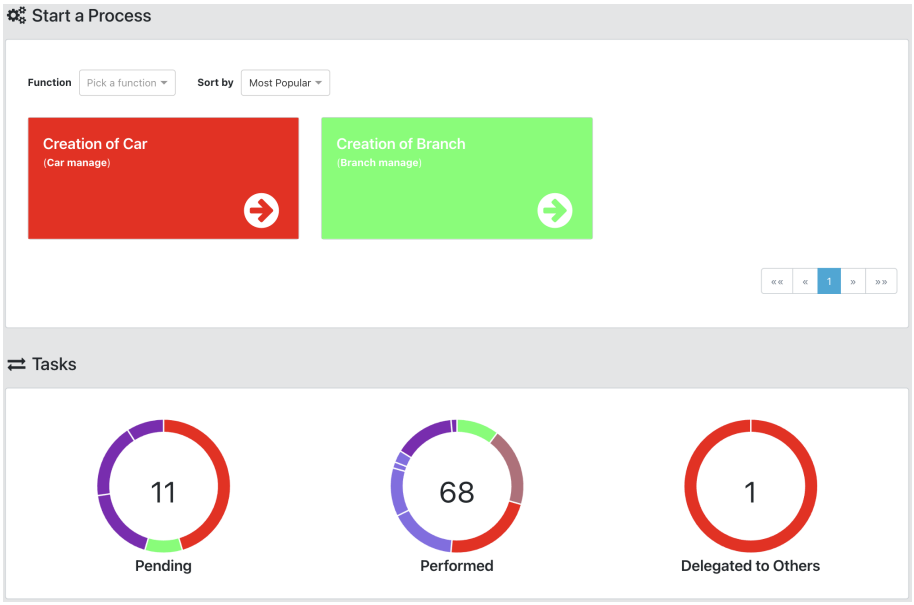


Fig. 28: Dashboard Interface - Start Process and Task Counting

To address this issue, the component's functionality was modified to show all the process types the user can initiate, sorted in descending order of the number of processes created. This alteration retains the principle of displaying the most frequently initiated processes, while also ensuring that all accessible process types are visible to the user. Additionally, a new feature was added that creates a process and transaction of the selected transaction type in the first state of the transactional pattern *request*, as seen in Figure 8, when the user clicks on the corresponding button, and executes it automatically through a call to the Execution Engine. The new task is also added to the list of pending tasks, displayed in Figure 29, and marked as a new task. These changes were implemented to improve the usability and accessibility of the Dashboard component and are important contributions to this project.

3.5.2 Efficient Retrieval and Execution of the Execution Engine-Integrated Pending Tasks Section in the Dashboard

The section dedicated to **Pending Tasks**, whether delegated or not, as depicted in Figure 29, was the most complex and time-consuming part of this component. Within this section, users are able to view the details of their organizational tasks awaiting completion, such as their creation date, the associated process, the transaction type and its current state, and to execute them.

Initially, the functionality of the Execution Engine was integrated into the relevant buttons. Then, handling for each of the *user intervention actions* that the Execution Engine can return to the Dashboard was dealt with, so that users can complete their actions and resume the Execution Engine's execution of the current task. The Dashboard's handling of these actions will be further explained in Section 3.5.4. Additionally, the server-side functions responsible for retrieving the

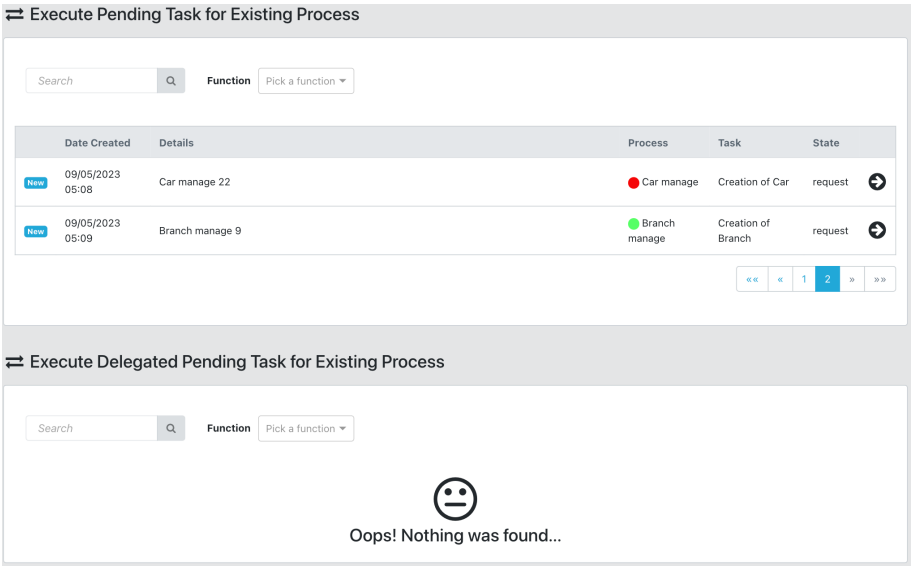


Fig. 29: Dashboard Interface - Pending Tasks

pending tasks were redefined to improve the selection and validation of the tasks displayed to each user. This ensures that only the tasks to which the user has access, based on their organizational roles and system restrictions, are displayed. An instance of this could be a task that is restricted to specific users in terms of their initiator/executer role or a task that can only be executed after another transaction has reached a specific state, known as *waiting links*.

For instance, in the process type *Rental*, the transaction type *Rental Completing* is only visible to the *Renter* who initiates the process and to the *rental completer* who first responds to its request. This restriction is due to applicable rules that limit the visibility of certain transactions based on the user's role and a flag indicating whether they are the only authorized user from their role to execute tasks within the process.

3.5.3 Refined Functionality for the Initiate Task of Existing Process Section in the Dashboard

In the final section of the Dashboard, the **Initiate Task of Existing Process** section, shown in Figure 30, users should be able to see transaction types that don't start processes. For instance, all transaction types belonging to the process type *Rental*, except for *Rental Completing*, as listed in Table 1, should be visible to users with their initiator role, allowing them to initiate the corresponding transaction in the desired process.

This enables users to initiate tasks that belong to an existing process. To achieve this, the functions that provide data to users have been redefined to align with the same principles used in other server-side functions of this component. Additional functionalities include the appearance of a modal for process selection, as can be seen in Figure 31, and the creation of a transaction and organizational task in the first state of the transactional pattern *request*, as seen in Figure 8, when required by the user by selecting by the desired process in the process selection modal, and the corresponding call to the Execution Engine to immediately execute the task.

Furthermore, to address the issue of limited visibility of transaction types in the first and last sections of the Dashboard, pagination was introduced to allow users to view more transaction types.

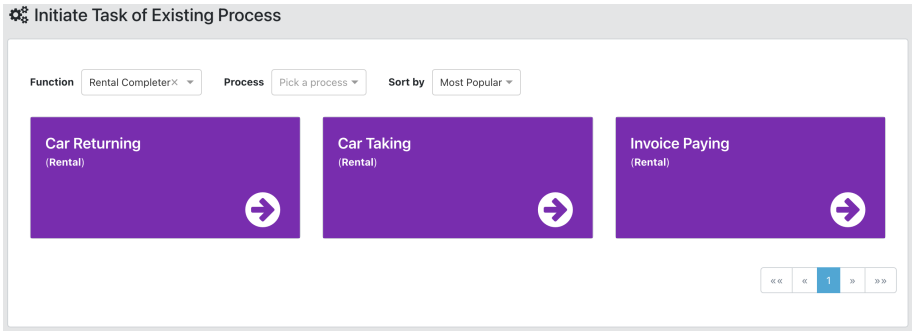


Fig. 30: Dashboard Interface - Initiate Task of Existing Process

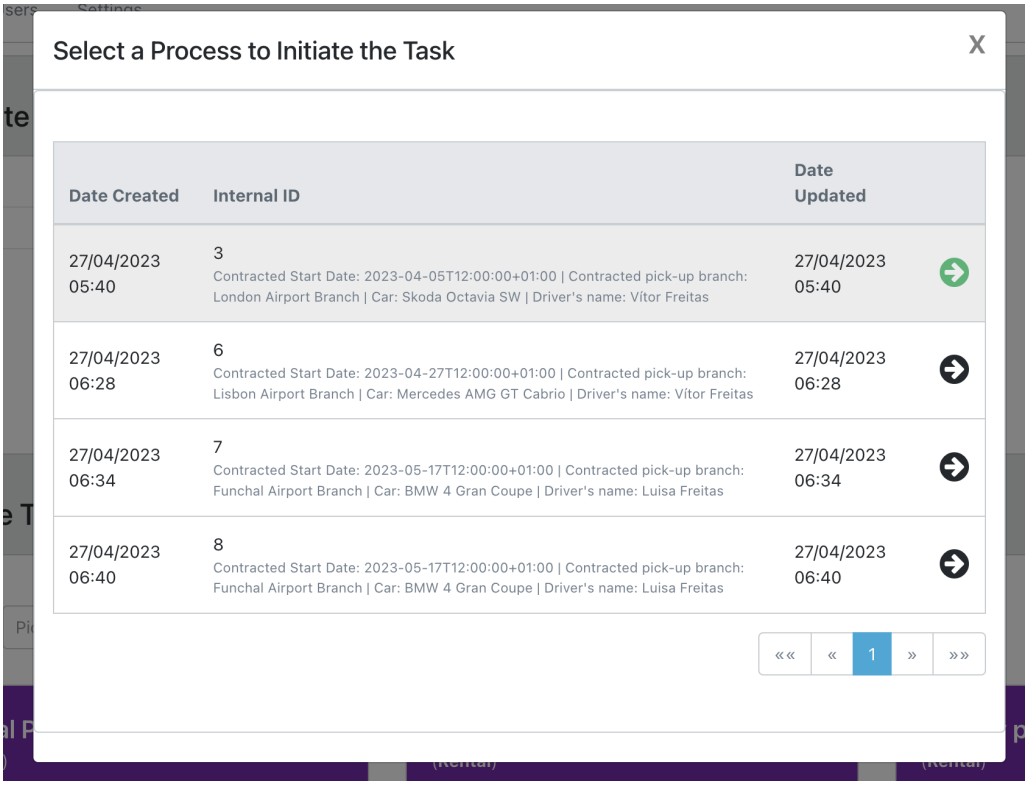


Fig. 31: Modal for process selection when initiating task of existing process

Prior to this change, these sections only displayed a maximum of six transaction types, which could result in the omission of other relevant transaction types that should appear in these sections. By implementing pagination, users can easily navigate through multiple pages of transaction types and access those that were previously hidden from view. This enhancement ensures that users have complete visibility and access to all transaction types, enabling them to initiate tasks and perform their organisational responsibilities more effectively.

3.5.4 Executing Tasks with Efficiency: An Application of the Dashboard's Execution Engine

After understanding all the Dashboard's sections' functioning, within this subsection, we shall undertake a thorough examination of the complete execution of Action Rules via the Execution Engine-powered Dashboard. For this example, our focus will be on the Action Rules that pertain to the *Rental Completing is requested* and *Rental Completing has been requested* tasks, which are illustrated in Figure 21. This analysis is essential for gaining a more comprehensive understanding of the functioning of the Execution Engine, which is the main aspect of this project, and its integration within the Dashboard that ultimately allows users to interact with organizational tasks through an intuitive interface in DISME.

The following demonstration encompasses images that are representative of the respective Dashboards belonging to two distinct users. For the purpose of clarity, User A will be taking on the *Renter* role, while User B shall embody the *rental completer* role within the context of the RAC case. It is noteworthy that these are the two pivotal roles, as depicted in Figure 12a, in the Rental Completing transaction kind, with User A serving as the initiator and User B assuming the role of the executor.

Such a transaction involves an intricate interplay of roles and responsibilities, which is crucial to ensure the smooth and efficient operation of the rental process. The *Renter* is tasked with providing all necessary information and documentation related to the rental completing, while the *rental completer* must undertake the necessary steps to process the rental transaction.

The commencement of the rental process is initiated by User A, who can access the dedicated *Start a Process* Dashboard section, as illustrated in Figure 32. Within this section, User A is presented with the option to initiate a Rental Completing process.

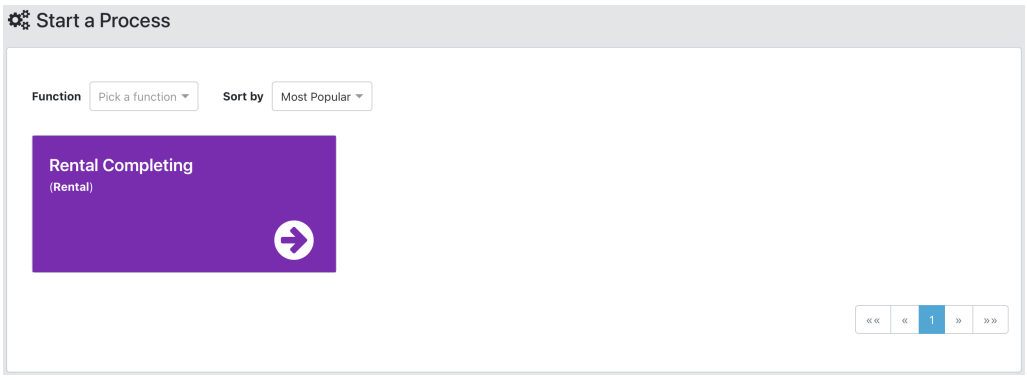


Fig. 32: Dashboard Demonstration - Start *Rental Completing* Process

Upon selecting this option, a *Rental Completing* transaction is created and enters the *request* state, originating the **Rental Completing is requested** task. This task is subsequently run through the Execution Engine and also added to User A's list of *Pending Tasks*, as demonstrated in Figure 33.

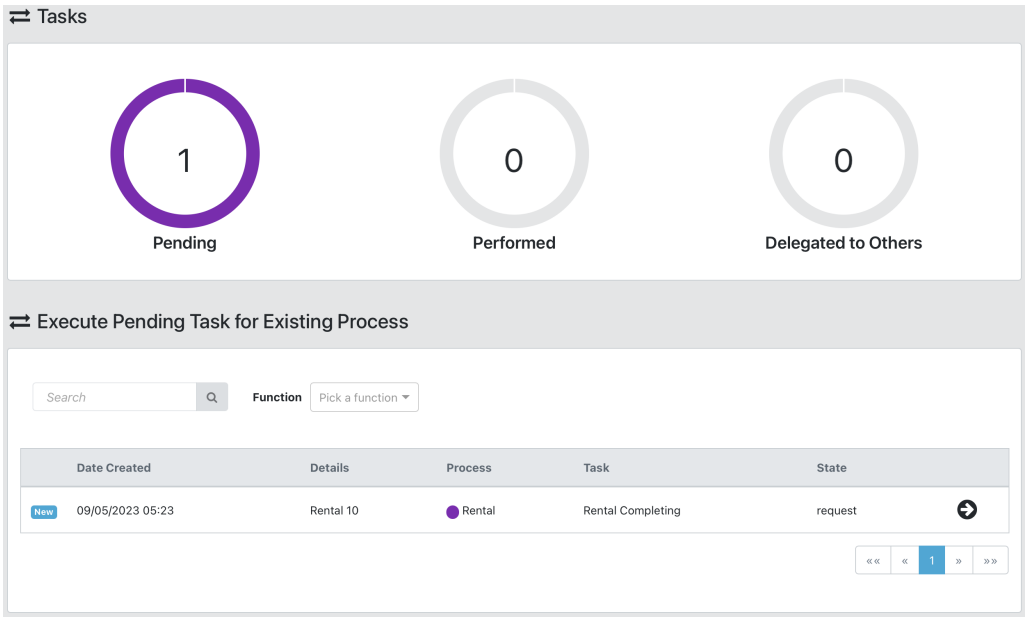


Fig. 33: Dashboard Demonstration - Task for *Rental Completing is Requested*

Upon the task's execution initiation through the direct execution at the time of its creation or through its respective button on User A's Dashboard, a call to the Execution Engine is made to execute the corresponding Action Rule, depicted in Figure 21a.

Upon receiving the Rental Completing transaction, the Execution Engine recognizes that it is the first time this task has been executed. Consequently, the engine retrieves the initial action from its Action Rule, which in this case requires user intervention and is classified as a *user output* action, depicting a welcoming message. As User A is the initiator of this organizational task and possesses the necessary authorization to execute the action, it can be returned to the client-side for execution.

The Execution Engine's response is then relayed to the Dashboard, which displays the action's template to the user in a modal form, as per its definition within the Action Rules Design component, as can be seen in Figure 34.

Upon User A's execution of the action via the modal's confirmation button, the Execution Engine is once again invoked to resume the task execution. As this is no longer the first execution of the corresponding Action Rule, the engine searches for the next action in sequence after the last one performed. In this case, the next action is also a user intervention action, with it being a *user input* action.

As the transaction state remains unchanged, and User A is still the initiator of this organizational task with the requisite authorization, the action is returned to the client-side for execution,

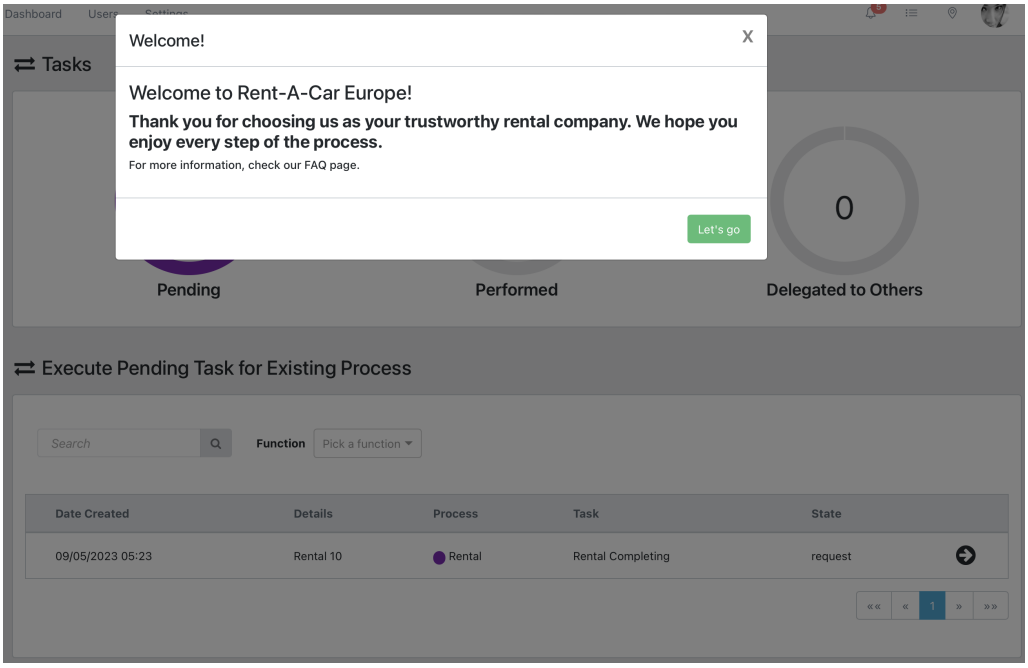


Fig. 34: Dashboard Demonstration - Execution of the *Rental Completing is Requested* Action Rule: User Output Action - Modal Type

where the Form Renderer is utilized to display the action's form in a modal-like fashion over the Dashboard, as is shown in Figure 35.

To execute this action, User A must complete the form that is presented. Once all required fields are filled in and all validation conditions are met, the *Submit* button becomes active, enabling the user to execute the action. The form data is then saved in the system's database, and another call is made to the Execution Engine to continue with the task's execution.

Once again, the engine searches for the next action in the sequence after the last one performed, and it finds another user intervention action. In this case, it is a *user output* action with a toast, or notification, type as defined in the action's template, as opposed to the modal definition used for the first action, informing the user that its request is now under review.

As the transaction state remains unchanged, and User A is still the initiator of the organizational task with the necessary authorization, the action is returned to the client-side for execution. The Dashboard then displays the action's template to the user in the form of a notification, as defined in the Action Rules Design component, as demonstrated in Figure 36.

After the notification is displayed to the user, completing its execution, the task execution resumes with another call to the Execution Engine. As this was the last action in the Action Rule, shown in Figure 21a, there are no more actions next to the last one performed, and since the task's current state *request* isn't a terminal transaction state, the Execution Engine proceeds the transaction to its next step, which is the fact of the *request* state, resulting in the creation of a new task called **Rental Completing has been requested**, whose Action Rule is displayed in Figure 21b. It then gets the respective Action Rule's first action, analyses it and realizes that is an action that requires user intervention, specifically an evaluation of an informal expression within an *if* condition evaluation.

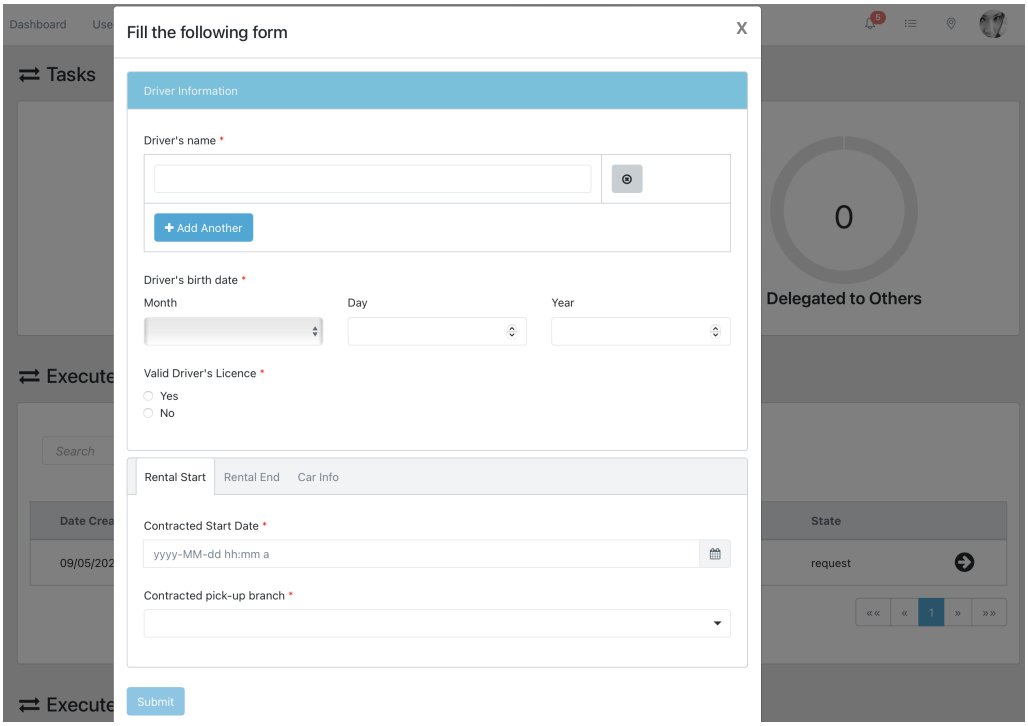


Fig. 35: Dashboard Demonstration - Execution of the *Rental Completing is Requested* Action Rule: User Input Action

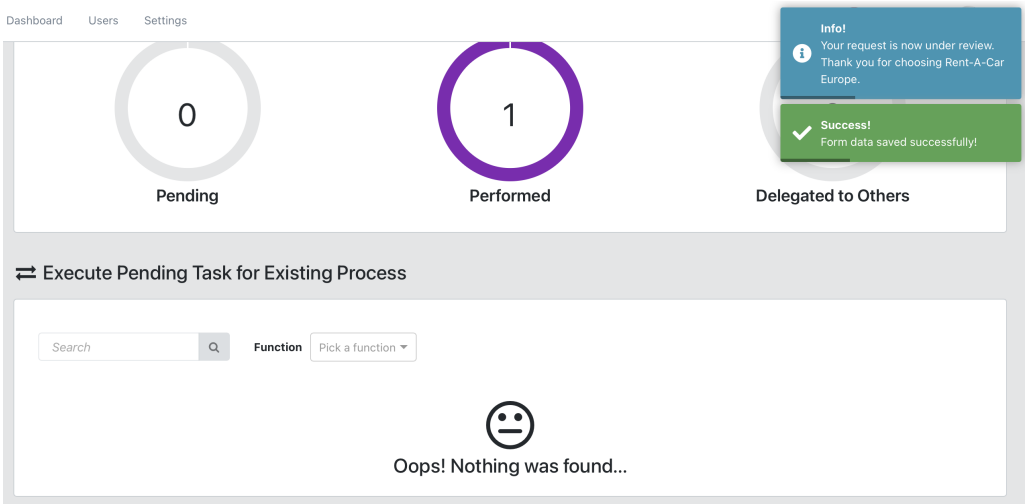


Fig. 36: Dashboard Demonstration - Execution of the *Rental Completing is Requested* Action Rule: User Output Action - Toast Type

However, as the transaction state has now changed, User A isn't the executor of the task and does not have the necessary authorization to execute this action. Therefore, the action is not returned to the client-side for execution, and the current Execution Engine call ends. Once the Execution Engine's empty response is received, User A's Dashboard is updated, with the counters for pending and performed tasks being adjusted, and the task no longer being displayed in the *Pending Tasks* section, as depicted in Figure 36.

Nonetheless, User B has the organizational role responsible for the execution of the newly created task, and thus it will appear on their Dashboard's *Pending Tasks* section. User B can proceed to execute the task as illustrated in Figure 37.

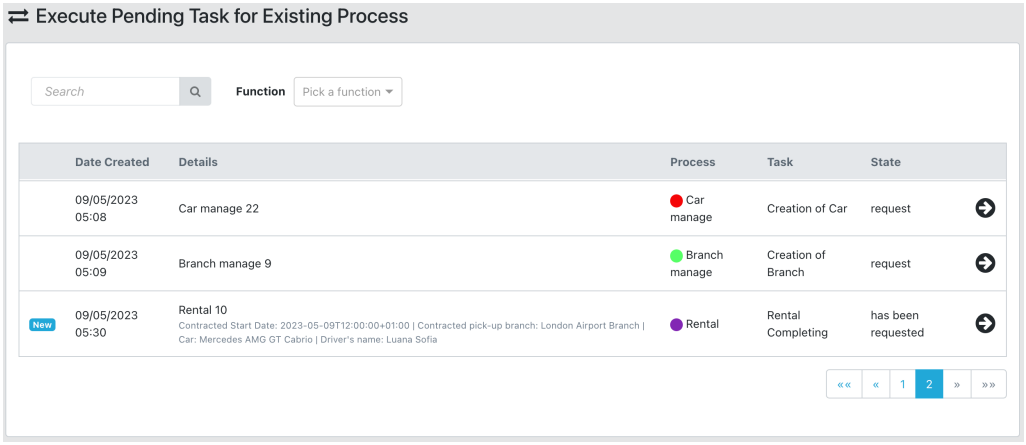


Fig. 37: Dashboard Demonstration - Task for *Rental Completing has been Requested*

When entering the execution of the **Rental Completing has been requested** organizational task through a call to the Execution Engine, it will recognize that this isn't the first time that this task is being executed, as it has already begun execution when the transaction's state was updated. As such, it will retrieve the action that was left pending, and as it is a user intervention action, being an evaluation of an informal expression belonging to an *if* condition, and User B is the executor of this organizational task with the requisite authorization to execute it, the action is returned to the client-side for execution.

Subsequently, the Dashboard presents the informal expression's text to User B in a modal format, enabling them to evaluate it using the *Yes* or *No* buttons, as can be seen in Figure 38.

Once the appropriate evaluation is chosen, the Execution Engine is called again to continue with the task's execution. The engine searches for the next action in the sequence after the last one performed and finds an automatic execution action. Since there are no more informal expressions to be evaluated by User B, the Execution Engine can construct and evaluate the complete **if** condition. As User B indicated that the client's driver's license is valid, and the condition is of type *not*, the Execution Engine evaluates the condition as *false* and checks for actions inside the **else** block, eventually finding that none were specified, as can be seen in Figure 21b.

As there are no actions inside the *else* part of the action block and there are no more actions specified after the current *if* action, and since the task's current state *request* isn't a terminal transaction state, the Execution Engine proceeds the transaction to its next step, which is the

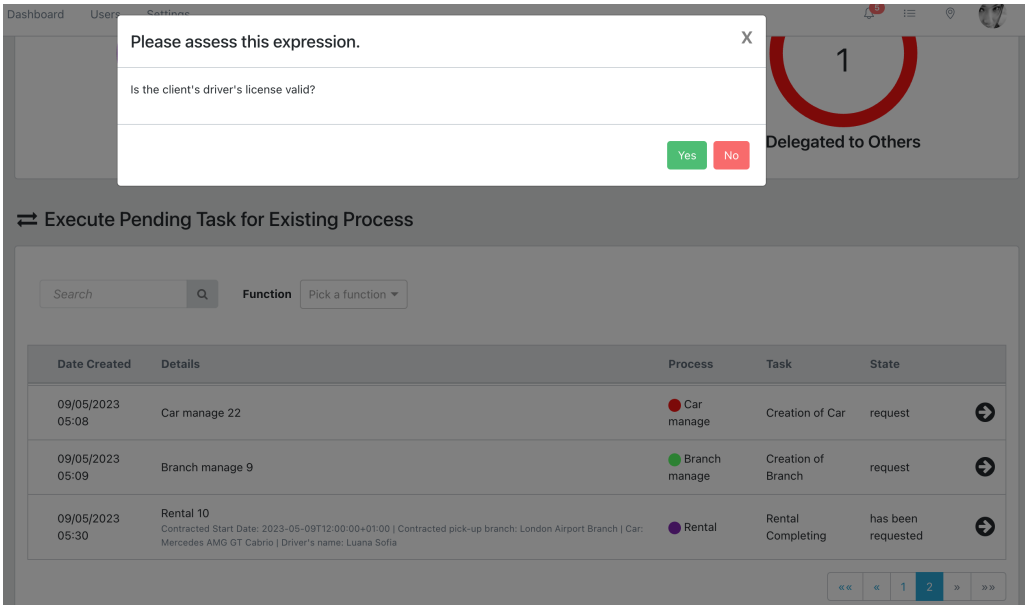


Fig. 38: Dashboard Demonstration - Execution of the *Rental Completing is Requested* Action Rule: If Evaluation of Informal Expression

act of the *promise* state, resulting in the creation of a new task called **Rental Completing is promised**, whose Action Rule is displayed in Figure 39.

This new organizational task is then executed by the Execution Engine. It searches for the respective task's Action Rule's first action, which is a *causal link* action. This automatic execution leads to the creation of the corresponding transaction step, the **Deposit Paying is requested** task, which is executed by the Execution Engine until a user intervention action is found. However, since the *causal link* action doesn't have the *continue if same user* flag active, the action on which the Execution Engine stopped this new task's execution in isn't returned to the client-side for further processing. Instead, the Execution Engine resumes the execution of the **Rental Completing is promised** task.

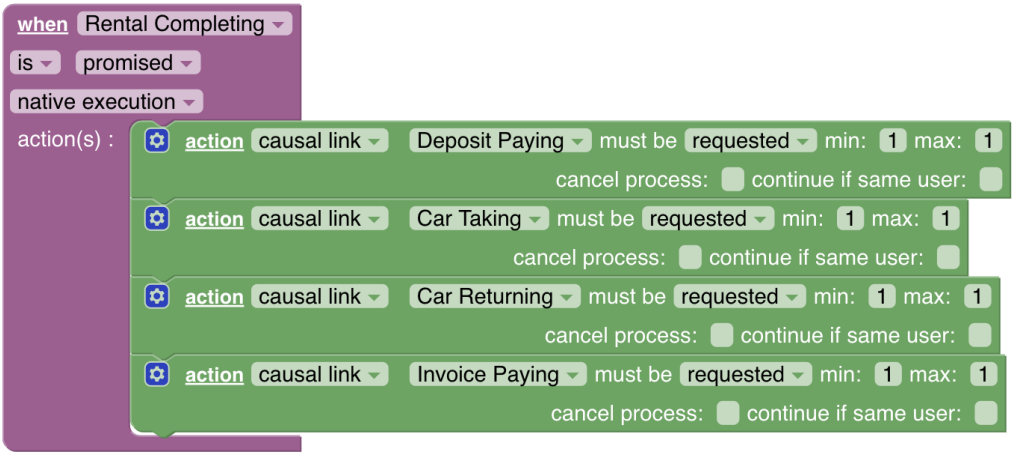


Fig. 39: Dashboard Demonstration - Execution of the *Rental Completing is Promised* Action Rule

Upon finding another *causal link* action after the last executed action, it triggers the corresponding task for the **Car Taking is requested** action. However, since this new transaction step includes a blocking waiting link associated with the *Deposit Paying has been accepted* task, as depicted in Figure 13a, and since this step has not yet been completed in the current process, the *Car Taking is requested* task enters a *waiting* state and its execution is halted, allowing the original task's execution to continue.

The same behaviour is replicated for the final two actions in the original **Rental Completing is promised** Action Rule. The **Car Returning is requested** task encounters a blocking waiting link with the *Car Taking has been accepted* task, while the **Invoice Paying is requested** task must wait for the *Car Returning has been accepted* task to be completed before resuming execution, as illustrated in Figure 13a. As a result, both tasks remain in a *waiting* state until their respective blocking waiting link tasks have been executed.

As this was the last action in the Action Rule, shown in Figure 39, there are no more actions next to the last one performed, and since the task's current state *promise* isn't a terminal transaction state, the Execution Engine proceeds the transaction to its next step, which is the fact of the *promise* state, resulting in the creation of a new task called **Rental Completing has been promised**. This organizational task does not have a specified Action Rule and therefore proceeds the transaction to its subsequent step, which is the *execute* state. Consequently, a new task called **Rental Completing is executed** is created.

However, as this new transaction step has a blocking waiting link referring to the *Invoice Paying has been accepted* task, as can be seen in Figure 13a, and as that transaction step hasn't yet been performed in the current process, the **Rental Completing is executed** task is put into a *waiting* state and its execution is stopped, forcing the current Execution Engine call to end. User B's Dashboard is then updated to reflect the status of the transaction, with the task no longer appearing in the *Pending Tasks* section, but the new **Deposit Paying is requested** task showing up, as can be seen in Figure 40, as User B's *rental completer* role is the initiator organizational role for this transaction type.

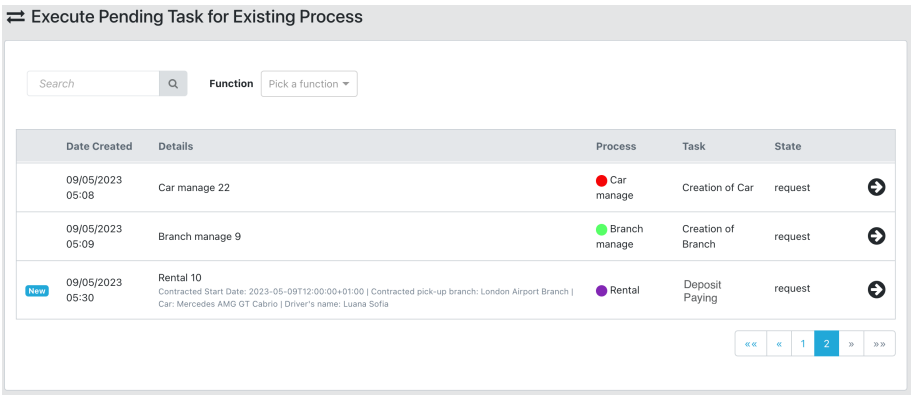


Fig. 40: Dashboard Demonstration - Task for *Deposit Paying is Requested*

Overall, the presented subsection offers a detailed and comprehensive analysis of the functioning of the Execution Engine and its integration with the Dashboard, showcasing its ability to execute tasks efficiently and effectively. The Execution Engine is shown to be a fundamental aspect of

DISME, allowing users to interact with organizational tasks through an intuitive Dashboard. The analysis of the Execution Engine's functioning and its integration with the Dashboard provides valuable insights into the process of executing Action Rules, particularly in the context of complex business transactions.

3.6 Parameterization Component

For all the aforementioned sections to properly function, DISME needs to be correctly parametrized with the organizational artefacts of the information system to be implemented. This subsection introduces an important component developed as part of this project – the parameterization component. Serving as the gateway for administrators or developers to specify the inner details of their organization, this component plays a crucial role in introducing the fine-tuning and customizing the behaviour and functionality of the application. The parameterization component enables administrators to define the scope of interest for the application. By configuring vital elements, including process types, transaction types, entity types, properties, roles and user roles attribution, and even user creation, the component facilitates granular customization and refinement. The development of this component became indispensable, as these organizational details were previously directly manipulated in the system's database. Hence, the parameterization component not only streamlines the domain-defining process, but also ensures the efficient functioning of the DISME application. This subsection not only provides an overview of the component's purpose, but also an overview of its implementation, underscoring its critical role in empowering administrators to specify, edit, and list the essential organizational details required for the successful operation of the application.

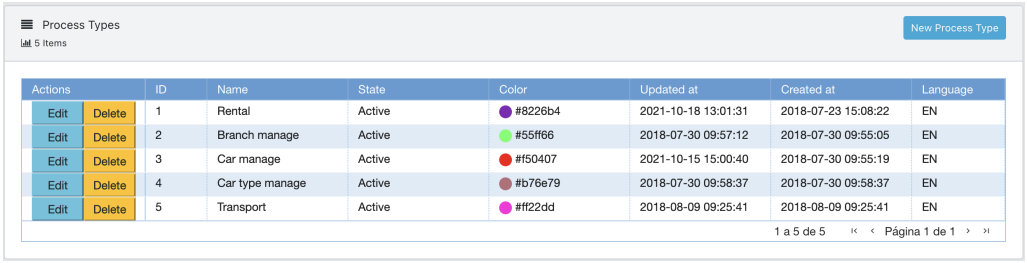
In this subsection, it is important to note that not all of its subsections are accompanied by images. This deliberate omission is due to the similarity observed among these subsections in terms of visual representation. Instead, the focus will primarily be on providing comprehensive textual explanations.

3.6.1 Process Types

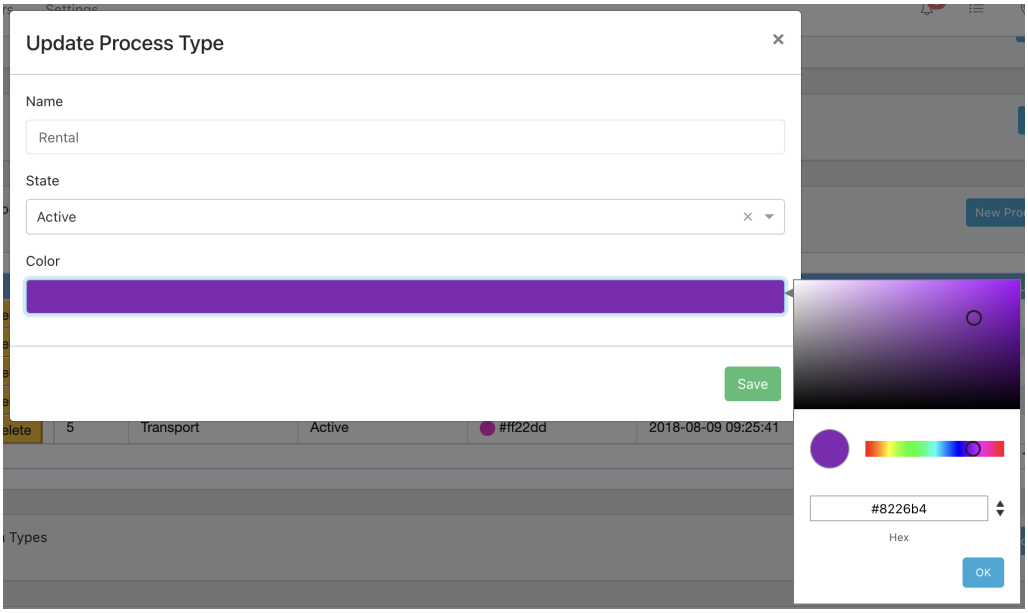
Within the *Process Types* section of the Parameterization Component, authorized individuals have the capability to list, create, and update the organization's different process types, as can be seen in Figure 41.

To create/update a new process type, it is necessary to provide specific details such as its name, state (active or inactive), and a designated colour that is chosen through the utilization of a colour picker, as can be seen in Figure 41a. This chosen colour will be applied to various sections of the Dashboard whenever a task or process associated with this particular *Process Type* is present. This visual representation can be observed in Figures 28, 29, and 30.

The Parameterization Component incorporates translating functionalities, aligning with DISME's multilingual support. When an authorized user accesses the Parameterization Component, any artefacts that have been created in a different language and have not yet been translated into the logged-in user's language are accompanied by a dedicated *translation* button within the list, replacing the regular *edit* and *delete* buttons. Upon clicking the respective button, a modal window is opened, providing the necessary tools to translate the attributes of the artefact that require multilingual definition.



(a) Process Type Listing



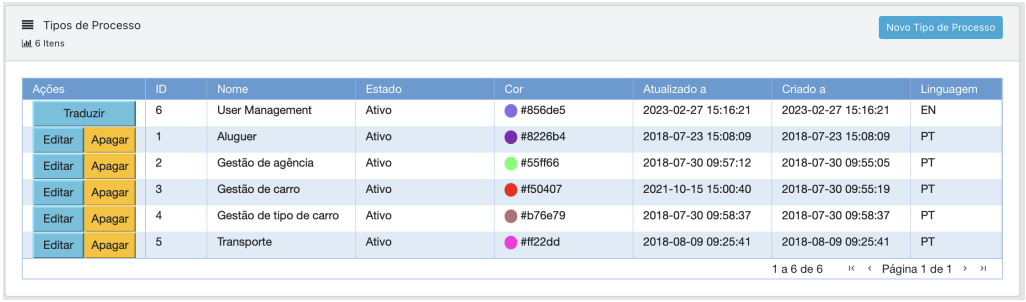
(b) Modal for process type creation/update

Fig. 41: Parameterization Component - Process Types Management

For instance, Figure 42a portrays the scenario where a Portuguese user accesses the *Process Types* section of the Parameterization Component. In this particular case, the *User Management* process type has not yet been translated, and thus, it is presented with the translation button instead of the buttons found on the other process types that have already been translated. The translation modal, as depicted in Figure 42b, allows the user to conveniently insert the translated name for the corresponding process type, utilizing its existing English name as a placeholder to ensure clarity regarding the specific process type being translated.

3.6.2 Transaction Types

Within the Parameterization Component, specifically in the *Transaction Types* section, users are empowered with various functionalities including listing, creating, updating, and translating the organization's distinct transaction types. When creating or updating a transaction type, it is necessary to provide essential information such as its name, state, corresponding process type, and executer role. Additionally, users have the option to specify a result name, as exemplified in Table 2. Moreover, toggle switches were made available for configuring flags associated with the transaction type. These flags include determining whether the transaction type initiates the selected process type, leading to its appearance in the first section of the Dashboard as seen in Figure 28, whether it signifies the completion of the process type upon its own completion, whether it has restricted



(a) Process Type Listing with Translation

(b) Modal for process type translation

Fig. 42: Parameterization Component - Process Types with Translation

access, meaning executer role's tasks belonging to a transaction instance of this type are solely visible to the user responsible for it within the users possessing the executing role, and whether it is a self-activating transaction type. It is noteworthy that the functionality to define the behaviour of a self-activating transaction type is currently disabled in DISME, awaiting further clarification.

To perform the translation of a transaction type, users must input the translated name and, if applicable, an optional result name within the dedicated modal window. It is important to note that the original names of the artefact serve as placeholders to facilitate clear identification during the translation process.

3.6.3 Entity Types

The *Entity Types* section enables users to perform listing, creation, editing, and translation of entity types following the Type Square Pattern illustrated in Figure 1. In order to create an entity type, users are required to provide its name, state, and corresponding transaction type. Additionally, there is an optional field to input an identifying name for the entity type being created or updated. When this field is filled, the entity type's name will be replaced with the identifying name whenever a property refers to this particular entity type. For example, in the context of the RAC case, *rental contract nr* would be used instead of just *Rental*.

Toggle switches are also available to configure flags associated with the transaction type. These flags determine whether the entity type is a one-to-many or many-to-many entity type, as well as whether the entity type is utilized for specifying additional user details beyond those initially requested when an administrator or user creates a user account.

To translate an entity type, users are required to input the translated name and, if applicable, an optional identifying name within the dedicated modal window. It is important to note that

the original names of the artefacts serve as placeholders to ensure clear identification during the translation process.

3.6.4 Properties

The *Properties* section provides authorized users with the functionality to list, create, edit, and translate properties and property unit types, as displayed in Figure 43. When creating or editing a property, as depicted in Figure 43b, users are required to input its name, state (active or inactive), corresponding entity type, value type, and scope (entity, process, or global). Additionally, if there are predefined property unit types available in the system, users can select one from a dropdown menu if desirable.

Furthermore, a new functionality introduced ensures that whenever there is an update made to a property name or an entity type's name, the system automatically propagates those changes to all associated form fields or datagrid labels, respectively. This automated process guarantees the coherence and consistency of the system, as all form labels referring to the updated property or entity type reflect the modified names accurately.

Toggle switches are also provided to configure specific flags associated with the property being created or updated. These flags determine whether the property is editable, meaning it can be modified after being given a value, whether it should be *soft deleted* when updated or deleted, meaning the property value is marked as deleted but still retained in the system in order to maintain a history of its value, and whether it can accept multiple values simultaneously.

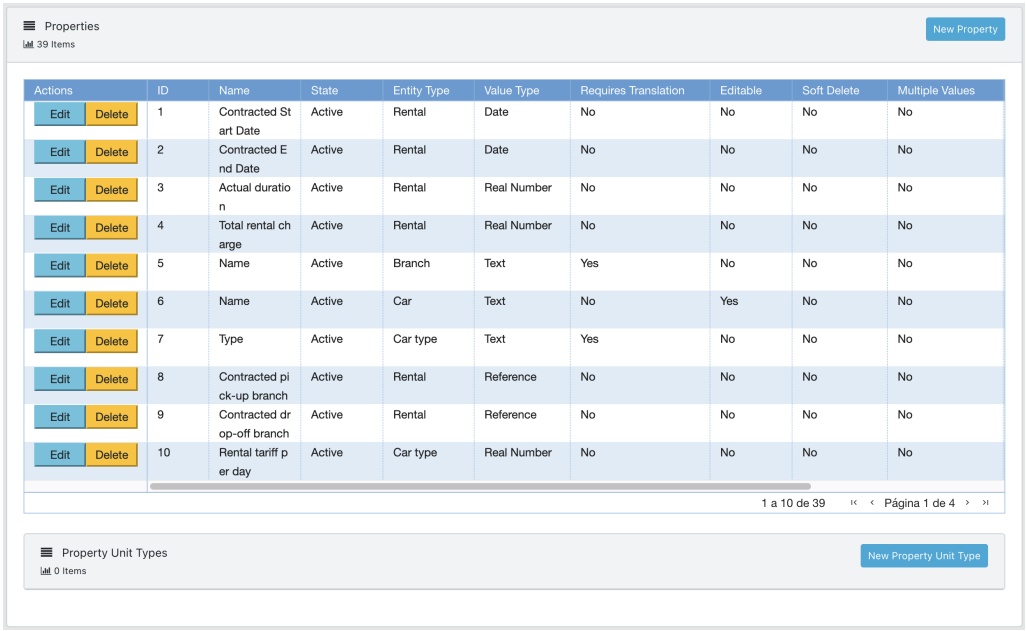
To translate a property, users are required to input the translated name within the dedicated modal window. It is important to note that the original name of the property serves as a placeholder to ensure clear identification during the translation process.

When creating or updating a property unit type, users are only required to provide its name, abbreviation, and state (active or inactive). For example, in the context of RAC, a property unit type could be *horsepower* with the abbreviation *hp*. These minimal details are sufficient to define a property unit type.

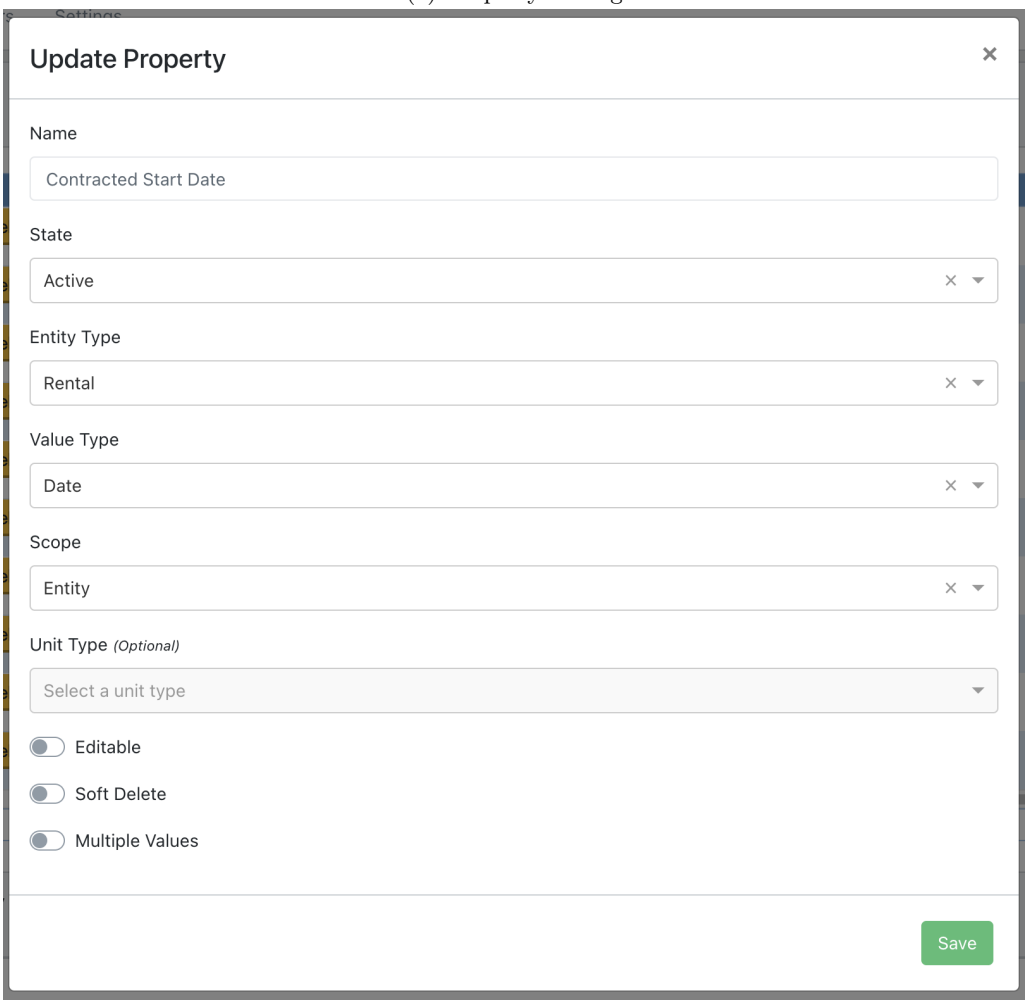
3.6.5 Roles Management

Within the Parameterization Component's *Roles* section, users have the capability to list, create, update, and translate the organizational roles essential for participating in business tasks. This section not only serves as a repository for managing and defining organizational roles, but it's also where it is listed and defined/updated which organizational roles initiate which transaction types and which roles are assigned to the system's users. When creating or editing an organizational role, the only mandatory field to be provided is its name. This single piece of information is crucial for identifying and differentiating roles within the system. Similarly, in the translation modal, the only field to be filled is the role's translated name, where the original name of the role acts as a temporary placeholder to guide the process of inputting the translated name for the respective role.

Within the subsection dedicated to associating initiator roles and transaction types, authorised users can access the functionalities of listing, creating, and editing this crucial information. When assigning an initiator role to a transaction type, it is necessary to provide the relevant details in the creation/update modal. In this modal, two fields must be selected from the provided dropdowns,



(a) Property Listing



(b) Modal for property creation/update

Fig. 43: Parameterization Component - Property Management

specifying the initiator role and the corresponding transaction type. Furthermore, a toggle switch is available to indicate whether the assigned initiator role has restricted access to this transaction type. Enabling this switch signifies that the tasks associated with a transaction instance of this type are exclusively visible to the user responsible for it, among the users possessing the initiator role.

Within the relevant subsection for assigning users their respective organizational roles, authorized users have the ability to perform essential operations such as listing, creating, and editing this important information. When assigning a role to a user, it is imperative to provide the necessary details within the creation/update modal. In this modal, two fields are required to be filled by selecting the appropriate options from the provided dropdown menus. These fields indicate the user and the corresponding role, establishing the association between them.

3.6.6 User Management

The *Users* section provides users with the functionalities of listing, creating, and editing user information. When creating a user using this component, several details must be specified, including the user's name, username, language, taxpayer identification number (NIF), email, and password. Additionally, users have the option to provide additional details by enabling the designated toggle switch. Once activated, a dropdown menu is presented, containing entity types specifically designed for capturing additional user details. Upon selecting an entity type, the corresponding properties associated with that entity type are displayed for users to fill in the necessary information.

4 Comparing the effort needed to implement an information system with a traditional and a low-code approach: The NexusBRaNT experiment

The objective of an experiment is to evaluate a hypothesis or relationship, as described in [35]. In this project, we decided to run an experiment whose objective was to examine the hypothesis that using a low-code approach for developing an information system requires less effort compared to traditional development approaches. The experiment defines two independent variables: the development method (low-code or traditional) and the selected frameworks. The dependent variables include the development time for both approaches, the lines of code (LOC) for the traditional approach, and the database record count for the low-code approach. The results of the experiment provide insights into the effectiveness of low-code approaches in reducing development efforts and improving efficiency in information system development.

Namely, the experiment focuses on the development of an information system with a traditional development approach, using the Express.js and React frameworks, vs. a low-code approach using the DISME platform that directly executes DEMO models as an information and workflow system. NexusBRaNT was projected as a system to access the BRaNT (Belief Revision applied to Neurorehabilitation Therapy)[5] project's back office. The aim was to produce a digital platform supporting the work of health practitioners who undertake cognitive rehabilitation, such as psychologists, neuropsychologists, and therapists. Its primary characteristics are the management of patients, neuropsychological testing, and cognitive training. This projected system was implemented by two developers, separately, with the two mentioned approaches. And the results were impressive. The traditional approach took 888 hours of work, while the low-code approach took a mere 47.5 hours, 5.37% of the effort needed with the standard approach.

4.1 The NexusBRaNT Case

The BRaNT project aims to develop technological tools for domestic cognitive rehabilitation with artificial intelligence assistance and to provide solutions for more resilient healthcare systems. Its goal is to improve its patients' quality of life through more frequent and customized interventions.

One of the tools being developed in this research project is NeuroAIreh@b, whose main goal is the development of a cognitive profile that optimizes the adaptation of prescriptions through artificial intelligence, offering therapists various suggestions for a highly customized and adaptable cognitive rehabilitation through activities in a virtual reality-based simulation of daily life.

To access the BRaNT project's back office, an online platform named NexusBRaNT was projected. A web-based system aiming to cater to health professionals who do cognitive rehabilitation (psychologists, neuropsychologists, and therapists, among others). For this system to meet the health professionals' basic needs, it was necessary to clearly specify their various requirements, making it adjustable and comprehensive.

A thorough analysis was conducted to identify the pertinent transactions within the domain of NexusBRaNT. This was achieved through the Transaction Description Table (TDT), proposed in [36], which presents a structured and detailed description of each task. It is important to note that the terms *transaction* and *task* are used interchangeably in this context. Within the table, each

[5] https://www.arditi.pt/en/projetos-finalizados/brant-project.html

task is accompanied by its description, often directly sourced from the text found in the system's requirements. Additionally, the table outlines the conditions or rules that must be verified for the execution of subsequent task(s) and any relevant time constraints associated with each task [36]. A section of this table can be seen in Figure 44, with the full TDT in Appendix D.

The main features and tasks of this system are patient management (adding a new patient, adding and browsing clinical information, and visualizing the patient's progression), management of neuropsychological assessments (creating an assessment session, inserting its instruments' evaluation, viewing its results, providing clinical interpretation and calculating its cognitive performance), and management of cognitive training (creating a cognitive training program, tracking its progress, and viewing each of its sessions in detail). This system also has a user that performs the role of an administrator, and this user's primary functionalities include creating users, viewing and editing its data alongside viewing their associated patients, and associating and disassociating patients from a health professional.

ID	Source	Section	Process	Name	Task Kind	Executing Function	Description	Origin Task(s)	Waits for task(s)	Target Task(s)	Conditions/Rules	Time Constraints
Neuropsychological Assessment Session (26 tasks)												
T31	Reas12/2022	RF 14 RF 14.1 RF 14.2 RF 14.3	**Neuropsychological Assessment Session**	Neuropsychological Assessment Session Creation	O	Health Professional	RF14 - The system should allow the creation of a neuropsychological assessment session associated with a patient. RF14.1 - The system should allow adding more than one assessment instrument in the assessment session. RF14.2 - The HP shall be able to enter data according to the results obtained by the patient in each selected assessment instrument. RF14.3 - The system should allow insertion of clinical interpretation into the assessment instruments when it contains.					
T32	Reas12/2022	RF 11.2	**Neuropsychological Assessment Session**	Neuropsychological Assessment Session Edition	O	Health Professional	RF11.2 - The system should allow the HP to edit the neuropsychological assessment (session number, results, clinical interpretation and observations).	T31 - Neuropsychological Assessment Session Creation				
T33	Reas12/2022	RF 11 RF 11.1 RF 11.5 UC 10	**Neuropsychological Assessment Session**	Check patient's neuropsychological assessment sessions	D	Health Professional	RF11 - The system shall display the patient's neuropsychological assessment sessions with the assessment session number, date, HP, and assessment instrument. RF11.1 - The system shall allow viewing detailed information of each neuropsychological assessment. RF11.5 - The system shall allow filtering/ordering of neuropsychological assessments by session number, date, assessment instrument, and HP. UC10. View neuropsychological assessments of the patient.					
T34	Reas12/2022	RF 10.1	**Neuropsychological Assessment Session**	Cognitive Profile Calculation	I	Health Professional	RF10.1 - The system should allow the calculation of the patient's cognitive profile through the last neuropsychological evaluation session.	T31 - Neuropsychological Assessment Session Creation			Must at least register one assessment instrument that alters the cognitive profile, else it will remain at zero	
T35	Reas12/2022	RF 10.2	**Neuropsychological Assessment Session**	Clinical Interpretation Calculation	I	Health Professional	RF10.2 - The system should allow to calculate the clinical interpretation of the patient through the last neuropsychological assessment session.	T31 - Neuropsychological Assessment Session Creation			Must at least register one assessment instrument that alters the clinical interpretation, else it will remain at zero	
T36	BdNexus		**Neuropsychological Assessment Session**	BDI-II Registration	O	Health Professional		T31 - Neuropsychological Assessment Session Creation				
T37	BdNexus		**Neuropsychological Assessment Session**	CDR Registration	O	Health Professional		T31 - Neuropsychological Assessment Session Creation				

Fig. 44: Transaction Description Table of the NexusBRaNT case.

An extensive analysis was then made to specify the relevant fact types in the domain of BRaNT: the concept types and the attribute types, of which instances are handled in their daily operation. This effort resulted in a global and synthetic view of all of NexusBRaNT domain's concepts while abstracting from their attributes. These fact types are consolidated into a diagram following the Concept and Relationships Diagram (CRD) proposed in [37], which overcomes DEMO's Fact Model ambiguity present in the keywords *entity* and *property* by using the generic terms *concept* and *attribute* and considering them both as facts. Part of the resulting diagram can be observed in Figure 45. We identified a total of 224 fact types in the NexusBRaNT case, with 193 attribute types aggregated into 31 concept types. Of these 31 concept types, 20 of them refer to Neuropsychological Assessment Instruments, which for image legibility reasons, we do not include in the shown diagram, presenting only 6 of them, chosen to showcase some of NexusBRaNT's supported neuropsychological assessment's instruments and their derived concept types.

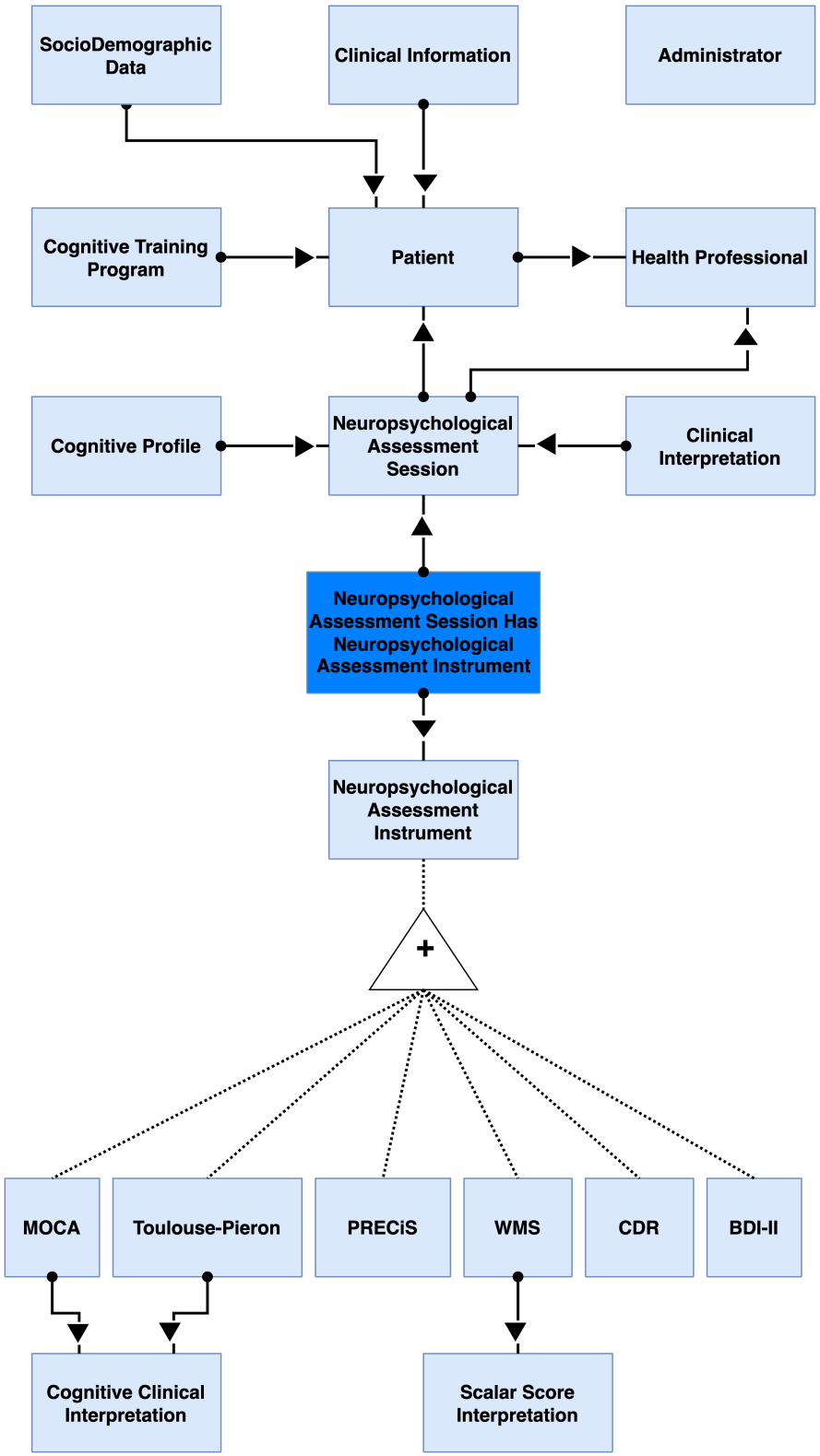


Fig. 45: Concept and Relationships Diagram of the NexusBRaNT case.

In this diagram, arrows are used to express relationships, which will always consist, in practice, of an attribute in one concept whose instances will be a reference to instances of the other concept [37]. The notion of explicit representation of dependency laws is done with a dark-filled circle. By associating this symbol with a concept in one connector, it is indicated that for this concept to exist, an instance of the concept at the other end of the connector must also exist [37]. For example, an instance of a Training Program cannot exist without a reference to an existing instance of a Patient. Many-to-many relationships are represented with an intermediate concept depicted with a darker colour; this concept will have many-to-one relationships with the concepts participating in this many-to-many relationship; both of these relationships will have dependency laws on the side of this intermediate concept; e.g., a Neuropsychological Assessment can have one or more Neuropsychological Assessment Instruments associated with it; and one Neuropsychological Assessment Instrument can be associated with one or more Neuropsychological Assessments as shown in Figure 45 [37].

With the main concepts and relationships known, it becomes necessary to determine the pertinent attributes associated with each concept. This is done through the Concept Attribute Diagram (CAD), which can be considered a variation or "expansion" of the CRD presented previously, proposed in [37]. The CAD utilizes a collapsible box representation to depict concepts. By expanding the box, the attributes associated with the concept are revealed and displayed one per line. The left side of each line specifies the attribute's value type, while the right side presents the attribute's name. Additionally, for categorical value types, a list of possible values is included [37]. To ensure legibility, Figure 46 displays a partial view of the CAD, showcasing selected concepts with expanded attributes. For further clarification, in the next paragraph, we will also list a few examples of attributes associated with the main concepts.

For the Patient concept type, there are the attributes Full Name, Email, Address, Phone Contact and associated Health Professional. The Sociodemographic Data concept type, which is always connected to a Patient, consists of the attributes of Education Level, Date of Birth, Sex, Laterality, Professional Situation, Job Title, Household and Marital Status. Concerning the Clinical Information concept, we have a reference to the Patient, its Type and Description. Concerning the specification of details of the platform's users, we have the Health Professional concept with the attributes Phone Contact, Professional License Number and Joining Date, whereas the Administrator concept type comprises only the Phone Contact attribute. The Training Program concept type's attributes consist of a reference to a Patient, the Starting Date, Number of Training Sessions, Weekly Recurrence, Weekdays, Duration of Each Session, State, and Minimum and Maximum Adaptation Challenge. Concerning the Neuropsychological Assessment Session, we have the reference to its Patient and Health Professional alongside the Session Number and its Realisation Date. To the extent of the Cognitive Profile and the Clinical Interpretation concept types, they have the same attributes as they are evaluated for the same domains, which are General Cognition, Memory, Executive Functions, Language, and Attention. Lastly, each Neuropsychological Assessment Instrument has its own attributes which depend on the instrument's parameters.

Altogether, the 20 instruments available for registration in NexusBRaNT combine for 144 attributes, with each instrument ranging from 2 to 17 attributes. Some of these instruments also have references to the Cognitive Clinical Interpretation or to the Scalar Score Interpretation concept types. These concept types have 2 attributes in common, these being the Interpretation Level and

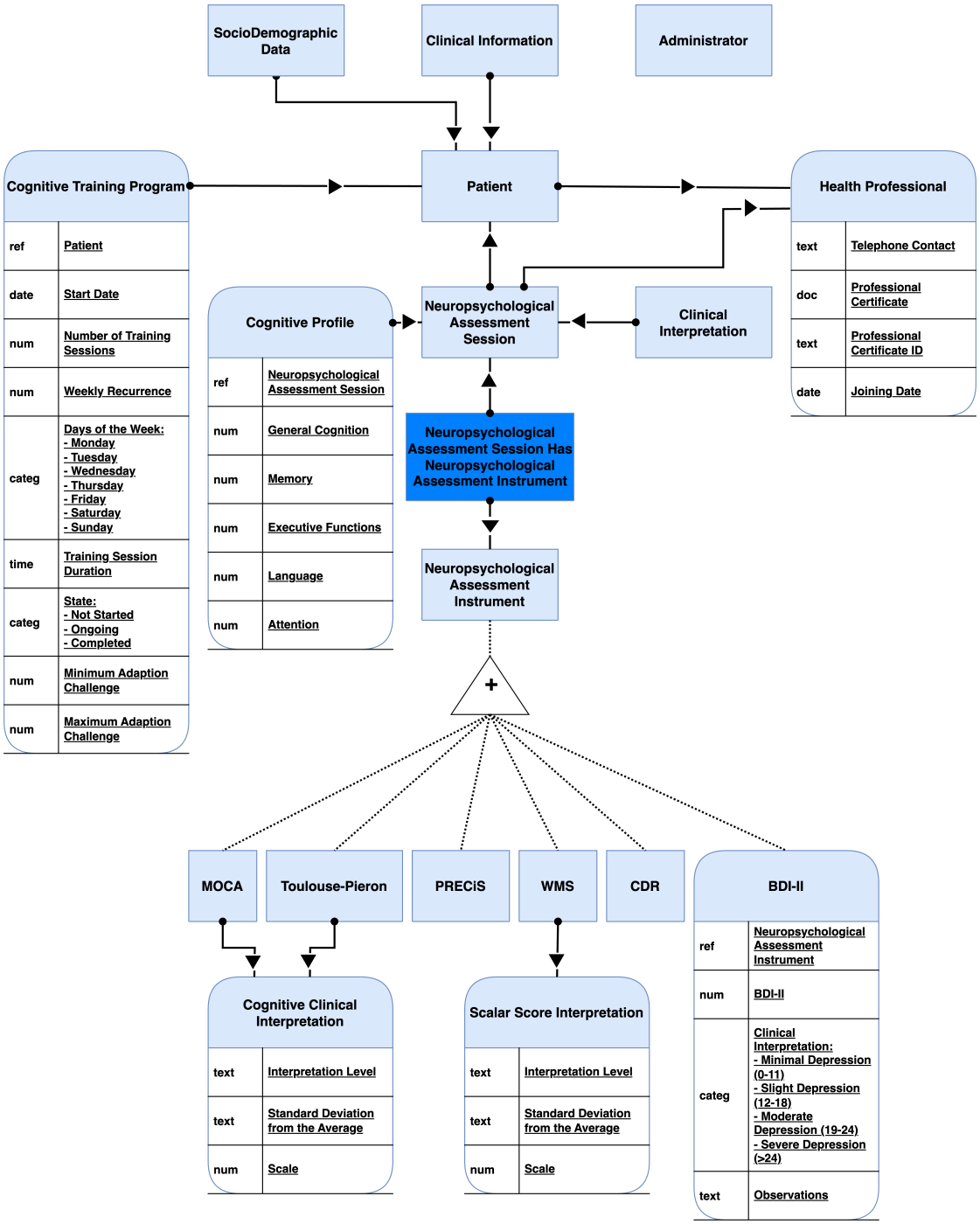


Fig. 46: Concept Attribute Diagram of the NexusBRaNT case.

Scale, and then, for the Cognitive Clinical Interpretation, we have the Standard Deviation from the Average attribute and for the Scalar Score Interpretation, we have the Scalar Score.

To complement the diagrams shown above by providing additional information that is not displayed visually, we have the Fact Description Table (FDT). It mirrors each concept from the diagram and includes a description of every attribute associated with it. This artefact serves as a comprehensive and traceable resource, connecting attributes to their sources and transactional history within the system. An excerpt of the FDT of the NexusBRaNT case can be seen in Figure 47, with the full FDT in Appendix E.

Scope	Source	Concept	Attribute Name	Attribute's value type	Referenced Concept / Category Values	Description	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Cognitive Training Program Instrinsic Concepts (9 attributes)										
Cognitive Training Program Instrinsic Concepts		**Cognitive Training Program**				RF15 - The system should allow the creation of a training program associated with a patient. RF15.1 - The system should allow adding to the training program the data referred to in requirement RF12.1. [RF12.1 - The system should allow accessing the detailed information of the patient's training programs (total number of sessions, date of first session, number of times per week, duration of each session, adaptation challenge and state)]. RF15.1.1 - The system should identify all fields as mandatory.	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Patient	reference	Patient	The patient who will undertake the cognitive training program	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Start Date	date		The date on which the program will start	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Number of Training Sessions	number		Number of training sessions the patient must do for this program	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Weekly Recurrence	number		Number of days per week that the patient must execute this training program	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Days of the Week	category	Monday	Which days of the week this program will take place	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Training Session Duration	number		Duration of each session of the training program	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition

Fig. 47: Fact Description Table of the NexusBRaNT case.

In this depiction of the Fact Description Table, we provide an in-depth analysis of each concept along with its corresponding attributes. The table encompasses a comprehensive overview of all attributes associated with each concept, including their scope, source, concept, name, value type, referenced concept (in case its value type is "reference")/category values (if the value type is "category"), description, and the tasks responsible for creating and modifying the concept. By organizing this information in a structured manner, we gain valuable insights into the characteristics and relationships of each concept within the system. The table serves as a valuable resource for understanding the various attributes associated with each concept, facilitating effective conceptual analysis.

As we can see from this excerpt of the full information system model, the NexusBRaNT system is quite complex in terms of information needs, and we found it to be a good candidate for our first study of comparing low-code and traditional approaches for implementing an information system.

4.2 Method

The process of developing software typically involves a number of different stages, including defining requirements, designing the database model, and actually programming the software itself.

During the initial stages of development, Developer A, Érica Cunha, from the NexusBRaNT development team, worked to specify in a very detailed way the system's requirements and to refine the overall DB model for this project. This process involved many interviews and careful analysis of the needs of the project, as well as consideration of the various components and features that would be required to meet those needs. The requirements, DB structure and overall description of the system to develop were shared with Developer B, the author of this document, to study the requirements and specifications thoroughly.

The developers then started to build the actual software system separately. Developer A followed a traditional approach to software development, using the Express.js and React frameworks, and Developer B followed the low-code approach, using the DISME platform.

As both Developer A and Developer B's master's thesis projects were developed concurrently, we can consider that they possess similar experience in the realm of application development, thus ensuring that this critical variable is at the same level for both developers and strengthening the outcome of the experience. Developer A's master's thesis project [14] involves implementing NexusBRaNT using traditional development methods, while Developer B's project aim was to finalize the development of certain components of the DISME platform and then validate the functionality of the platform with the implementation of a real system. It was then a perfect opportunity to join the effort of these two projects for the presented experiment.

The DISME platform was developed by Developer B as part of the respective master's thesis project and was chosen for the experiment as using another would not make sense. On one hand, developer B had experience in the testing and use of the platform, and, on the other hand, and more importantly, the platform's potential to address most inhibitors for the adoption of LCPs makes it a natural choice for this and other future studies or experiments.

Once the processes, tasks, entities, and properties have been defined, the developer can then move on to designing the Action Rules. These rules are used to specify which actions should be executed in the task flow of the system's engine. Figure 48 is an example of an Action Rule definition, in this case for the task *Neuropsychological Assessment Session Creation*, where there is an action defined for user input of the session's information.

An example of a form created with the Forms Management component in DISME, from the aforementioned Action Rule, can be seen in Figure 49.

After all the system specification is done, users can interact with their daily-life enterprise activities through the system's Dashboard, seen in Figure 50, which serves as the user interface for users to interact when performing organizational tasks following the previously defined system.

4.3 Results and Discussion

For both approaches, a time log for the implementation of each main system task was made, and the information was merged and compiled for comparison. The resulting table, where we present the time required for the traditional software development method vs the low-code approach, can be seen in Table 4.

The table lists the tasks in the middle column, with the times for the traditional implementation approach, in hours, listed in its left-side column and the times for low-code implementation using DISME, also in hours, listed in the right-side columns. In the right-sided columns, we have the

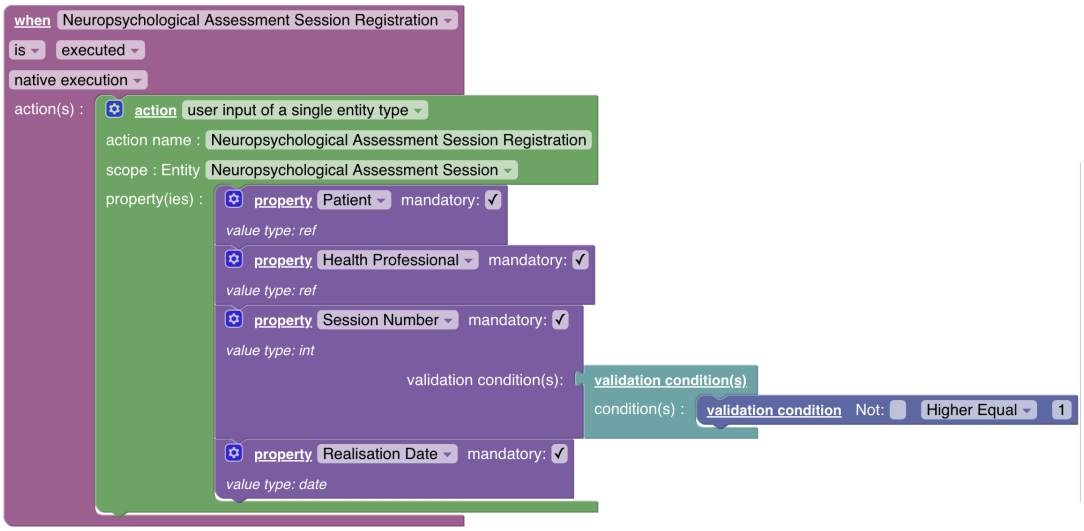


Fig. 48: Example of an Action Rule Definition in the NexusBRaNT case for the registration of a neuropsychological assessment session.

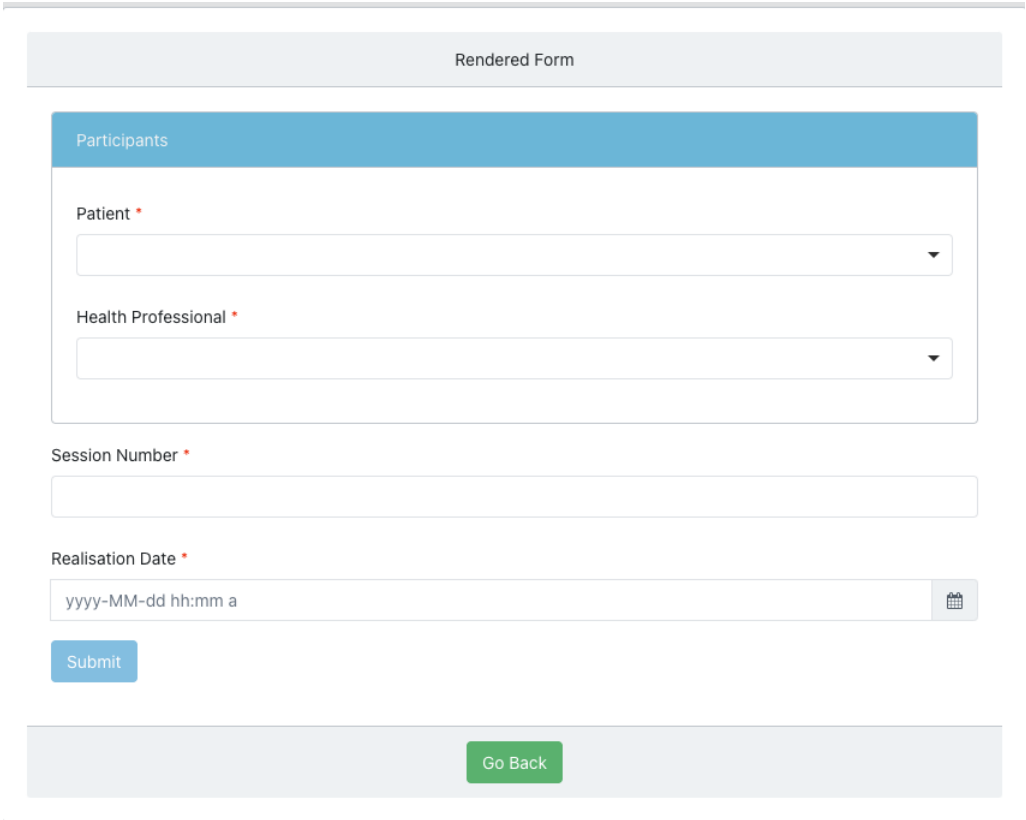


Fig. 49: Example of a Form Design in the NexusBRaNT case for the registration of a neuropsychological assessment session.

Table 4: Comparison of the traditional vs. low-code implementation effort for NexusBRaNT.

Times in the Traditional Approach (Hours)	**Task**	**Implementation times in DISME (Hours)**					
		Processes, Tasks, Entities and Attributes	**Action Rules**	**Forms**	**Pages**	**Total**	**In Comparison To Traditional approach**
30	Login						0%
30	Patient Registration	1	0.5	0.5		2	6.7%
18	Sociodemographic Data Edition	1	0.5	0.5		2	11.1%
12	Patient Listing				1	1	8.3%
12	Patient's Sociodemographic Data Listing				1	1	8.3%
18	Clinical Information Registration and Edition	1	0.5	0.5		2	11.1%
18	Clinical Information Listing				1	1	5.6%
18	Clinical Information Detailed Listing				2	2	11.1%
210	Neuropsychological Assessment Session Registration	4.5	2	1.5		8	3.8%
30	Neuropsychological Assessment Session Edition	1	2	1.5		4.5	15%
30	Neuropsychological Assessment Session Listing				1	1	3.3%
60	Neuropsychological Assessment Session Detailed Listing				2	2	3.3%
60	Clinical Interpretation Registration	1.5	0.5	0.5		2.5	4.2%
30	Training Program Registration	1.5	0.5	0.5		2.5	8.3%
12	Training Program Edition	0.5	0.5	0.5		1.5	12.5%
30	Training Program Listing				1	1	3.3%
18	Training Program Listing (ongoing or finalized)				1	1	5.6%
60	Ongoing Training Program Listing Detailed Listing				1	1	1.7%
60	Intervention Parameters Listing		2		1	3	5%
30	Cognitive Profile Visualization				1	1	3.3%
12	Health Professional Registration	0.5	0.2	0.2		0.9	7.5%
12	Health Professional's Data Edition	0.5	0.3	0.3		1.1	9.2%
12	Health Professional's Detailed Data Listing (for Health Professional)				1	1	8.3%
18	Health Professional's Detailed Data Listing (for Administrator)				1	1	5.6%
12	Health Professional's Associated Patients Listing				1	1	8.3%
12	Associate/Disassociate Patient	0.5	0.1	0.1		0.7	5.8%
6	Administrator's Data Registration and Edition	0.5	0.1	0.1		0.7	11.7%
18	User's Listing				1	1	5.6%
888	**Total**	**14**	**9.7**	**6.7**	**17**	**47.4**	**5.3%**

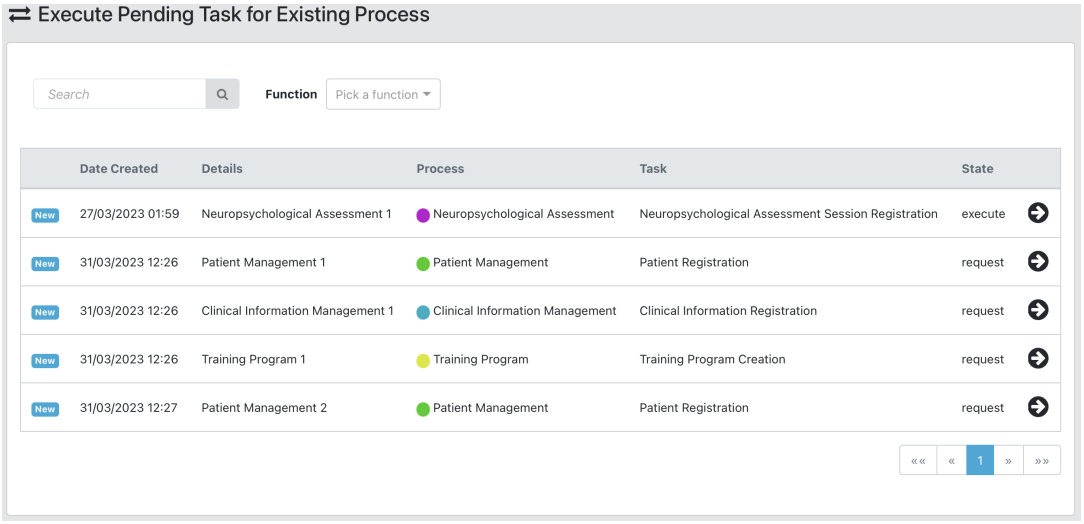


Fig. 50: Example of DISME's Dashboard in the NexusBRaNT case for execution of pending tasks.

development time spent in defining processes, tasks, entities and attributes gathered in one column, as it is all done within the same DISME component, followed by the time spent in Action Rules design, as can be seen in Figure 48. Then, we have the time spent in Forms designing, as can be seen in Figure 49. As NexusBRaNT is a very data-intensive application, a large part of the actions used in Action Rule designing regard user input and thus require form designing.

Next up, we have the time spent in setting up pages for the listing of some information desired by the user, such as the listing of all Patients' information. When this column is filled, with some exceptions that will be brought up in the table analysis, all of its previous right-sided columns are empty, as the user only wants to design a page to see already stored information and thus doesn't need to define anything, except for the page itself, as the information to be displayed is already in the system and one only wants to list it. On the other side, when the other three columns are filled, this one is normally left empty because there is no need to list anything, what is needed is for the registered actions to be executed.

Lastly, columns were added to showcase the total time spent and to exhibit the relative percentage of time spent in low-code implementation tasks when compared to the time spent in the traditional development approach, for every main system task.

As a web-based system, DISME already provides authentication features such as a login for its users, so there is no time spent in a custom login component in this type of development through our platform, which is from the start already a great time-saving option, considering it took approximately 30 hours when done in the traditional development way.

The table shows that low-code development through DISME requires significantly less time as compared to a traditional software method, for all the implemented tasks. For example, for the Neuropsychological Assessment Session and respective Neuropsychological Assessment Instruments registration tasks, the low-code development only took about 4% of the traditional development time. The more significant effort needed in DISME was for the specification of the processes, tasks, entities and attributes, as there are only 4 attributes for the Neuropsychological Assessment Session concept type, but there are 20 concept types expressing 144 different attributes regarding

its Instruments to correctly specify in our platform. As this task results only in forms showing up for Health Professionals to fill, their Action Rules definition is fairly straightforward, as is the designing of their forms. This can be seen through Figure 48 and Figure 49, where we can see the Action Rule and respective Form designs, respectively, concerning the Neuropsychological Assessment Session Registration task.

As for the listing tasks, one can also observe how low-code development with DISME takes a great deal less time than the traditional software development approach. An example of this can be seen in Table 4 through the Patient's Sociodemographic Data Listing task, where the relative development time for the low-code development was only about 8% of the traditional approach. For this kind of task, in DISME, the user makes use of the dynamic search component. This component works in the way of specification of queries based on triplets of property-operator-value, chosen by the user selecting the relevant options in a user-friendly graphical interface and without the need of any programming experience. The user can then save the specified query so that there is no further need to detail it again. There is also the possibility to specify parameters for a query to be introduced at run-time, if one wants, for example, to only retrieve a specific Patient's sociodemographic data. In this case, the Patient is chosen when the user opens the corresponding task's page and the system retrieves the most recent data available in its database.

Although not common in this demonstration of DISME, sometimes there is the need to mix the designing of Action Rules and pages. This happens when there is a need to access an external API in order to retrieve data to exhibit on the desired page. An example of this kind of task can be seen in Table 4 in the Intervention Parameters Listing task. The reason that this needs to happen is that the Intervention Parameters to list are specified in the main BRaNT research project's database. As they aren't added through NexusBRaNT, one needs to access the needed data through an API call from DISME to the BRaNT project. This is specified by simple drag-and-drop operations in our friendly Action Rules design component, where actions are defined for it to automatically scan data provided by the external API that our system can call, facilitating the integration of external information into our local system. As there is only need for the incoming data to be listed on the desired page, no attributes need to be specified in DISME to store this information. This API RESTful component is currently being developed by another colleague also working on DISME.

Another comparison that is possible to do is in terms of the amount of code in the traditional development approach vs the amount of objects in the low-code approach. This is done in Table 5, which presents the lines of code count for the traditional development approach and the database records count of DISME's low-code implementation for the NexusBRaNT project. We remind that we follow in DISME the AOM approach where everything (both types and instances) are objects and for this count, we basically considered the number of records of tables of a type kind, which are the ones created in DISME's System Modeller functional interface. We determined the code line count [38] that resulted from the implementation following the traditional approach for the following types of code: database, client-side and server-side. For the low-code approach, we present the equivalent in terms of types of records, namely: for the database concern, we consider DISME's *Entity Type* and *Property* tables; for the client-side concern we consider the tables storing information regarding *Forms* tables, and we consider the remaining tables of DISME of type kind as specifying the server-side concern.

One could argue that it is not appropriate to compare these two counts, as manually programmed lines of code are of a different nature from records created in a database of a low-code

Table 5: Comparison of traditional implementation code line count vs low-code implementation database record count for NexusBRaNT.

Traditional Implementation Code Line Count		**DISME Implementation Database Record Count**		**Relative percentage**
Database (SQL)	2160	Entity Type and Property tables	428	19.8%
Client-Side	6025	Forms tables	501	8.3%
Server-Side	4125	Other Remaining Tables	794	19.2%
Total	**12310**	**Total**	**1723**	**14%**

platform. Nevertheless, they are both a technical implementation of a shared set of requirements, and we consider that the presented comparison gives an approximate measure of the reduction in complexity of the implementation of an information system, using a low-code approach vs. a traditional one. In the case of comparing the needed effort in time for both approaches, the reduction is impressive.

Our results show that the low-code implementation approach required significantly fewer records compared to the traditional approach's code line count, across the three concerns analysed. Specifically, the total database code line count for the traditional approach was 2160, while the total *Entity Type* and *Property* tables record count for the low-code implementation was 428, indicating an 80.2% of reduction in complexity. Similarly, the total client-side code line count for the traditional approach was 6025, while the total *Forms* tables record count for the low-code implementation was 501, indicating a 91.7% of reduction in complexity. The total server-side code line count for the traditional approach was 4125, while the total remaining tables record count for the low-code implementation was 794, indicating an 80.8% reduction in complexity. Overall, the total global code line count for the traditional approach was 12310, while the total global tables record count for the low-code implementation was 1723, indicating a general reduction of 86% in the complexity of the technical implementation.

Our results show that the low-code development approach required about only 5% of the time spent in traditional development and resulted in a technical implementation with 14% of the complexity, while matching the functionality and improving on the quality of traditional software development methods. As the logic, rules, and forms specifications are based on robust methods and consolidated open-source tools, it is less likely that unexpected bugs appear in the application's execution environment when compared to the traditional development process with manual programming.

The low-code application ends up being more customizable than the traditional approach, in the sense that forms' layouts can be changed anytime without impact on the system's functioning; and information and process flow can be adapted/complemented without having the need to change source code. For example, one could add more Psychological Assessment Session Instruments through the addition of the respective tasks, entities, and attributes in the system's parameterization component, without having to change or adapt the system's database structure as would be done in the traditional development approach, making it accessible to people with less or no programming skills and at the same time preserving the system's database integrity. However, for tasks that require more complex calculations or algorithms, none of which happened in the implementation of the NexusBRaNT case, manual programming effort would be needed in DISME, and it's part of our plans to add to our platform support for programming and execu-

tion of custom code. The traditional development approach was also found to be more complex and required a higher level of coding expertise. The time and cost savings offered by low-code development is particularly beneficial for organizations that need to rapidly develop and deploy applications. The success of the low-code approach in our experiment can be attributed to the use of a low-code platform with a robust set of pre-built components and a highly visual development environment. By using a platform that offers these features, developers can focus on the application logic and user experience, rather than spending time on low-level coding.

5 Validation of the usability of DISME using qualitative and quantitative methods

We next present the method and procedures for the qualitative and quantitative validations performed on the usability of the DISME platform, and in the following subsections the results of these validations, respectively.

5.1 Method and procedures

To evaluate the usability of the DISME platform implementing the NexusBRaNT information system, a comprehensive study was conducted that employed a combination of the Think Aloud method and the System Usability Scale (SUS) to gather qualitative and quantitative data on the platform's usability.

To assess the perceived functionality of the usability of the NexusBRaNT project implemented in DISME, we recruited a sample of health professionals in the field of psychology, with domain knowledge of the modelled processes and implemented system ($N = 10$, nine females and one male, *Mdn* age = 34, age range = 28-58 years). All participants have a Human and Social Sciences background. Namely, among the participants, two are currently pursuing a Bachelor's Degree in Psychology, one is currently pursuing a Master's Degree in Clinical, Health, and Well-Being Psychology, and the remaining seven are health professionals in the field of psychology (Scholar levels: Bachelor's degree $N = 2$; Master's degree $N = 7$; and Doctoral degree $N = 1$). Notably, two of these health professionals are directly associated with the BRaNT project, adding valuable expertise and insights to the study, as can be seen in Table 6.

Table 6: DISME Individual Usability Tests Participant's Demographic Data

Participant ID	**Age**	**Gender**	**Professional Status**
1	28	Female	Health Professional
2	38	Female	BRaNT Health Professional
3	29	Female	MSc Student
4	40	Female	BSc Student
5	58	Female	BSc Student
6	30	Female	BRaNT Health Professional
7	27	Female	Health Professional
8	28	Female	Health Professional
9	29	Female	Health Professional
10	30	Male	Health Professional

The Think Aloud Method is a technique used in usability testing and cognitive psychology to gain insights into a person's thought processes while performing a task. During the think-aloud method, participants are asked to verbalize their thoughts, observations, and decision-making as they navigate through a task or interact with a system. They are encouraged to express their feelings, confusion, and any difficulties they encounter. This method allows researchers to understand

the cognitive and perceptual processes of users in real time. By listening to participants' verbalizations, researchers can gain valuable insights into the strategies, assumptions, and mental models users employ while interacting with a system. This helps identify usability issues, comprehension problems, and areas for improvement [39].

The System Usability Scale is a cost-effective and dependable measure used for evaluating the usability of systems on a global level. It serves as a usability scale that provides reliable insights into the user experience. The SUS is designed to assess the usability of a wide range of systems, including software, websites, and various technological interfaces. It offers a standardized questionnaire that users can complete to evaluate the usability of a system. By employing the SUS, researchers, and practitioners can obtain valuable information regarding the overall usability of a system in a straightforward and efficient manner [40].

In these individual testing sessions, participants were introduced to the DISME platform's fundamentals, including a brief explanation of low-code platforms. Then, the NexusBRaNT project and its main functionalities were presented. Finally, participants were guided through the DISME Dashboard's sections, providing an explanation of each section and its relevance to the project. An overview of the individual testing programme can be seen in Table 7.

Table 7: DISME Individual Usability Tests Programme

Stage	**Task**	**Average Duration (minutes)**
Introduction	DISME and Low-Code	3
	BRaNT and NexusBRaNT	3
DISME Dashboard	Presentation of the DISME Dashboard's functionalities	3
Platform Familiarization	Free Exploration	5-8
Tasks	Patient Registration	3
	Clinical Information Registration	2
	Neuropsychological Session Registration with Instruments	6
	Calculate Cognitive Profile	1
	Training Program Creation	3
SUS Google Forms	SUS Questionnaire	2
Final Feedback	Final Feedback on DISME	3-5
Conclusion	Appreciation	1
	Total Duration (minutes)	35-40

After familiarizing the participants with DISME and the NexusBRaNT case and their respective functionalities, participants were initially given the opportunity for free exploration of the DISME platform, allowing them to familiarize themselves with its interface, features, and functionalities.

Throughout the session, participants had the opportunity to ask questions and provide feedback to ensure their understanding and engagement with the platform. Following this exploration phase, participants were asked to perform a series of specific tasks that represented everyday activities in the NexusBRaNT domain. The tasks were:

- Patient Registration: Participants were instructed to register a new patient in the system, providing relevant personal and demographic information.
- Clinical Information Registration: Participants were required to enter at least one clinical information for the registered patient, such as clinical history, pharmacological therapy, complementary examinations, and clinical record of sessions.
- Neuropsychological Assessment Session Creation: Participants were tasked with creating a session for a neuropsychological assessment, inputting relevant details such as session number and realisation date.
- Neuropsychological Assessment Instrument Registration: Participants were asked to register at least 2 out of the 20 neuropsychological assessment instruments available in the system into the created assessment session. They were also required to verify that the system automatically calculated the patient's cognitive profile based on the test results.
- Cognitive Training Program Creation: Lastly, participants were instructed to create a cognitive training program for the patient, inputting the relevant program's details such as its number of sessions, weekly recurrence and duration of each session.

During the tasks, participants were encouraged to vocalize their thoughts and provide feedback using the Think Aloud method. This allowed researchers to gain insights into the participants' cognitive processes, challenges encountered, and perceptions of the DISME platform's usability. Valuable feedback and suggestions were provided, highlighting both areas for improvement and the strengths of DISME. The presentation that accompanied these individual tests can be seen in Figure 56 in Appendix A.

Additionally, the System Usability Scale was administered to participants upon completion of the tasks. The SUS questionnaire, consisting of ten statements related to usability, was rated on a Likert scale ranging from 1 (strongly disagree) to 5 (strongly agree). The responses were collected and analysed to quantitatively assess participants' perceptions of DISME's usability.

The combination of the Think Aloud method and the SUS questionnaire yielded valuable insights into the user experience of health professionals using a low-code platform for managing information in the context of cognitive rehabilitation.

5.2 Qualitative validation results

The Think Aloud sessions conducted with the participants yielded valuable feedback that emphasized several areas for improvement and positive aspects of DISME.

Currently, in the platform, tasks are separated based on whether they initiate a process or if they require a process to already have been created in order to be executed. Some participants suggested that all tasks should be in the same section, rather than being divided based on the initiation of processes. Some liked the separation but recommended grouping the tasks that are

initiated after a process is started by process type and also to present the assessment session's instruments registration tasks separated by neuropsychological executive function. Some participants also found it illogical to have all system tasks that can be initiated in the context of different processes in one section and suggested dividing them into separate sections. As for the tasks' panels, there was a request to order them in chronological order according to the system's operation, as some participants found the current task order based on popularity confusing. On the topic of the tasks' panels, one participant also recommended clicking on the tasks' panels instead of using their specific buttons for improved usability. One participant also mentioned that notifications accumulated quickly, indicating the need for a more manageable notification system. When registering neuropsychological assessment session instruments, almost all of the participants suggested the automatic filling of the instrument's normative data, as it is based on patient information already inserted in the system and doesn't change, instead of manual insertion based on provided images in the forms. However, the NexusBRaNT implemented in DISME followed strictly the requirements also implemented in the system developed in the traditional way, which implied manual insertion. This change can be easily done in the NexusBRaNT implementation of DISME.

Concerning the positive aspects of the platform, participants found the platform highly engaging and, one in particular, even expressed interest in using it to implement information systems for their own personal projects. All of them praised the platform's intuitive and visually appealing design, describing the forms as simple and easy to navigate. The choice of colours was also widely appreciated, with one participant stating that "I wouldn't change the colours for anything", and another one comparing it directly with the NexusBRaNT application developed in the traditional way, stating that they "didn't like the colour scheme of the original NexusBRaNT system, it was too dark", but liked DISME's colour range and the "ability to easily change the Dashboard's tasks colours", which is done in a straightforward manner in the parameterization component and is directly applied in the Dashboard. The platform's intuitiveness and responsiveness were lauded, making tasks manageable and enjoyable. Additionally, one participant expressed their satisfaction with the division of tasks into well-organized sections, which facilitated navigation and task completion. Lastly, concerning the section where the number of pending tasks and executed tasks are represented in a graphical manner, one participant referred that "the description of the name of the tasks in the statistic wheels when hovering its process-type coded colours is good for people who are colour-blind", as was their case (had difficulty seeing green and red).

These findings from the Think Aloud sessions provide valuable insights for improving the usability and user experience of the DISME platform. The improvement suggestions and negative feedback highlight specific areas where enhancements can be made, such as task organization, filtering options, automatic data filling, and the notification system. The positive feedback reinforces the platform's intuitiveness, user-friendly interface, visually pleasing design, and overall ease of use, reinforcing its potential as a valuable tool in the low-code development field. Incorporating these suggestions and addressing the identified issues will contribute to the ongoing development and optimization of the DISME platform, ensuring an even better user-friendly experience, and increasing the already very high score obtained in the System Usability Scale presented in the next subsection.

5.3 Quantitative validation results

The quantitative analysis of DISME was conducted using the System Usability Scale, which consists of ten statements that participants rated on a scale from 1 (*strongly disagree*) to 5 (*strongly agree*). The collected responses were then subjected to thorough analysis, enabling a quantitative evaluation of participants' perceptions regarding the usability of the DISME platform. The full questionnaire given to participants can be seen in Figure 57 in Appendix B. Additionally, the corresponding collected answers, grouped by question, are available in Figure 58 in Appendix C.

The first statement of the SUS used to assess the usability of DISME focused on participants' likelihood to frequently use the system. The responses from the ten participants involved in the study, which can be seen in Figure 51, showed a high level of positive feedback. The participants' inclination to frequently use the DISME platform can be attributed to its user-friendly design, intuitive features, and its ability to support their professional activities effectively. The high ratings in this question indicate that the system successfully captured the attention and interest of the users, suggesting that it holds promise as a valuable tool for low-code development.

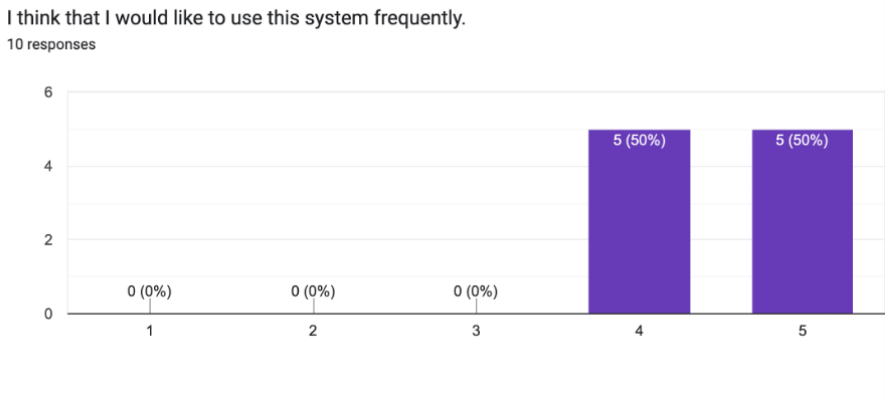


Fig. 51: System Usability Scale's first question scores.

The analysis of the fourth question from the System Usability Scale, which can be observed in Figure 52, sheds light on participants' perceptions regarding the need for technical support when using DISME. Among the participants, six individuals rated the statement with a score of 1, indicating strong disagreement with the notion of requiring technical support to use the system. This suggests that these participants felt confident and capable of independently navigating and utilizing the DISME platform without the assistance of a technical person. Two participants rated the statement with a score of 2, expressing a slightly higher inclination towards perceiving the need for technical support. One participant rated it with a score of 3, indicating moderate agreement with the statement, and another participant rated it with a score of 5, indicating strong agreement. The mixed responses suggest that while the majority of participants felt comfortable using the DISME platform without technical support, a subset of participants expressed varying degrees of agreement regarding the need for such assistance. This indicates that some participants may have encountered challenges or perceived complexities that they believed would require technical guidance. These findings emphasize the importance of providing user-friendly interfaces and clear instructions within the DISME platform to minimize the perceived need for external technical

support. Enhancements aimed at improving user self-sufficiency and reducing potential barriers to independent usage could further enhance the platform's overall usability.

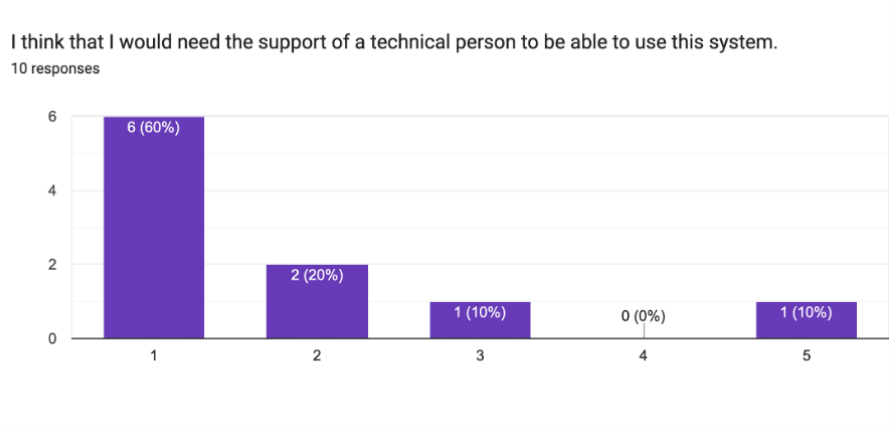


Fig. 52: System Usability Scale's fourth question scores.

It is also noteworthy that all ten participants strongly disagreed with the eighth statement of the SUS, "I found the system very cumbersome to use.", which corresponds to a score of 1 on the Likert Scale, as can be seen in Figure 53. This indicates that participants did not perceive the DISME platform as cumbersome, suggesting that they found it to be user-friendly and manageable. It is encouraging to observe that participants did not encounter significant difficulties or perceive the system as burdensome during their interaction with the DISME platform. This positive feedback reflects the platform's usability and suggests that it effectively supports the tasks and activities of the implemented NexusBRaNT system.

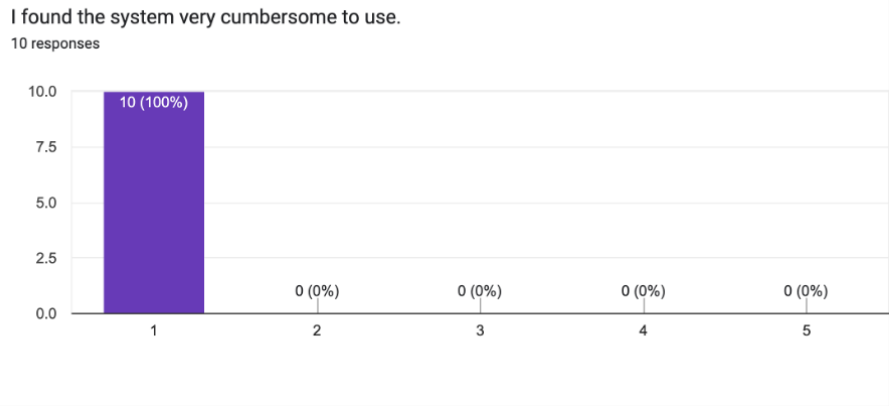


Fig. 53: System Usability Scale's eighth question scores.

To calculate the overall usability score, the ratings were converted into a numerical scale using the SUS equation that can be seen in Equation 1. The obtained average score for the DISME platform was 89.25%. The maximum score obtained was 100%, while the minimum score was 80%, as can be seen in Figure 54, which indicates a favourable level of usability and user satisfaction. This score exceeded the threshold of 68%, which is typically considered above average [40]. Specific

aspects of usability, such as ease of use, system complexity, and confidence in using the platform, received positive ratings from the participants.

$$\text{SUS} = ((\text{Q1-1}) + (5\text{-Q2}) + (\text{Q3-1}) + (5\text{-Q4}) + (\text{Q5-1}) + (5\text{-Q6}) + (\text{Q7-1}) + (5\text{-Q8}) + (\text{Q9-1}) + (5\text{- Q10})) \text{ x } 2{,}5 \quad (1)$$

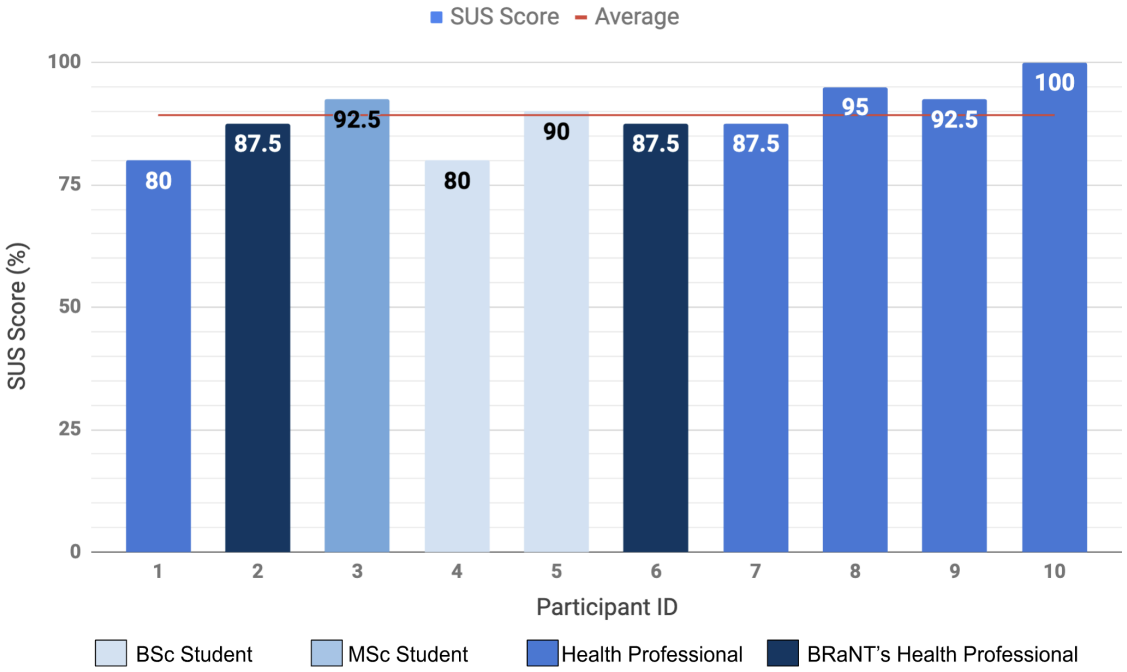


Fig. 54: System Usability Scale scores of the DISME platform implementing the NexusBRaNT system.

Comparing the usability scores of DISME implementing the NexusBRaNT as a low-code platform with the traditionally developed NexusBRaNT system reveals intriguing findings. The DISME platform obtained an average usability score of 89.25%, demonstrating a high level of usability and user satisfaction. The use of low-code development techniques likely contributed to its favourable score by providing a more intuitive and user-friendly experience, allowing for efficient navigation and task completion.

In contrast, the traditionally developed NexusBRaNT system achieved an impressive average usability score of 92%, ranging between 75% and 100%, as can be seen in Figure 55, with a sample of health professionals in the field of psychology with domain knowledge of the modelled processes and implemented system (N = 16, fifteen females and one male, Mdn age = 27.5, age range = 21 - 48 years) [14]. The applied method was identical to the one applied to the tests in the low-code platform, differentiating in the task of creating clinical information that did not exist in this case. All participants also had a Human and Social Sciences background. Namely, among the participants, six are currently pursuing a Master's Degree in Psychology, and the remaining ten are health professionals in the field of psychology. Notably, two of these health professionals are directly associated with the BRaNT project [14]. The obtained scores indicate a generally high level of usability, although there may be some variability in the user experience. The traditional development ap-

proach signifies a robust and well-designed system that meets the needs and expectations of its users.

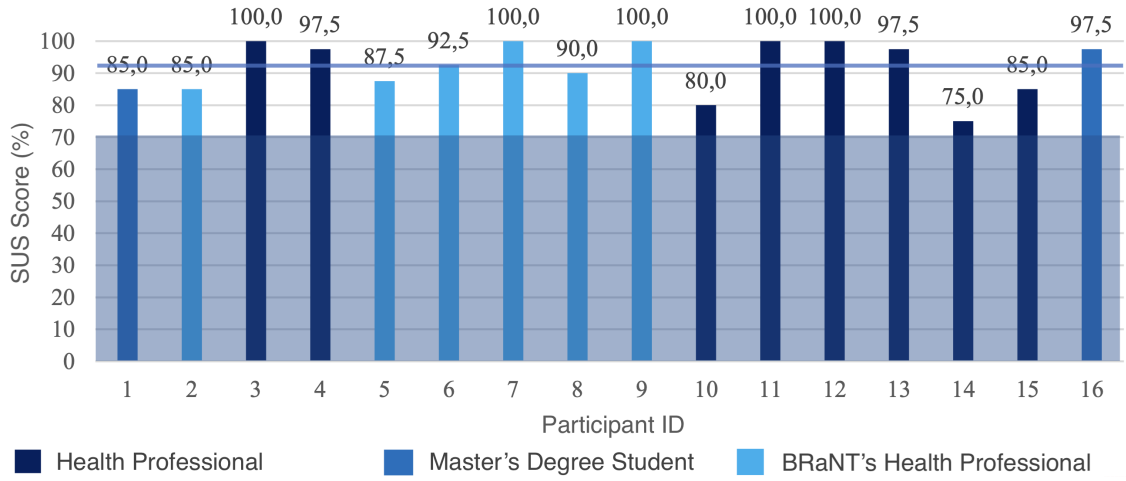


Fig. 55: System Usability Scores of the traditionally developed NexusBRaNT system. [14]

The comparison suggests that both the low-code DISME platform and the traditionally developed NexusBRaNT system offer commendable levels of usability. While the DISME platform benefits from its low-code approach in terms of user-friendliness, the traditionally developed NexusBRaNT system demonstrates its effectiveness in delivering a reliable and highly usable system. These findings highlight the potential benefits of low-code platforms in terms of ease of use and efficiency, while also acknowledging the strengths of traditional development methods in achieving high usability standards.

6 Conclusion

DISME aims to be a distinctive and innovative option among low-code platforms, standing out from other choices in the market, by utilizing unique features like the DEMO foundations or the AOM approach. By being entirely open source, both DISME and the existing plugins it uses, organizations using DISME don't face the risk of becoming locked into a specific vendor. This was the first time DISME was used to implement a complete information system for a concrete institution, and it is not public yet. It's still in an alpha version and will be tested in a couple more real scenarios before we publicly release a beta version, to be made available on GitHub[6].

This master thesis project revolves around the implementation of the DISME platform's Execution Engine, which carries out the execution of Action Rules, and is based on the new specification of the action meta-model and grammar for Action Rules. Throughout the extension of this new action meta-model and grammar for AR specification and development of this main component, it was also noted that extensions and enhancements to existing DISME components were required, these being the Action Rules Management component, the Forms Management Component, and the Dashboard Component. It also became indispensable the introduction of a component to parametrize the system to be run in DISME, so the parameterization component was created.

In the extension of the new action meta-model and grammar for AR specification, we have introduced the concept of decomposing *normal* Action Rules into separate rules for the act and the created fact, which allows for clearer separation of responsibilities in DEMO models. A new *edit entity instance* action type is also introduced to address the need for editing previously filled data in data-intensive information systems. Additionally, the inclusion of new flags in the causal link action specification enhances the behaviour of the Execution Engine.

With these new specifications, adjustments were also needed in the Action Rules Management component, so that it stays updated and acts as a component for Action Rules creation and edition according to the latest definition of the action meta-model and grammar for AR specification. To further improve this component's behaviour, the outdated setup, relying on local files, was replaced with a streamlined approach using an NPM package installation.

In the forms' management component, several updates were also made to follow DISME's latest specifications, such as properties' *multiple values* definition, the addition of support for *has-many entity types*, this is, *entity types* that represent one-to-many or many-to-many relationships through the usage of datagrid components, and the loading of already-stored values when in presence of an *edit entity instance* action.

The main focus of this thesis' project, the Execution Engine, was then developed from scratch in light of evolving requirements and changes in the database structure. It is a server-side controller responsible for executing predefined Action Rules in the system. The fresh start allowed for a more robust and well-documented system to be developed, rectifying the limitations of the previous version, known as the Expression Engine. The Execution Engine's seamless handling of various types of actions showcases the effectiveness and efficiency of the developed system. The automatic execution of actions until user intervention is required, followed by their seamless transition to the client-side for completion, reflects the careful design and thoughtful implementation of the Execution Engine. Furthermore, the emphasis on the significance of logging within the Execution Engine

[6]https://github.com/orgs/EnterpriseEngineeringLab/projects

highlights the attention to detail and commitment to system development, auditing, debugging, and maintenance taken.

After developing the Execution Engine, its integration in the DISME Dashboard was imperative for users to have an interface that empowers them to interact with organizational tasks efficiently. Simultaneously, significant effort was dedicated towards redefining server-side functions to enhance efficiency, updating the functionalities of the remaining buttons within the Dashboard, implementing pagination in relevant sections, and rectifying the filtering behaviour of each section. The successful integration of the Execution Engine, along with the various improvements implemented throughout the component, significantly enhance its functionality and usability, ultimately contributing to the overall effectiveness of the system.

Then, to showcase the Execution Engine's usability and the hypothesis that using a low-code approach for developing an information system requires less effort compared to traditional development approaches, the evolved DISME system was used. On this experiment, it was noticed that a component to parametrize the organizational artefacts that define the system's behaviour was needed instead of focusing on manual insertion of these parameters in DISME's database.

The experiment presented in this thesis demonstrates, with clear metrics, how low-code development has the potential to revolutionize the way organizations approach software development, by offering significant time and cost savings, as well as complexity reduction, while still delivering high-quality and functional applications. The success of the low-code approach can be attributed to the use of a platform, such as DISME, which provides pre-built components and a highly visual development environment, enabling even non-technical users to participate in the development process. Low-code development can be a viable alternative to traditional software development methods, especially for organizations that need to rapidly develop and deploy applications. A potential limitation of the present experiment relates to the reported metrics of the traditional development approach. Given that Developer A, who is at the outset of their professional trajectory as a developer, implemented the NexusBRaNT system in the context of a master thesis, we assume that the development time for this platform could be significantly reduced if an experienced developer would do it. Nonetheless, as both Developer A and Developer B's master's thesis projects were developed concurrently, and even if an experienced developer could halve the development time, the reduction in the effort needed by the utilization of the DISME low-code development platform would remain substantial and impressive, in the order of 90%. The fact that Developer B already had experience with the use and testing of DISME due to developing some final functionalities of the platform, might imply that development time in DISME could be higher by someone not knowing the platform deeply. However, given the usability quality of the platform and given the minimal training necessary for the use of the DISME platform, it is our belief that the difference would be marginal. Even if the developers switched their roles, we consider that the differences would, again, be marginal. In any case, we plan to have, in future experiments, developers without previous experience in DISME. However, it is important to note that the results may also vary depending on the specific requirements of the application and the expertise of the developers and users involved.

The usability of DISME was then evaluated using both qualitative and quantitative methods. The qualitative evaluation through the Think Aloud method provided valuable feedback on various aspects of the platform, including task organization, interface design, and user experience. Participants found the platform engaging, intuitive, visually appealing, and user-friendly. Refinements

were suggested, such as task grouping, improved filtering options, and automation of data filling. The positive feedback reinforced the platform's strengths and highlighted its potential as a valuable tool in low-code development. The quantitative evaluation using the System Usability Scale (SUS) confirmed these findings. Participants agreed that the platform was user-friendly and supported their professional activities effectively, with the overall usability score for DISME being calculated to be 89.25%. As the main limitation, we identify the small sample size. Therefore, results must be interpreted with caution and generalizability of the findings may be limited.

Thus, considering the context of the thesis project, it is concluded that the Execution Engine and the remaining components of DISME were successfully developed and tested, and the objectives have been fully met.

6.1 Future Work

To further improve the work carried out in this thesis project, several recommendations are presented.

One crucial aspect is the further refinement of the EBNF to specify the syntax of Action Rules, and consequently the evolution of the system's database and Execution Engine. Specifically, there is a need to incorporate dates as a new type of term within the EBNF syntax, expanding the range of possible inputs. Subsequently, these dates should be appropriately handled in the Action Rules Management component and processed by the Execution Engine. Additionally, there is the implementation of the processing of *while* and *for* cycles by the Execution Engine, as well as other action types that have not yet been defined. Once the syntax for these action types is established, they can be integrated into the design of Action Rules and subsequently handled by the Execution Engine.

Another area of focus for future work is the expansion of functionalities within the form components. These components play a significant role in facilitating user interaction and data input. Therefore, enhancing their capabilities and making them more versatile would greatly benefit the system's usability and effectiveness.

Furthermore, the Dashboard component requires further refinement to improve its functionality and user experience. This component serves as a central hub for users to interact with organizational tasks and monitor their progress. Therefore, investing effort into refining the Dashboard component would contribute to a more intuitive and efficient user interface.

Further research is also needed to better understand the strengths and weaknesses of each approach, low-code and traditional, and to develop guidelines for choosing the most appropriate method for different types of applications. For instance, future studies could investigate how to integrate low-code development with other emerging technologies such as artificial intelligence, machine learning, or blockchain to develop more complex and sophisticated applications.

Moreover, research should also focus on identifying key factors that determine the success of low-code development, such as the role of user feedback, the importance of testing, and the scalability of the applications. Additionally, it would be beneficial to compare the cost-effectiveness of low-code development versus traditional development methods in different settings, such as small vs. large organizations, or in-house vs. outsourced development. By addressing these questions, we can gain a deeper understanding of the potential of low-code development and how it can be used to drive digital transformation and innovation in various industries.

In the context of usability testing, for future work we foresee additional evaluations in the context of new implementations in other scenarios, continuous refinement of the DISME platform, including support for collaborative modelling and development. With ongoing efforts, we envision DISME becoming an invaluable tool for rapidly developing effective and user-friendly information systems in various domains.

References

[1] S. Dalal and R. S. Chhillar, "Case studies of most common and severe types of software system failure," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, pp. 341–347, 2012.

[2] F. Shull, V. Basili, B. Boehm, A. Brown, P. Costa, M. Lindvall, D. Port, I. Rus, R. Tesoriero, and M. Zelkowitz, "What we have learned about fighting defects," in *Proceedings Eighth IEEE Symposium on Software Metrics*, 2002, pp. 249–258.

[3] A. Zeller and R. Hildebrandt, "Simplifying and isolating failure-inducing input," *IEEE Transactions on Software Engineering*, vol. 28, no. 2, pp. 183–200, 2002.

[4] M. Ibraigheeth and S. A. Fadzli, "Core Factors for Software Projects Success," *JOIV : International Journal on Informatics Visualization*, vol. 3, pp. 69–74, 2019, number: 1.

[5] A. C. Bock and U. Frank, "Low-Code platform," *Business & Information Systems Engineering*, vol. 63, no. 6, pp. 733–740, Dec. 2021.

[6] P. M. Gomes and M. A. Brito, "Low-code development platforms: A descriptive study," in *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, 2022, pp. 1–4.

[7] R. Waszkowski, "Low-code platform for automating business processes in manufacturing," *IFAC-PapersOnLine*, vol. 52, no. 10, pp. 376–381, 2019, 13th IFAC Workshop on Intelligent Manufacturing Systems IMS 2019.

[8] P. Vincent, Y. Natis, K. Iijima, J. Wong, S. Ray, A. Jain, and A. Leow, "Magic quadrant for enterprise low-code application platforms," 2020.

[9] S. Käss, S. Strahringer, and M. Westner, "Drivers and inhibitors of low code development platform adoption," in *2022 IEEE 24th Conference on Business Informatics (CBI)*, vol. 01, 2022, pp. 196–205.

[10] J. Dietz and H. Mulder, *Enterprise Ontology: A Human-Centric Approach to Understanding the Essence of Organisation*, 01 2020.

[11] M. Dumay, J. Dietz, and H. Mulder, "Evaluation of demo and the language/action perspective after 10 years of experience," 07 2008.

[12] A. Perinforma, *The Essence of Organisation: An Introduction to Enterprise Engineering.* Sapio Enterprise Engineering, 2015.

[13] J. L. G. Dietz and H. B. F. Mulder, "The PSI theory: Understanding the operation of organisations," in *Enterprise Ontology: A Human-Centric Approach to Understanding the Essence of Organisation*, J. L. G. Dietz and H. B. F. Mulder, Eds. Cham: Springer International Publishing, 2020, pp. 119–157.

[14] Cunha, "Nexus brant backoffice para brant," Jan 2023. [Online]. Available: http://hdl.handle.net/10400.13/5102

[15] J. W. Yoder and R. Johnson, *The Adaptive Object-Model Architectural Style*. Boston, MA: Springer US, 2002, pp. 3–27.

[16] H. Ferreira, A. Aguiar, and J. Faria, "Adaptive object-modelling: Patterns, tools and applications," 09 2009, pp. 530–535.

[17] H. Ferreira, F. Correia, A. Aguiar, and J. Yoder, "The lazy semantics pattern on the context of meta-architectures," 01 2011.

[18] M. Andrade, "Modelação e implementação de software de apoio a processos de logística," Master's thesis, University of Madeira, 2018.

[19] M. Andrade., D. Aveiro., and D. Pinto., "Demo based dynamic information system modeller and executer," in *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KEOD,*, INSTICC. SciTePress, 2018, pp. 383–390.

[20] A. Hevner and S. Chatterjee, "Design science research in information systems," in *Design Research in Information Systems: Theory and Practice*. Boston, MA: Springer US, 2010, pp. 9–22.

[21] A. Hevner, "A three cycle view of design science research," *Scandinavian Journal of Information Systems*, vol. 19, 01 2007.

[22] D. Aveiro and V. Freitas, "A new action meta-model and grammar for a demo based low-code platform rules processing engine," in *Advances in Enterprise Engineering XVI*, C. Griffo, S. Guerreiro, and M. E. Iacob, Eds. Cham: Springer Nature Switzerland, 2023, pp. 33–52.

[23] D. Aveiro, V. Freitas, E. Cunha, F. Quintal, and Y. Almeida, "Traditional vs. low-code development: comparing needed effort and system complexity in the nexusbrant experiment," in *25th IEEE Conference on Business Informatics, CBI 2023, Prague, Czechia, June 21-23, 2023 (forthcoming)*. IEEE.

[24] M. Andrade, D. Aveiro, and D. Pinto, "Bridging ontology and implementation with a new demo action meta-model and engine," in *Advances in Enterprise Engineering XIII*. Cham: Springer International Publishing, 2020, pp. 66–82.

[25] W. E. Deming, *Out of the Crisis*. The MIT Press, 10 2018. [Online]. Available: https://doi.org/10.7551/mitpress/11457.001.0001

[26] J. Dietz, J. Hoogervorst, A. Albani, D. Aveiro, E. Babkin, J. Barjis, A. Caetano, P. Huysmans, J. Iijima, S. Kervel, H. Mulder, M. Op 't Land, H. Proper, J. Sanz, L. Terlouw, J. Tribolet, J. Verelst, and R. Winter, "The discipline of enterprise engineering," *International Journal of Organisational Design and Engineering*, vol. 3, pp. 86–114, 05 2013.

[27] J. L. G. Dietz, "On the nature of business rules," in *Advances in Enterprise Engineering I*. Springer Berlin Heidelberg, 2008, pp. 1–15.

[28] *ISO/IEC 14977:1996 Information Technology - Syntactic Metalanguage - Extended BNF*, Std., 1996.

[29] P. Bollen, "Sbvr: A fact-oriented omg standard," in *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*, R. Meersman, Z. Tari, and P. Herrero, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 718–727.

[30] "Introduction to blockly | google developers," Mar 2022. [Online]. Available: https://developers.google.com/blockly/guides/overview?hl=en

[31] "Blockly | google developers." [Online]. Available: https://developers.google.com/blockly

[32] "The most advanced wysiwyg editor: Trusted rich text editor." [Online]. Available: https://www.tiny.cloud/

[33] "Form.io | a form and data management platform." [Online]. Available: https://form.io/

[34] Symfony, "The expression language component (symfony docs)." [Online]. Available: https://symfony.com/doc/current/components/expression_language.html

[35] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[36] D. Pinto, D. Aveiro, D. Pacheco, B. Gouveia, and D. Gouveia, "Validation of demo's conciseness quality and proposal of improvements to the process model," in *Advances in Enterprise Engineering XIV*, D. Aveiro, G. Guizzardi, R. Pergl, and H. A. Proper, Eds. Cham: Springer International Publishing, 2021, pp. 133–152.

[37] B. Gouveia, D. Aveiro, D. Pacheco, D. Pinto, and D. Gouveia, *Fact Model in DEMO - Urban Law Case and Proposal of Representation Improvements*, 04 2021, pp. 173–190.

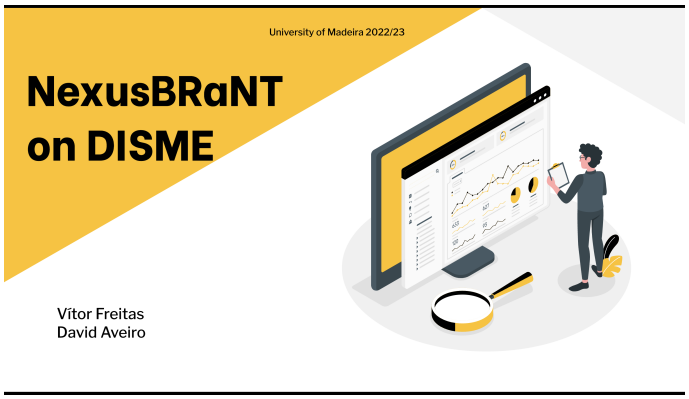
[38] AlDanial, S. Snel, Jolkdarr, C. Beckmann, MichaelDimmitt, , Roman, G. Chaves, J. Wilk, BoB Rudis, Asrmchq, A. Gough, J. Tang, J. Dursi, Achary, A. Ali, C. Ebberson, L. David, D. Ulrich, Erkmos, L. Brinkhoff, LoganDark, T. Irländer, W. Rösler, B1f6c1c4, S. HOUZÉ, A. Ryan, A. Shinn, A. Mastrean, A. Molinaro, and A. Turner, "Aldanial/cloc: 1.92," 2021. [Online]. Available: https://zenodo.org/record/5760077

[39] D. W. Eccles and G. Arsal, "The think aloud method: what is it and how do i use it?" *Qualitative Research in Sport, Exercise and Health*, vol. 9, no. 4, pp. 514–531, 2017. [Online]. Available: https://doi.org/10.1080/2159676X.2017.1331501

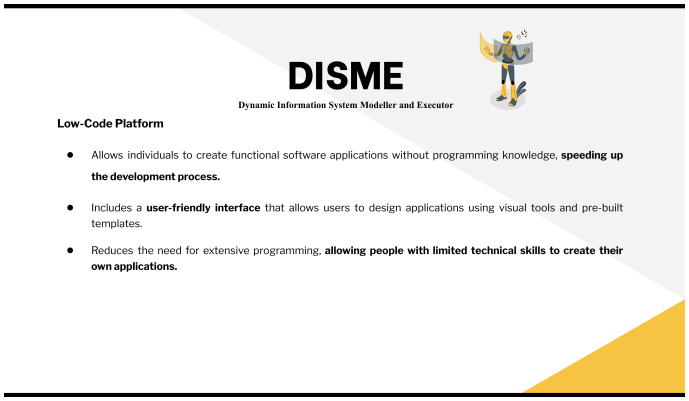
[40] J. Brooke, "Sus: A quick and dirty usability scale," *Usability Eval. Ind.*, vol. 189, 11 1995.

A System Usability Scale Testing Presentation

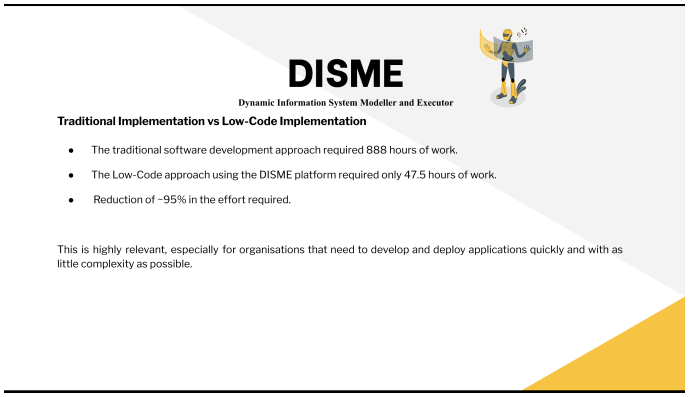
Below are a collection of images showcasing the slides from the presentation that accompanied the usability tests. These images offer a glimpse into the presentation given throughout DISME's usability testing.



(a) SUS Presentation - Cover

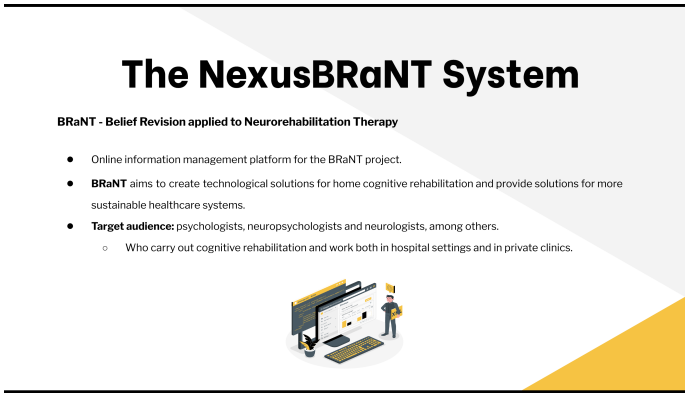


(b) SUS Presentation - Low-Code Platforms

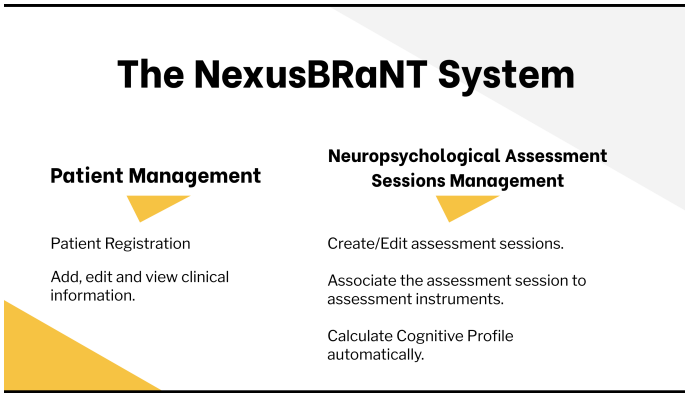


(c) SUS Presentation - Traditional vs Low-Code Experiment

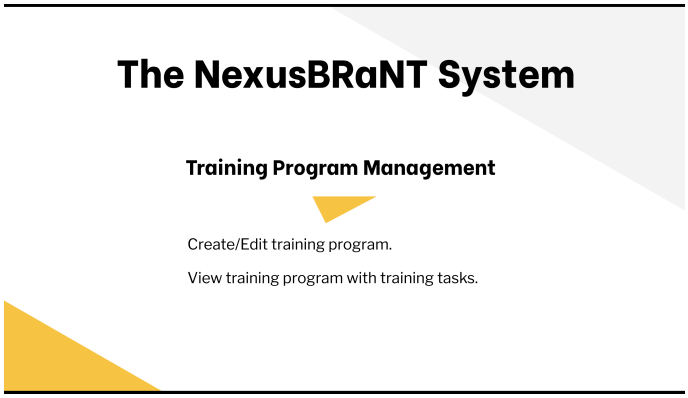
Fig. 56: SUS Presentation Slides



(d) SUS Presentation - The NexusBRaNT System

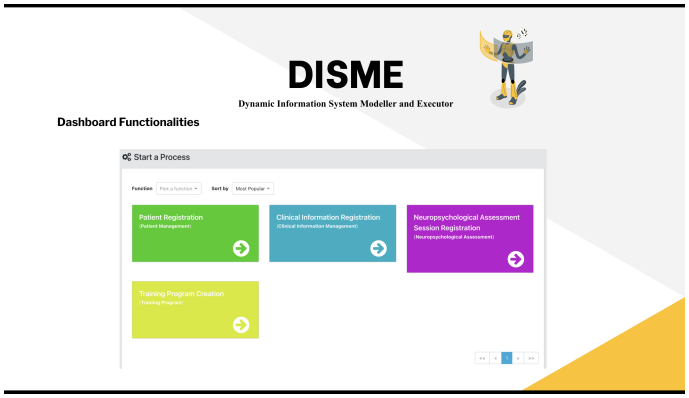


(e) SUS Presentation - The NexusBRaNT System Main Functionalities

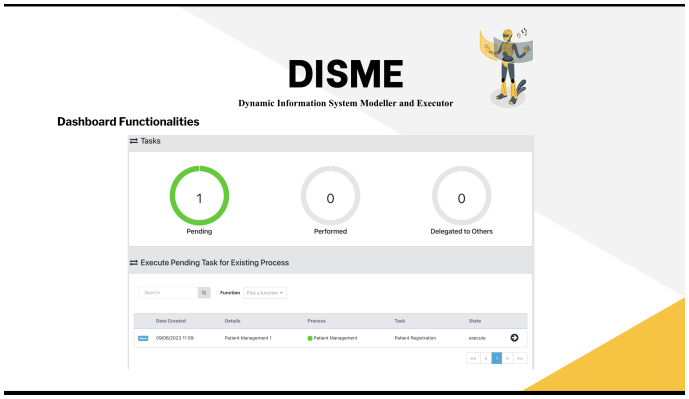


(f) SUS Presentation - The NexusBRaNT System Main Functionalities II

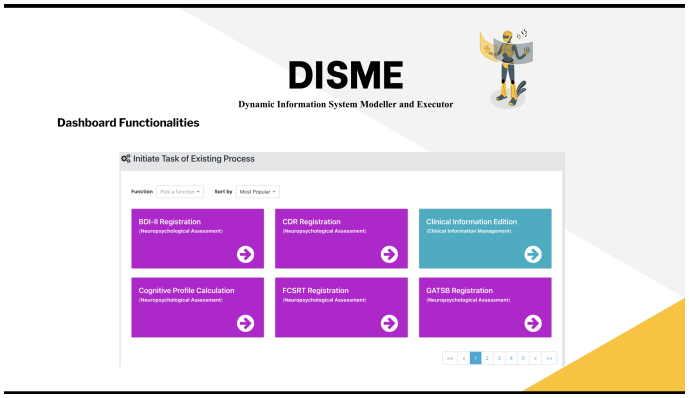
Fig. 56: SUS Presentation Slides



(g) SUS Presentation - DISME Dashboard Functionalities - Section 1

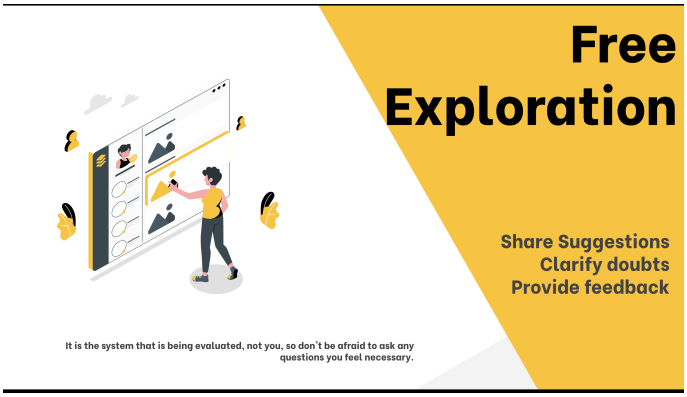


(h) SUS Presentation - DISME Dashboard Functionalities - Section 2

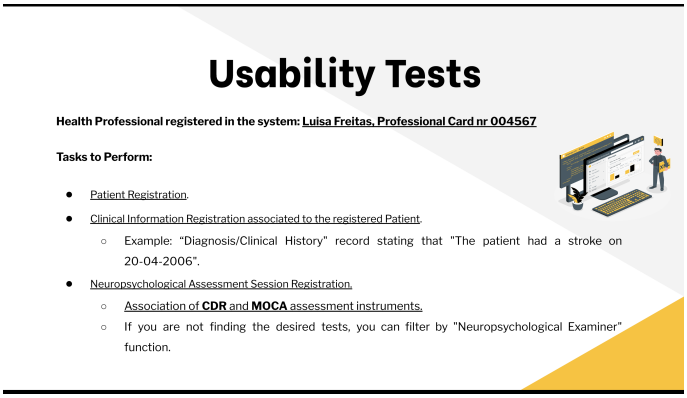


(i) SUS Presentation - DISME Dashboard Functionalities - Section 3

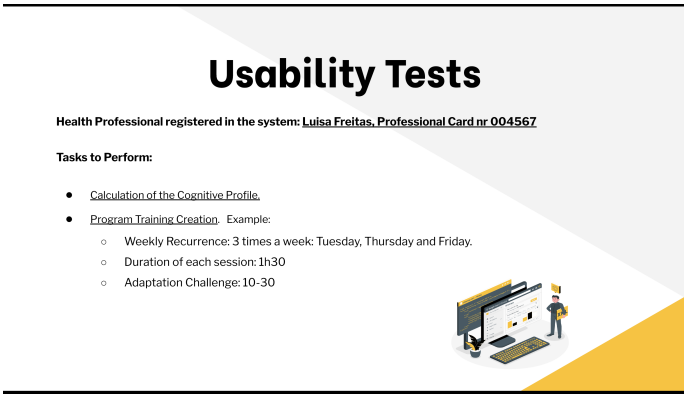
Fig. 56: SUS Presentation Slides



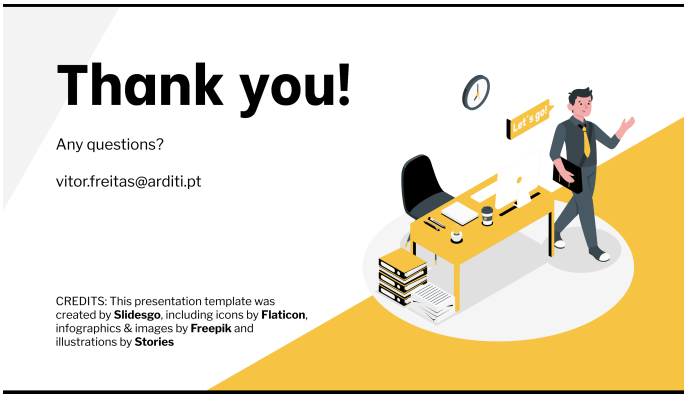
(j) SUS Presentation - Free Exploration



(k) SUS Presentation - Usability Tests Tasks



(l) SUS Presentation - Usability Tests Tasks II

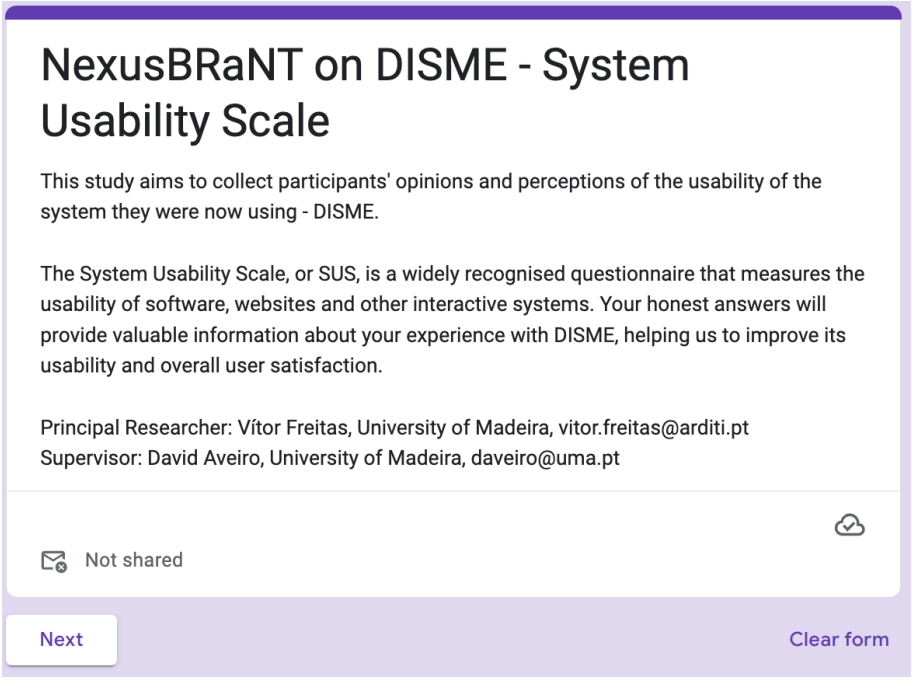


(m) SUS Presentation - Appreciation

Fig. 56: SUS Presentation Slides

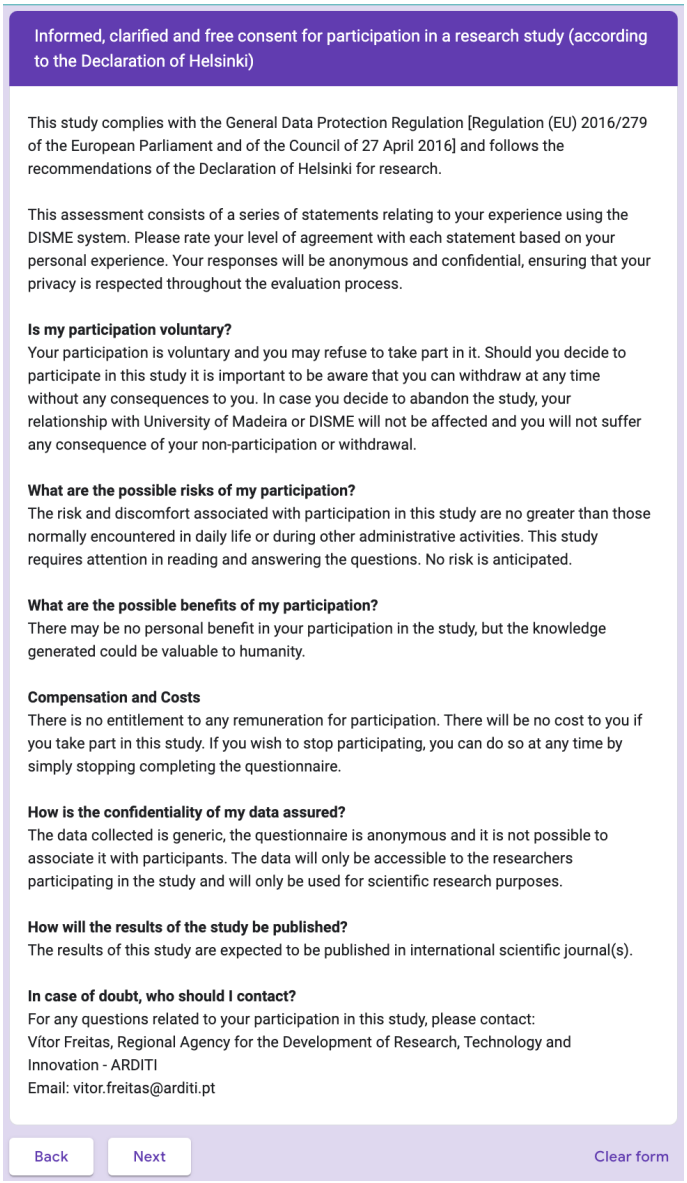
B System Usability Scale Testing Questionnaire

Below is a collection of images showcasing the SUS questionnaire provided to participants. These images offer a glimpse into the SUS questionnaire and its layout.

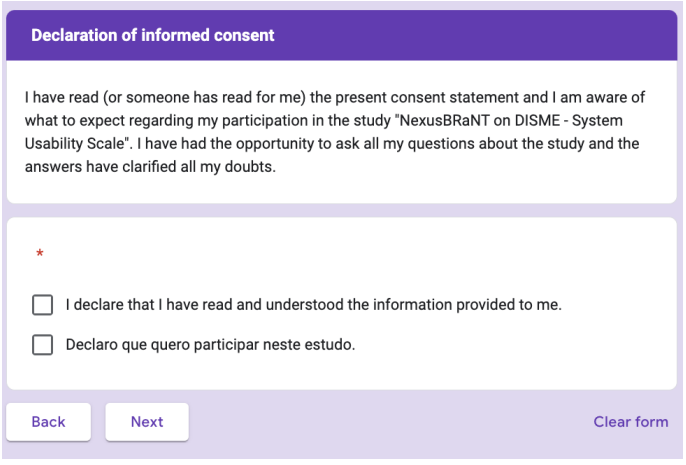


(a) SUS Questionnaire - Introduction

Fig. 57: System Usability Scale Questionnaire

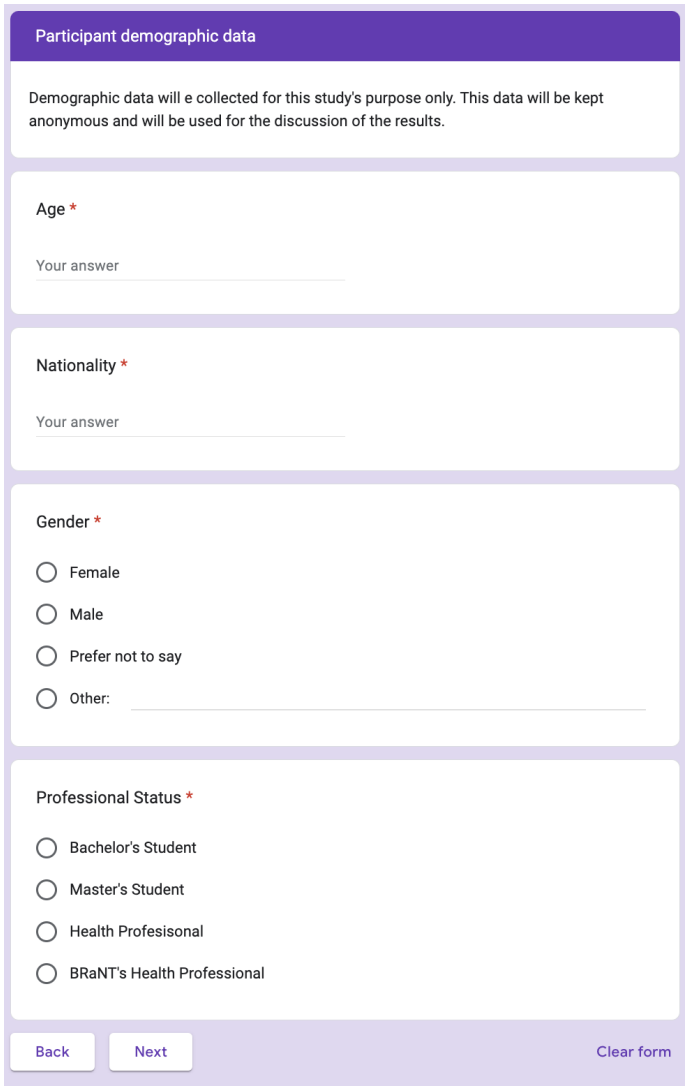


(b) SUS Questionnaire - Informed Consent Information

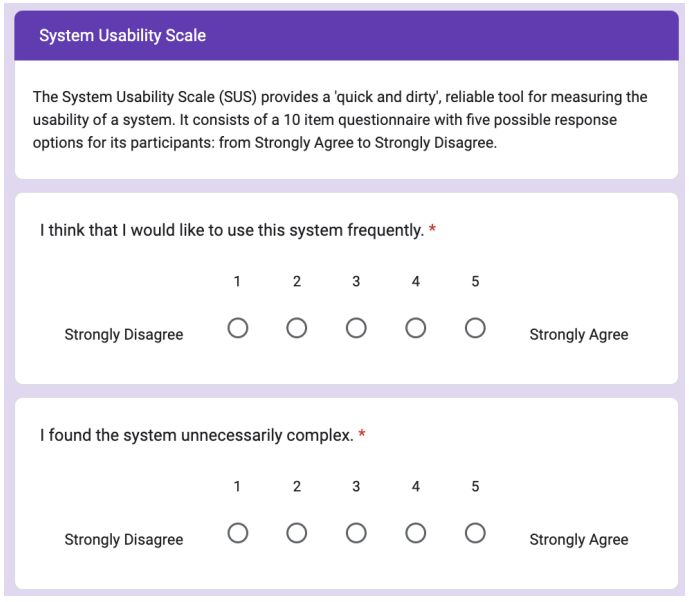


(c) SUS Questionnaire - Declaration of Informed Consent

Fig. 57: System Usability Scale Questionnaire

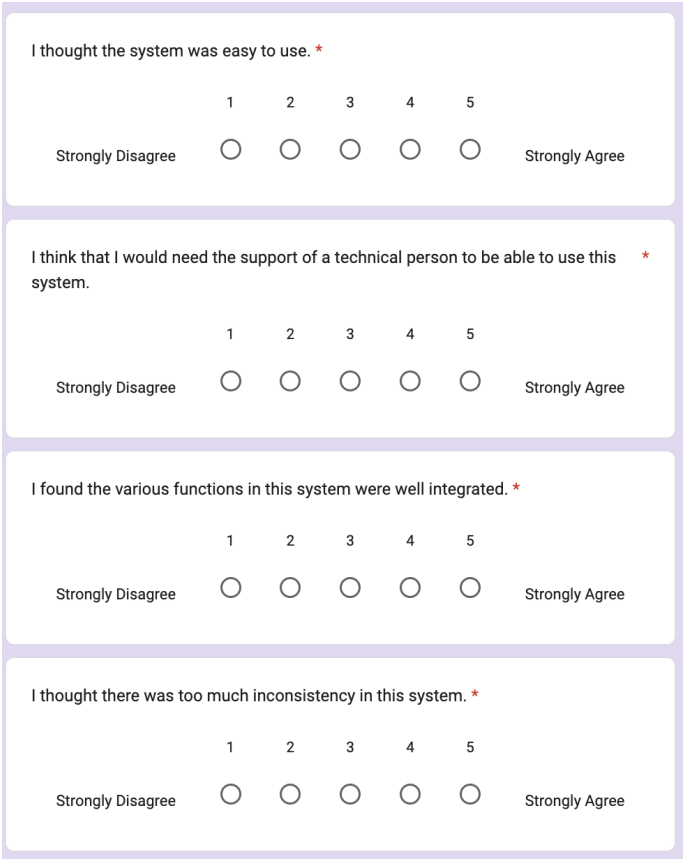


(d) SUS Questionnaire - Sociodemographic Data

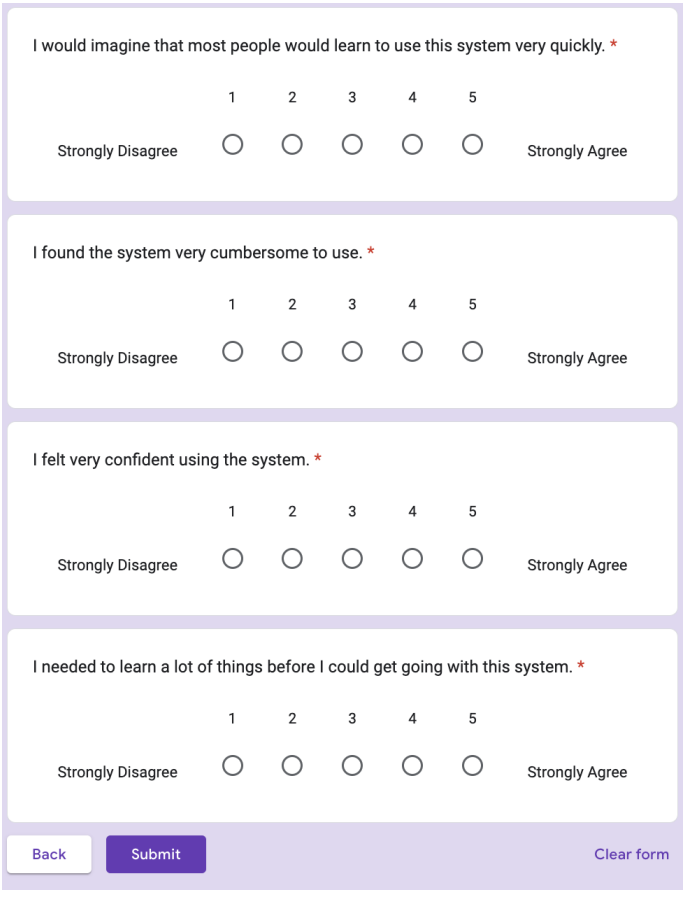


(e) SUS Questionnaire - SUS Questions 1-2

Fig. 57: System Usability Scale Questionnaire



(f) SUS Questionnaire - SUS Questions 3-6

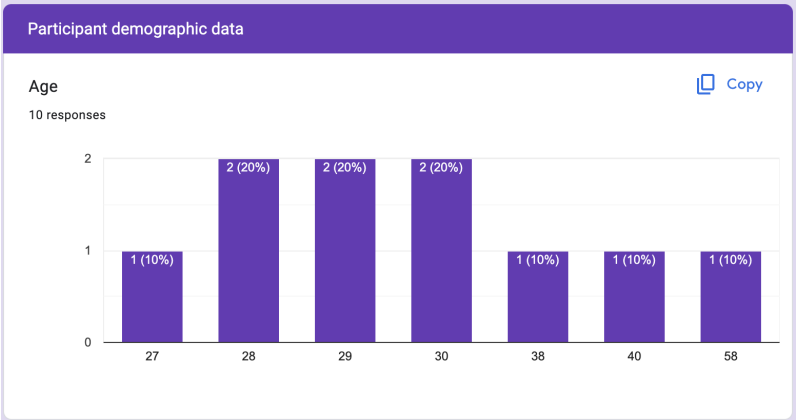


(g) SUS Questionnaire - SUS Questions 7-10

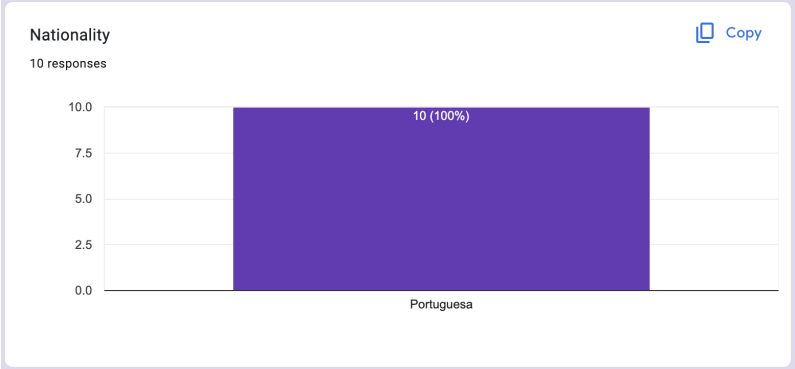
Fig. 57: System Usability Scale Questionnaire

C System Usability Scale Questionnaire Answers

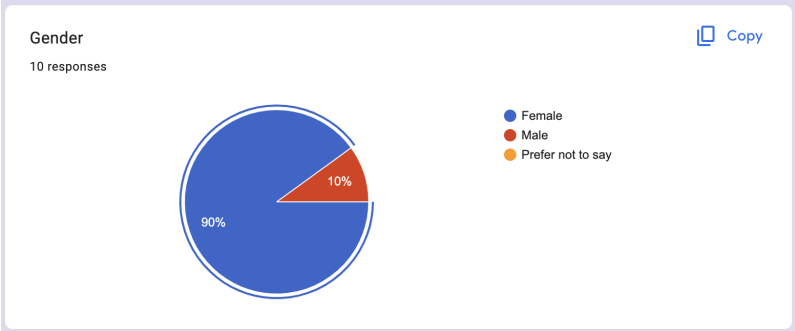
Below is a collection of images showcasing the SUS questionnaire answers collected from the participants. These images offer a glimpse into all the SUS questionnaire answers and their statistics.



(a) SUS Questionnaire - Age

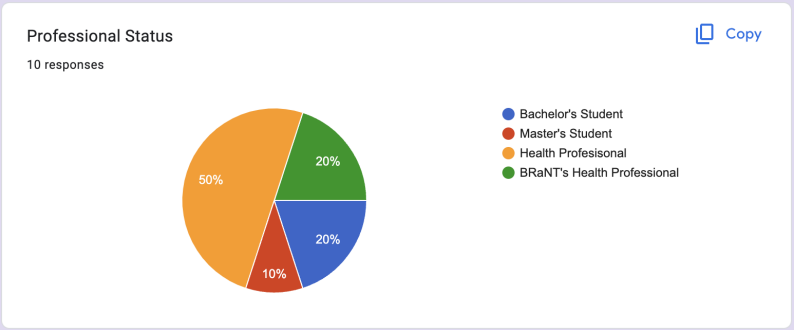


(b) SUS Questionnaire - Nationality

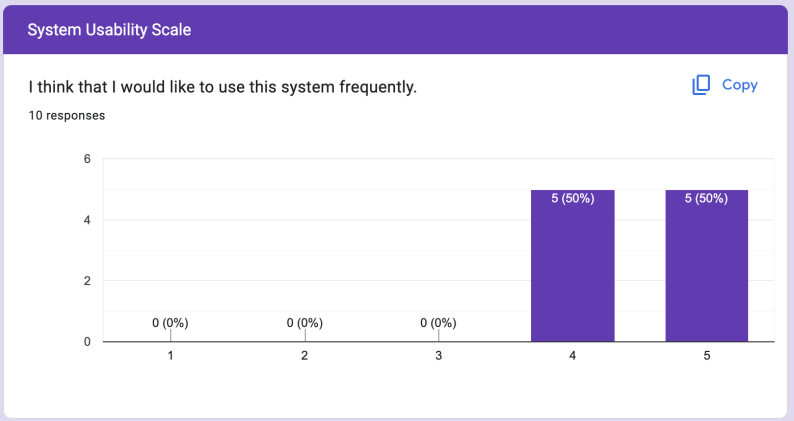


(c) SUS Questionnaire - Gender

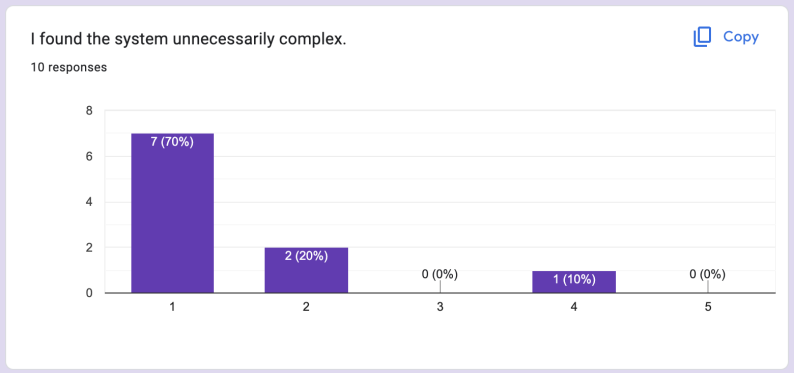
Fig. 58: System Usability Scale Questionnaire Answers



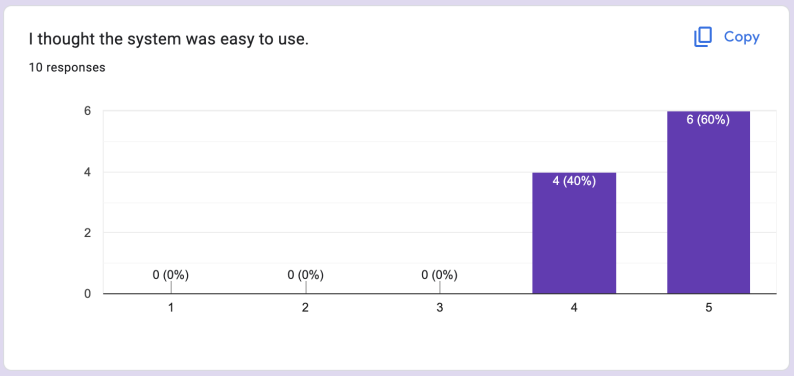
(d) SUS Questionnaire - Professional Status



(e) SUS Questionnaire - Question 1

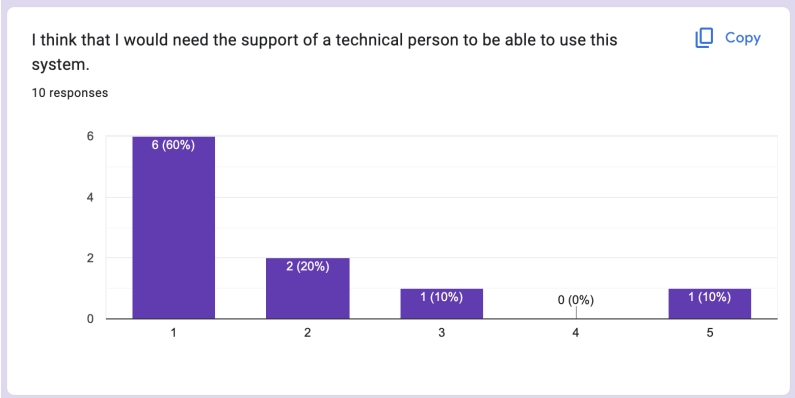


(f) SUS Questionnaire - Question 2

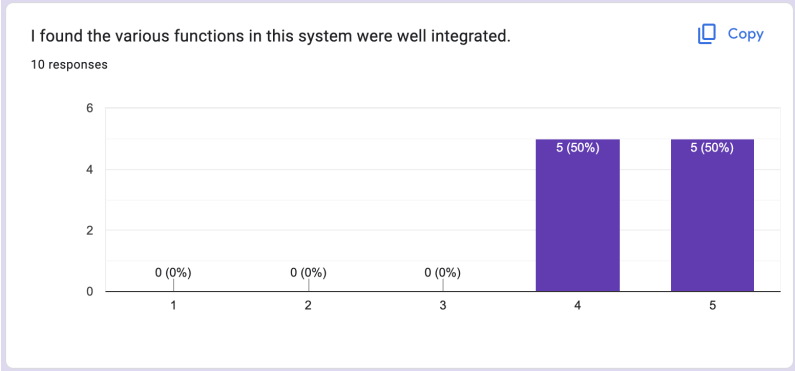


(g) SUS Questionnaire - Question 3

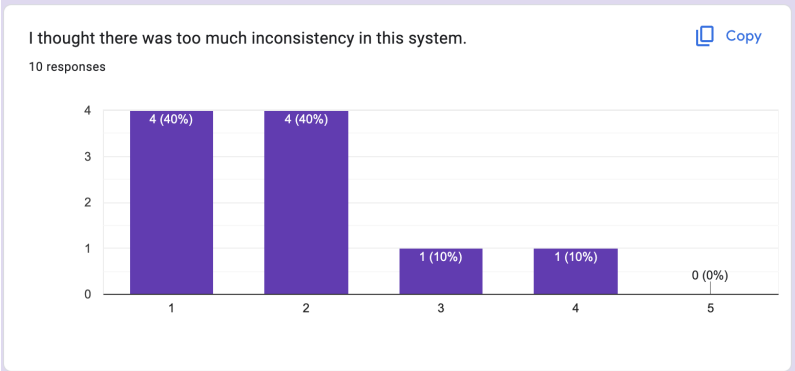
Fig. 58: System Usability Scale Questionnaire Answers



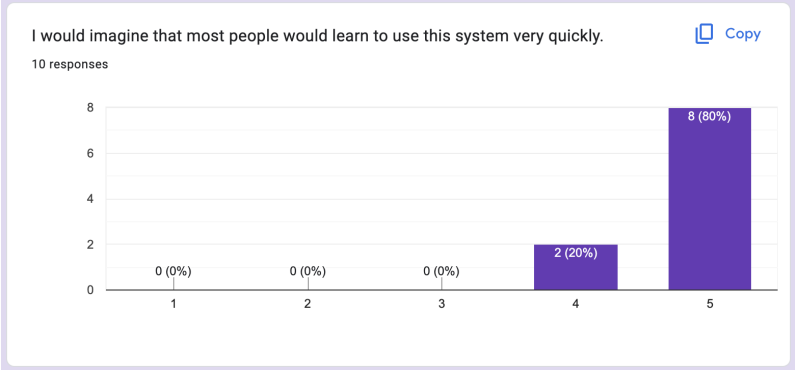
(h) SUS Questionnaire - Question 4



(i) SUS Questionnaire - Question 5

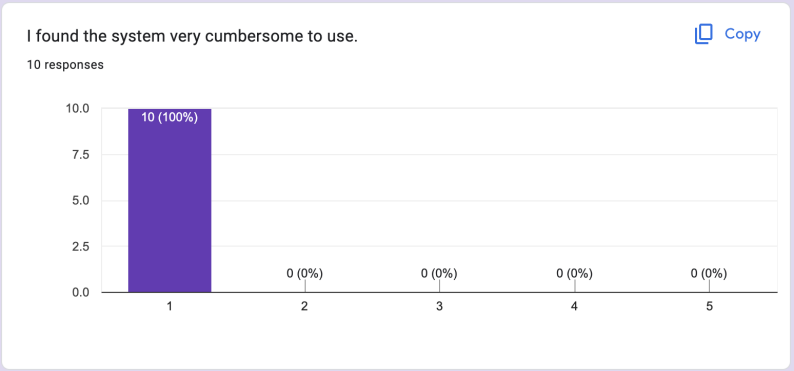


(j) SUS Questionnaire - Question 6

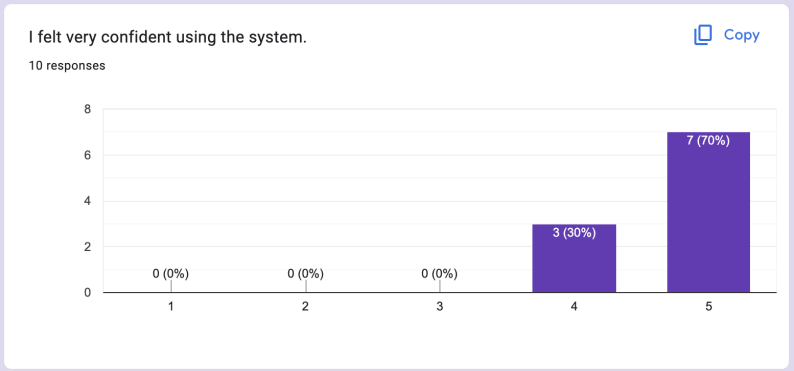


(k) SUS Questionnaire - Question 7

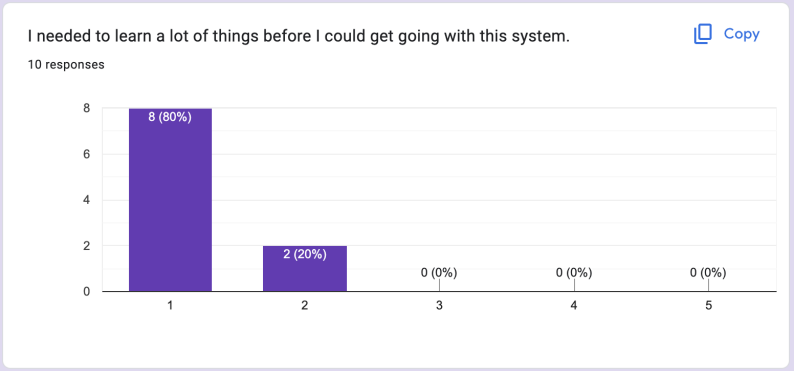
Fig. 58: System Usability Scale Questionnaire Answers



(l) SUS Questionnaire - Question 8



(m) SUS Questionnaire - Question 9



(n) SUS Questionnaire - Question 10

Fig. 58: System Usability Scale Questionnaire Answers

D NexusBRaNT's Full Transaction Description Table

The entire Transaction Description Table presented here is divided into 2 parts for legibility reasons. In reality, these 2 parts represent only one complete table.

ID	Process	Name	Origin Task(s)	Waits for task(s)	Target Task(s)	Conditions/Rules	Time Constraints
Patient Management (5 tasks)							
T1	**Patient Management**	Patient Registration					
T2	**Patient Management**	View patients associated with HP					
T3	**Patient Management**	Sociodemographic Data Update	T1 - Patient Registration				
T4	**Patient Management**	View the patient's sociodemographic data					
T5	**Patient Management**	Editing of Patient association to the HP					
User Management (5 tasks)							
T11	**User Management**	Health Professional Details Registration					
T12	**User Management**	Health Professional Details Edition	T11 - Health Professional Details Registration				
T13	**User Management**	Consult user information					
T14	**User Management**	Administrator Details Registration					
T15	**User Management**	Administrator Details Edition	T14 - Administrator Details Registration				
Clinical Information Management (3 tasks)							
T21	**Clinical Information Management**	Clinical Information Registration					
T22	**Clinical Information Management**	Clinical Information Edition	T21 - Clinical Information Registration				
T23	**Clinical Information Management**	View a patient's Clinical Information					
Neuropsychological Assessment Session (27 tasks)							
T31	**Neuropsychological Assessment Session**	Neuropsychological Assessment Session Creation					
T32	**Neuropsychological Assessment Session**	Neuropsychological Assessment Session Edition	T31 - Neuropsychological Assessment Session Creation				
T33	**Neuropsychological Assessment Session**	Check patient's neuropsychological assessment sessions					
T34	**Neuropsychological Assessment Session**	Cognitive Profile Calculation	T31 - Neuropsychological Assessment Session Creation			Must at least register one assessment instrument that alters the cognitive profile, else it will remain at zero	
T35	**Neuropsychological Assessment Session**	Clinical Interpretation Calculation	T31 - Neuropsychological Assessment Session Creation			Must at least register one assessment instrument that alters the clinical interpretation, else it will remain at zero	
T36	**Neuropsychological Assessment Session**	BDI-II Registration	T31 - Neuropsychological Assessment Session Creation				
T37	**Neuropsychological Assessment Session**	CDR Registration	T31 - Neuropsychological Assessment Session Creation				
T38	**Neuropsychological Assessment Session**	FCSRT Registration	T31 - Neuropsychological Assessment Session Creation				

ID	Process	Name	Origin Task(s)	Waits for task(s)	Target Task(s)	Conditions/Rules	Time Constraints
T39	**Neuropsychological Assessment Session**	Rey's Complex Figure Registration	T31 - Neuropsychological Assessment Session Creation				
T40	**Neuropsychological Assessment Session**	Semantic and Phonemic Verbal Fluency Registration	T31 - Neuropsychological Assessment Session Creation				
T41	**Neuropsychological Assessment Session**	GATSB Registration	T31 - Neuropsychological Assessment Session Creation				
T42	**Neuropsychological Assessment Session**	GDS-15 Registration	T31 - Neuropsychological Assessment Session Creation				
T43	**Neuropsychological Assessment Session**	GDS-30 Registration	T31 - Neuropsychological Assessment Session Creation				
T44	**Neuropsychological Assessment Session**	IAFAI Registration	T31 - Neuropsychological Assessment Session Creation				
T45	**Neuropsychological Assessment Session**	MOCA Registration	T31 - Neuropsychological Assessment Session Creation				
T46	**Neuropsychological Assessment Session**	PRECiS Registration	T31 - Neuropsychological Assessment Session Creation				
T47	**Neuropsychological Assessment Session**	QOLIBRI Registration	T31 - Neuropsychological Assessment Session Creation				
T48	**Neuropsychological Assessment Session**	QSM Registration	T31 - Neuropsychological Assessment Session Creation				
T49	**Neuropsychological Assessment Session**	Rey15 Registration	T31 - Neuropsychological Assessment Session Creation				
T50	**Neuropsychological Assessment Session**	SIS Registration	T31 - Neuropsychological Assessment Session Creation				
T51	**Neuropsychological Assessment Session**	Toulouse-Piéron Registration	T31 - Neuropsychological Assessment Session Creation				
T52	**Neuropsychological Assessment Session**	WAIS-III Registration	T31 - Neuropsychological Assessment Session Creation				
T53	**Neuropsychological Assessment Session**	WHOQOL-BREF Registration	T31 - Neuropsychological Assessment Session Creation				
T54	**Neuropsychological Assessment Session**	WHOQOL-OLD Registration	T31 - Neuropsychological Assessment Session Creation				
T55	**Neuropsychological Assessment Session**	WMS Registration	T31 - Neuropsychological Assessment Session Creation				

ID	Process	Name	Origin Task(s)	Waits for task(s)	Target Task(s)	Conditions/Rules	Time Constraints
T56	**Neuropsychological Assessment Session**	View the patient's cognitive profile.	T31 - Neuropsychological Assessment Session Creation				
T57	**Neuropsychological Assessment Session**	View the patient's clinical interpretation	T31 - Neuropsychological Assessment Session Creation				
Cognitive Training Programs (3 tasks)							
T61	**Cognitive Training Programs**	Cognitive Training Program Creation					
T62	**Cognitive Training Programs**	Cognitive Training Program Edition	T61 - Cognitive Training Program Creation				
T63	**Cognitive Training Programs**	Cognitive Training Program Consultation					
Cognitive Clinical Interpretation Management (4 tasks)							
T71	**Cognitive Clinical Interpretation Management**	Cognitive Normative Data Interpretation Creation					
T72	**Cognitive Clinical Interpretation Management**	Cognitive Normative Data Interpretation Edition	T71 - Cognitive Normative Data Interpretation Creation				
T73	**Cognitive Clinical Interpretation Management**	Scalar Score Interpretation Creation					
T74	**Cognitive Clinical Interpretation Management**	Scalar Score Interpretation Edition	T73 - Scalar Score Interpretation Creation				

ID	Source	Section	Process	Name	Task Kind	Executing Function	Description
Patient Management (5 tasks)							
T1	Reqs12/2022	RF 7 RF 7.1	**Patient Management**	Patient Registration	O	Health Professional	RF7 - The system should allow HPs to register a new patient with the name, email, address, profession, professional status, date of birth, gender, education, laterality, household, telephone contact and marital status. RF7.1 - The system should identify all fields as mandatory. RF7.4 - The system should allow saving previous sociodemographic information of the patient.
T2	Reqs12/2022	RF 8 RF 8.1	**Patient Management**	View patients associated with HP	D	Health Professional	RF8 - The system should allow HPs to view patients associated only with themselves. RF8.1 - The system should allow filtering/sorting of patients by name, age and profession.
T3	Reqs12/2022	RF 7.3	**Patient Management**	Sociodemographic Data Update	O	Health Professional	RF7.3 - The system should allow HPs to update patients' sociodemographic data.
T4	Reqs12/2022	RF 7.2 UC 3	**Patient Management**	View the patient's sociodemographic data	D	Health Professional	RF7.2 - The system must allow HPs to view the patient's sociodemographic data. UC3. Query sociodemographic and clinical information of associated patients.
T5	Reqs	RF5	**Patient Management**	Editing of Patient association to the HP	O	Administrator	RF5. The system shall allow the administrator to associate and disassociate a patient to the HP.
User Management (5 tasks)							
T11	Reqs12/2022	RF 2 RF 2.1 RF 2.1.1	**User Management**	Health Professional Details Registration	O	Administrator	RF2 - The system administrator may register new HPs. RF2.1 - The system should allow registering a new HP with the name, email, date of registration, account type (research HP or clinical HP), password, professional cell, and telephone contact. RF2.1.1 - The system shall identify as mandatory fields the following data: name, email, date of registration, password, professional cell, and telephone contact.
T12	UC	UC 19	**User Management**	Health Professional Details Edition	O	Administrator	UC19. Edit user information
T13	Reqs12/2022	RF 3 UC 18	**User Management**	Consult user information	D	Administrator	RF3. The system administrator may view all registered users and only the ID, name and email of the patients associated to each user. UC18. Viewing users' information.
T14	Protótipo Filipa		**User Management**	Administrator Details Registration	O	Administrator	
T15	Protótipo Filipa		**User Management**	Administrator Details Edition	O	Administrator	
Clinical Information Management (3 tasks)							
T21	Reqs12/2022	RF9.1	**Clinical Information Management**	Clinical Information Registration	O	Health Professional	RF9.1 - The system shall add clinical information of the patient with the type of information (diagnosis/clinical history, pharmacological therapy, complementary tests and clinical records of sessions), date of registration and description of the information.
T22	Reqs12/2022	RF 9.3	**Clinical Information Management**	Clinical Information Edition	O	Health Professional	RF9.3 - The system should allow HPs to edit the patient's clinical information.
T23	Reqs12/2022	RF 9 RF 9.4 RF 9.5 UC 3	**Clinical Information Management**	View a patient's Clinical Information	D	Health Professional	RF9 - The system must allow HPs to view the patient's clinical information. RF9.4 - The system should allow filtering/sorting of clinical information by date of record, type of information, and description. RF9.5 - The system must allow viewing of detailed clinical information. UC3. Viewing socio-demographic and clinical information of associated patients.
Neuropsychological Assessment Session (27 tasks)							

ID	Source	Section	Process	Name	Task Kind	Executing Function	Description
T31	Reqs12/2022	RF 14 RF 14.1 RF 14.2 RF 14.3	**Neuropsychological Assessment Session**	Neuropsychological Assessment Session Creation	O	Health Professional	RF14 - The system should allow the creation of a neuropsychological assessment session associated with a patient. RF14.1 - The system should allow adding more than one assessment instrument in the assessment session. RF14.2 - The HP shall be able to enter data according to the results obtained by the patient in each selected assessment instrument. RF14.3 - The system should allow insertion of clinical interpretation into the assessment instruments when it contains.
T32	Reqs12/2022	RF 11.2	**Neuropsychological Assessment Session**	Neuropsychological Assessment Session Edition	O	Health Professional	RF11.2 - The system should allow the HP to edit the neuropsychological assessment (session number, results, clinical interpretation and observations).
T33	Reqs12/2022	RF 11 RF 11.1 RF 11.5 UC 10	**Neuropsychological Assessment Session**	Check patient's neuropsychological assessment sessions	D	Health Professional	RF11 - The system shall display the patient's neuropsychological assessment sessions with the assessment session number, date, HP, and assessment instrument. RF11.1 - The system shall allow viewing detailed information of each neuropsychological assessment. RF11.5 - The system shall allow filtering/ordering of neuropsychological assessments by session number, date, assessment instrument, and HP. UC10. View neuropsychological assessments of the patient.
T34	Reqs12/2022	RF 10.1	**Neuropsychological Assessment Session**	Cognitive Profile Calculation	I	Health Professional	RF10.1 - The system should allow the calculation of the patient's cognitive profile through the last neuropsychological evaluation session.
T35	Reqs12/2022	RF 10.2	**Neuropsychological Assessment Session**	Clinical Interpretation Calculation	I	Health Professional	RF10.2 - The system should allow to calculate the clinical interpretation of the patient through the last neuropsychological assessment session.
T36	BdNexus		**Neuropsychological Assessment Session**	BDI-II Registration	O	Health Professional	
T37	BdNexus		**Neuropsychological Assessment Session**	CDR Registration	O	Health Professional	
T38	BdNexus		**Neuropsychological Assessment Session**	FCSRT Registration	O	Health Professional	
T39	BdNexus		**Neuropsychological Assessment Session**	Rey's Complex Figure Registration	O	Health Professional	
T40	BdNexus		**Neuropsychological Assessment Session**	Semantic and Phonemic Verbal Fluency Registration	O	Health Professional	
T41	BdNexus		**Neuropsychological Assessment Session**	GATSB Registration	O	Health Professional	
T42	BdNexus		**Neuropsychological Assessment Session**	GDS-15 Registration	O	Health Professional	
T43	BdNexus		**Neuropsychological Assessment Session**	GDS-30 Registration	O	Health Professional	
T44	BdNexus		**Neuropsychological Assessment Session**	IAFAI Registration	O	Health Professional	
T45	BdNexus		**Neuropsychological Assessment Session**	MOCA Registration	O	Health Professional	

ID	Source	Section	Process	Name	Task Kind	Executing Function	Description
T46	BdNexus		**Neuropsychological Assessment Session**	PRECiS Registration	O	Health Professional	
T47	BdNexus		**Neuropsychological Assessment Session**	QOLIBRI Registration	O	Health Professional	
T48	BdNexus		**Neuropsychological Assessment Session**	QSM Registration	O	Health Professional	
T49	BdNexus		**Neuropsychological Assessment Session**	Rey15 Registration	O	Health Professional	
T50	BdNexus		**Neuropsychological Assessment Session**	SIS Registration	O	Health Professional	
T51	BdNexus		**Neuropsychological Assessment Session**	Toulouse-Piéron Registration	O	Health Professional	
T52	BdNexus		**Neuropsychological Assessment Session**	WAIS-III Registration	O	Health Professional	
T53	BdNexus		**Neuropsychological Assessment Session**	WHOQOL-BREF Registration	O	Health Professional	
T54	BdNexus		**Neuropsychological Assessment Session**	WHOQOL-OLD Registration	O	Health Professional	
T55	BdNexus		**Neuropsychological Assessment Session**	WMS Registration	O	Health Professional	
T56	Reqs12/2022	RF 10	**Neuropsychological Assessment Session**	View the patient's cognitive profile.	D	Health Professional	RF10 - The system must allow HPs to view the cognitive profile of their patients. UC6. View patient's cognitive profile.
T57	Protótipo Filipa		**Neuropsychological Assessment Session**	View the patient's clinical interpretation	D	Health Professional	
Cognitive Training Programs (3 tasks)							
T61	Reqs12/2022	RF 15 RF 15.1 RF 15.1.1	**Cognitive Training Programs**	Cognitive Training Program Creation	O	Health Professional	RF15 - The system should allow the creation of a training program associated with a patient. RF15.1 - The system should allow adding to the training program the data referred to in requirement RF12.1. [RF12.1 - The system shall allow accessing the detailed information of the patient's training programs (total number of sessions, date of first session, number of times per week, duration of each session, adaptation challenge and state)]. RF15.1.1 - The system should identify all fields as mandatory.
T62	Reqs12/2022	RF 12.3	**Cognitive Training Programs**	Cognitive Training Program Edition	O	Health Professional	RF12.3 - The system should allow HP to edit the training programme with the data referred to in RF12.1. [RF12.1 - The system should allow access to the detailed information of the patient's training programmes (total number of sessions, date of first session, number of times per week, duration of each session, adaptation challenge and state)].

ID	Source	Section	Process	Name	Task Kind	Executing Function	Description
T63	Reqs12/2022	RF 12 RF 12.1 RF 12.4 UC 14	**Cognitive Training Programs**	Cognitive Training Program Consultation	D	Health Professional	RF12 - The system should display patient's training programs. RF12.1 - The system should allow access to detailed information of patient's training programs (total number of sessions, date of first session, number of times per week, duration of each session, adaptation challenge and status). RF12.4 - The system must allow filtering/ordering of training programs by training program number, start date, total sessions, PS and state. UC14. Query a patient's cognitive training program.
Cognitive Clinical Interpretation Management (4 tasks)							
T71	BdNexus		**Cognitive Clinical Interpretation Management**	Cognitive Normative Data Interpretation Creation	O	Administrator	
T72	BdNexus		**Cognitive Clinical Interpretation Management**	Cognitive Normative Data Interpretation Edition	O	Administrator	
T73	BdNexus		**Cognitive Clinical Interpretation Management**	Scalar Score Interpretation Creation	O	Administrator	
T74	BdNexus		**Cognitive Clinical Interpretation Management**	Scalar Score Interpretation Edition	O	Administrator	

E NexusBRaNT's Full Fact Description Table

The entire Fact Description Table presented here is divided into 2 parts for legibility reasons. In reality, these 2 parts represent only one complete table.

Scope	Source	Concept	Attribute Name	Attribute's value type	Referenced Concept / Category Values	Description
Patient Intrinsic Concepts (5 attributes)						
Patient Intrinsic Concepts		**Patient**				RF7 - The system should allow HPs to register a new patient with the name, email, address, occupation, employment status, date of birth, gender, education, laterality, household, telephone contact and marital status. RF7.1 - The system should identify all fields as mandatory.
Patient Intrinsic Concepts	Reqs12/2022	**Patient**	Full Name	text		
Patient Intrinsic Concepts	Reqs12/2022	**Patient**	Email	text		
Patient Intrinsic Concepts	Reqs12/2022	**Patient**	Address	text		
Patient Intrinsic Concepts	Reqs12/2022	**Patient**	Phone Contact	text		
Patient Intrinsic Concepts	Reqs12/2022	**Patient**	Health Professional	reference	Health Professional	
Patient Related Concepts (11 attributes)						
Patient Related Concepts		**Sociodemographic Data**				RF7 - The system should allow HPs to register a new patient with name, email, address, profession, professional status, date of birth, gender, education, laterality, household, telephone contact and marital status. RF7.2 - The system should allow HPs to view sociodemographic data of the patient. RF7.3 - The system should allow the HP to update the patient's sociodemographic data. RF7.4 - The system should allow storing of the patient's previous sociodemographic information.
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Education Level	category	High School	Using the interval of years, as exemplified in https://en.wikipedia.org/wiki/Educational_stage
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Date of Birth	date		
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Sex	category	Male	
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Laterality	category	Right	
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Professional Situation	category	Employed	
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Profession	text		
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Household	category	Mother	To be able to specify the patient's household. As it is only enough to state that the household is e.g. made up of mother and father, it is not necessary to know their names, this multiple_values attribute is sufficient.
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Civil Status	category	Single	
Patient Related Concepts		**Clinical Information**				RF9 - O sistema deverá permitir aos PS visualizar a informação clínica do paciente. RF9.1 – O sistema deverá adicionar informação clínica do paciente com o tipo de informação (diagnóstico principal, terapêutica farmacológica, exames complementares, histórico clínico e data de registo da informação) e descrição da informação.
Patient Related Concepts	Reqs12/2022	**Clinical Information**	Patient	reference	Patient	
Patient Related Concepts	Reqs12/2022	**Clinical Information**	Type	category	Diagnosis/Clinical History	

Scope	Source	Concept	Attribute Name	Attribute's value type	Referenced Concept / Category Values	Description
Patient Related Concepts	Reqs12/2022	**Clinical Information**	Description	text		
	Health Professional Intrinsic Concepts (3 attributes)					
Health Professional Intrinsic Concepts		**Health Professional**				RF2 - The system administrator may register new HPs. RF2.1 - The system should allow registering a new HP with the name, email, date of registration, account type (research HP or clinical HP), password, professional cell, and telephone contact. RF2.1.1 - The system shall identify as mandatory fields the following data: name, email, date of registration, password, professional cell, and telephone contact.
Health Professional Intrinsic Concepts	Reqs12/2022	**Health Professional**	Phone Contact	text		
Health Professional Intrinsic Concepts	Reqs12/2022	**Health Professional**	Professional Cell	doc & text		
Health Professional Intrinsic Concepts	Reqs12/2022	**Health Professional**	Joining Date	date		
	Administrator Intrinsic Concepts (1 attributes)					
Administrator Intrinsic Concepts		**Administrator**				
Administrator Intrinsic Concepts	NexusBRaNT Prototype	**Administrator**	Phone Contact	text		
	Cognitive Training Program Instrinsic Concepts (9 attributes)					
Cognitive Training Program Instrinsic Concepts		**Cognitive Training Program**				RF15 - The system should allow the creation of a training program associated with a patient. RF15.1 - The system should allow adding to the training program the data referred to in requirement RF12.1. [RF12.1 - The system should allow accessing the detailed information of the patient's training programs (total number of sessions, date of first session, number of times per week, duration of each session, adaptation challenge and state)]. RF15.1.1 - The system should identify all fields as mandatory.
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Patient	reference	Patient	The patient who will undertake the cognitive training program
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Start Date	date		The date on which the program will start
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Number of Training Sessions	number		Number of training sessions the patient must do for this program
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Weekly Recurrence	number		Number of days per week that the patient must execute this training program
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Days of the Week	category	Monday	Which days of the week this program will take place
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Training Session Duration	number		Duration of each session of the training program
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	State	category	Not Started	Current state of the program
Cognitive Training Program Instrinsic Concepts	BdNexus	**Cognitive Training Program**	Minimum Adaptation Challenge	number		The minimum adaptation challenge that this program must meet (0-100)
Cognitive Training Program Instrinsic Concepts	BdNexus	**Cognitive Training Program**	Maximum Adaptation Challenge	number		The maximum adaptation challenge that this program must meet (0-100)

Scope	Source	Concept	Attribute Name	Attribute's value type	Referenced Concept / Category Values	Description
Neuropsychological Assessment Session Intrinsic Concepts (4 attributes)						
Neuropsychological Assessment Session Intrinsic Concepts		**Neuropsychological Assessment Session**				RF11 - The system shall display the patient's neuropsychological assessment sessions with the assessment session number, date, HP, and assessment instrument. RF11.2 - The system shall allow the HP to edit the neuropsychological assessment (session number, results, clinical interpretation and notes). DB - 'Sessions_Assessment
Neuropsychological Assessment Session Intrinsic Concepts	BdNexus	**Neuropsychological Assessment Session**	Patient	reference	Patient	
Neuropsychological Assessment Session Intrinsic Concepts	BdNexus	**Neuropsychological Assessment Session**	Health Professional	reference	Health Professional	
Neuropsychological Assessment Session Intrinsic Concepts	Regs12/2022	**Neuropsychological Assessment Session**	Session Number	number		
Neuropsychological Assessment Session Intrinsic Concepts	Regs12/2022	**Neuropsychological Assessment Session**	Execution Date	date		
Neuropsychological Assessment Session Related Concepts (160 attributes)						
Neuropsychological Assessment Session Related Concepts		**Clinical Interpretation**				
Neuropsychological Assessment Session Related Concepts	BdNexus	**Clinical Interpretation**	General Cognition	number		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Clinical Interpretation**	Memory	number		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Clinical Interpretation**	Executive Functions	number		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Clinical Interpretation**	Language	number		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Clinical Interpretation**	Attention	number		
Neuropsychological Assessment Session Related Concepts		**Cognitive Profile**				
Neuropsychological Assessment Session Related Concepts	BdNexus	**Cognitive Profile**	General Cognition	number		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Cognitive Profile**	Memory	number		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Cognitive Profile**	Executive Functions	number		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Cognitive Profile**	Language	number		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Cognitive Profile**	Attention	number		
Neuropsychological Assessment Session Related Concepts		**Cognitive Normative Data Interpretation**				
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Cognitive Normative Data Interpretation**	Interpretation Level	text		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Cognitive Normative Data Interpretation**	Standard Deviation from the Average	text		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Cognitive Normative Data Interpretation**	Scale	number		
Neuropsychological Assessment Session Related Concepts		**Scalar Score Interpretation**				

Scope	Source	Concept	Attribute Name	Attribute's value type	Referenced Concept / Category Values	Description
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Scalar Score Interpretation**	Interpretation Level	text		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Scalar Score Interpretation**	Scalar Score	text		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Scalar Score Interpretation**	Scale	number		
Neuropsychological Assessment Instruments						
Neuropsychological Assessment Session Related Concepts		**MOCA**				Storage of the results of a MOCA test, inserted in a certain cognitive assessment. DB Table - MOCA
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Visuospatial / Executive	number		Min: 0 Max: 5
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Nomination	number		Min: 0 Max: 3
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Attention	number		Min: 0 Max: 6
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Language	number		Min: 0 Max: 3
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Abstraction	number		Min: 0 Max: 2
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Deferred Recall	number		Min: 0 Max: 5
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Orientation	number		Min: 0 Max: 6
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Total	number		Min: 0 Max: 30
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**MOCA**	Total - Average	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**MOCA**	Total - Standard Deviation	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**MOCA**	Total - Clinical Interpretation	reference	Cognitive Normative Data Interpretation	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**MOCA**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**WAIS-III**				Storage of the results of a WAIS-III test, inserted in a certain cognitive assessment. DB Table - WAIS-III
Neuropsychological Assessment Session Related Concepts	BdNexus	**WAIS-III**	Symbol Search	number		Min: 0 Max: 60
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Symbol Search - Scalar Score	number		Min: 1 Max: 19
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Symbol Search - Clinical Interpretation	reference	Scalar Score Interpretation	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Code	number		Min: 0 Max: 133

Scope	Source	Concept	Attribute Name	Attribute's value type	Referenced Concept / Category Values	Description
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Code - Scalar Score	number		Min: 1 Max: 19
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Code - Clinical Interpretation	reference	Scalar Score Interpretation	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Vocabulary	number		Min: 0 Max: 66
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Vocabulary - Scalar Score	number		Min: 1 Max: 19
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Vocabulary - Clinical Interpreation	reference	Scalar Score Interpretation	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**GDS-15**				Storage of the results of a GDS-15 test, inserted in a certain cognitive assessment. DB Table - GDS-15
Neuropsychological Assessment Session Related Concepts	BdNexus	**GDS-15**	GDS-15	number		Min: 0 Max: 15
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**GDS-15**	Clinical Interpretation	category	Normal	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**GDS-15**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**GDS-30**				Storage of the results of a GDS-30 test, inserted in a certain cognitive assessment. DB Table - GDS-30
Neuropsychological Assessment Session Related Concepts	BdNexus	**GDS-30**	GDS-30	number		Min: 0 Max: 30
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**GDS-30**	Clinical Interpretation	category	Normal Symptomatology	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**GDS-30**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**GATSB**				Storage of the results of a GATSB test, inserted in a certain cognitive assessment. DB Table - GATSB
Neuropsychological Assessment Session Related Concepts	BdNexus	**GATSB**	GATSB	number		Min: 0 Max: 58
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**GATSB**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**Semantic and Phonemic Verbal Fluency**				Storage of the results of a Verbal and Semantic Fluency test, inserted in a certain cognitive assessment. DB Table - PhonemicSemanticVerbalFluence
Neuropsychological Assessment Session Related Concepts	BdNexus	**Semantic and Phonemic Verbal Fluency**	Semantic Verbal Fluency - Animals	number		No max and min
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Semantic and Phonemic Verbal Fluency**	Semantic Verbal Fluency - Animals - Scalar Score	number		Min: 1 Max: 19
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Semantic and Phonemic Verbal Fluency**	Semantic Verbal Fluency - Animals - Clinical Interpretation	reference	Scalar Score Interpretation	

Scope	Source	Concept	Attribute Name	Attribute's value type	Referenced Concept / Category Values	Description
Neuropsychological Assessment Session Related Concepts	BdNexus	**Semantic and Phonemic Verbal Fluency**	Semantic Verbal Fluency - Professions (FVSP)	number		No max and min
Neuropsychological Assessment Session Related Concepts	BdNexus	**Semantic and Phonemic Verbal Fluency**	Phonemic Verbal Fluency M (FVFM)	number		No max and min
Neuropsychological Assessment Session Related Concepts	BdNexus	**Semantic and Phonemic Verbal Fluency**	Phonemic Verbal Fluency R (FVFR)	number		No max and min
Neuropsychological Assessment Session Related Concepts	BdNexus	**Semantic and Phonemic Verbal Fluency**	Phonemic Verbal Fluency P (FVFP)	number		No max and min
Neuropsychological Assessment Session Related Concepts	BdNexus	**Semantic and Phonemic Verbal Fluency**	Phonemic Verbal Fluency - Total	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Semantic and Phonemic Verbal Fluency**	Phonemic Verbal Fluency - Scalar Score	number		Min: 1 Max: 19
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Semantic and Phonemic Verbal Fluency**	Phonemic Verbal Fluency - Clinical Interpretation	reference	Scalar Score Interpretation	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Semantic and Phonemic Verbal Fluency**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**FCSRT**				Storage of the results of a FCSTR test, inserted in a certain cognitive assessment. DB Table - FCSTR
Neuropsychological Assessment Session Related Concepts	BdNexus	**FCSRT**	Immediate Free Recall	number		Min: 0 Máx: 48
Neuropsychological Assessment Session Related Concepts	BdNexus	**FCSRT**	Immediate Guided Recall	number		Min: 0 Máx: 48
Neuropsychological Assessment Session Related Concepts	BdNexus	**FCSRT**	Total Immediate Recall (Free + Guided)	number		Min: 0 Máx: 48
Neuropsychological Assessment Session Related Concepts	BdNexus	**FCSRT**	Deferred Free Recall	number		Min: 0 Máx: 16
Neuropsychological Assessment Session Related Concepts	BdNexus	**FCSRT**	Deferred Guided Recall	number		Min: 0 Máx: 16
Neuropsychological Assessment Session Related Concepts	BdNexus	**FCSRT**	Total Deferred Recall (Free + Guided)	number		Min: 0 Máx: 48
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**FCSRT**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**IAFAI**				Storage of the results of an IAFAI test, inserted in a certain cognitive assessment. DB Table - IAFAI
Neuropsychological Assessment Session Related Concepts	BdNexus	**IAFAI**	ABVD	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**IAFAI**	AIVD-F	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**IAFAI**	AIVD-A	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**IAFAI**	Total	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**IAFAI**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**PRECiS**				Storage of the results of a PRECiS test, inserted in a certain cognitive assessment. DB Table - PRECiS

Scope	Source	Concept	Attribute Name	Attribute's value type	Referenced Concept / Category Values	Description
Neuropsychological Assessment Session Related Concepts	BdNexus	**PRECiS**	Total Score	number		Min: 0 Max: 128
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**PRECiS**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**QSM**				Storage of the results of a QSM test, inserted in a certain cognitive assessment. DB Table - QSM
Neuropsychological Assessment Session Related Concepts	BdNexus	**QSM**	Total Score	number		Min: 0 Max: 21
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**QSM**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**Toulouse-Piéron**				Storage of the results of a Toulouse-Pieron test, inserted in a certain cognitive assessment. DB Table - ToulousePieronCancellationTest
Neuropsychological Assessment Session Related Concepts	BdNexus	**Toulouse-Piéron**	Realization Capacity	number		Min: -37.5 Max: 37.5
Neuropsychological Assessment Session Related Concepts	BdNexus	**Toulouse-Piéron**	Dispersion Index	number		DI = (omissions + errors)/CR × 100 Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Dispersion Index - Average	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Dispersion Index - Standard Deviation	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Dispersion Index - Clinical Interpretation	reference	Cognitive Normative Data Interpretation	
Neuropsychological Assessment Session Related Concepts	BdNexus	**Toulouse-Piéron**	Work Efficiency	number		WR = RC - (omissions + errors) Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Work Efficiency - Average	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Work Efficiency - Standard Deviation	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Work Efficiency - Clinical Interpretation	reference	Cognitive Normative Data Interpretation	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**SIS**				Storage of the results of a SIS test, inserted in a certain cognitive assessment. DB Table - SIS
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Strength	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Hand Strength	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Mobility	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Daily Life Activities	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Emotional	number		Min: 0 Max: 100

Scope	Source	Concept	Attribute Name	Attribute's value type	Referenced Concept / Category Values	Description
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Memory	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Communication	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Social Engagement	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Stroke Recovery	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**SIS**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**WMS**				Storage of the results of a WMS test, inserted in a certain cognitive assessment. DB Table - WMS
Neuropsychological Assessment Session Related Concepts	BdNexus	**WMS**	Immediate Recall	number		Min: 0 Max: 104
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Immediate Recall - Scalar Score	number		Min: 1 Max: 19
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Immediate Recall - Clinical Interpretation	reference	Scalar Score Interpretation	
Neuropsychological Assessment Session Related Concepts	BdNexus	**WMS**	Long-term Recall	number		Min: 0 Max: 104
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Long-term Recall - Scalar Score	number		Min: 1 Max: 19
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Long-term Recall - Clinical Interpretation	reference	Scalar Score Interpretation	
Neuropsychological Assessment Session Related Concepts	BdNexus	**WMS**	Copy	number		Min: 0 Max: 104
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Copy - Scalar Score	number		Min: 1 Max: 19
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Copy - Clinical Interpretation	reference	Scalar Score Interpretation	
Neuropsychological Assessment Session Related Concepts	BdNexus	**WMS**	Discrimination	number		Min: 0 Max: 7
Neuropsychological Assessment Session Related Concepts	BdNexus	**WMS**	Recognition	number		Min: 0 Max: 48
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Recognition - Scalar Score	number		Min: 1 Max: 19
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Recognition - Clinical Interpretation	reference	Scalar Score Interpretation	
Neuropsychological Assessment Session Related Concepts	BdNexus	**WMS**	Retention Percentage	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Retention Percentage - Scalar Score	number		Min: 1 Max: 19
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Retention Percentage - Clinical Interpretation	reference	Scalar Score Interpretation	

Scope	Source	Concept	Attribute Name	Attribute's value type	Referenced Concept / Category Values	Description
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**WHOQOL-BREF**				Storage of the results of a WHOQOL-BREF test, inserted in a certain cognitive assessment. DB Table - WHOQOL-BREF-PT
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-BREF**	Domain 1	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-BREF**	Domain 2	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-BREF**	Domain 3	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-BREF**	Domain 4	number		Min: 0 Max: 100
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WHOQOL-BREF**	Observações	text		
Neuropsychological Assessment Session Related Concepts		**WHOQOL-OLD**				Storage of the results of a WHOQOL-OLD test, inserted in a certain cognitive assessment. DB Table - WHOQOL-OLD
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Sensory Functioning	number		Min: 0 Max: 20
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Autonomy	number		Min: 0 Max: 20
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Past / Present / Future Activities	number		Min: 0 Max: 20
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Social Engagement	number		Min: 0 Max: 20
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Death / Dying	number		Min: 0 Max: 20
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Intimacy	number		Min: 0 Max: 20
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Family and Family Life	number		Min: 0 Max: 20
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Total	number		Min: 0 Max: 140
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WHOQOL-OLD**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**Rey's Complex Figure**				Storage of the results of a Rey's Complex Figure test, inserted in a certain cognitive assessment. DB Table - FCREY
Neuropsychological Assessment Session Related Concepts	BdNexus	**Rey's Complex Figure**	Copy	number		Min: 0 Máx: 36
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Copy - Average	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Copy - Standard Deviation	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Copy - Clinical Interpretation	reference	Cognitive Normative Data Interpretation	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Immediate Memory	number		Min: 0 Máx: 36

Scope	Source	Concept	Attribute Name	Attribute's value type	Referenced Concept / Category Values	Description
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Immediate Memory - Average	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Immediate Memory - Standard Deviation	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Immediate Memory - Clinical Interpretation	reference	Cognitive Normative Data Interpretation	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Deferred Memory	number		Min: 0 Máx: 36
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Deferred Memory - Average	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Deferred Memory - Standard Deviation	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Deferred Memory - Clinical Interpretation	reference	Cognitive Normative Data Interpretation	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**Rey15**				Storage of the results of a Rey15 test, inserted in a certain cognitive assessment. Rey15 Table
Neuropsychological Assessment Session Related Concepts	BdNexus	**Rey15**	Immediate Recall Test	number		Min: 0 Máx: 15
Neuropsychological Assessment Session Related Concepts	BdNexus	**Rey15**	Combined Recognition Result	number		Min: 0 Máx: 30
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey15**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**BDI-II**				Storage of the results of a BDI-II test, inserted in a certain cognitive assessment. DB Table - BDI-II
Neuropsychological Assessment Session Related Concepts	BdNexus	**BDI-II**	BDI-II	number		Min: 0 Max: 63
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**BDI-II**	Clinical Interpretation	category	Minimal Depression	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**BDI-II**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**QOLIBRI**				Storage of the results of a QOLIBRI test, inserted in a certain cognitive assessment. QOLIBRI Table
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	Total	number		Range 0-100
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	Cognitive	number		Range 0-100
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	Self	number		Range 0-100
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	AVDs	number		Range 0-100
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	Social Relationships	number		Range 0-100

Scope	Source	Concept	Attribute Name	Attribute's value type	Referenced Concept / Category Values	Description
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	Emotions	number		Range 0-100
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	Physical Condition	number		Range 0-100
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**QOLIBRI**	Observations	text		
Neuropsychological Assessment Session Related Concepts		**CDR**				Storage of the results of a CDR test, inserted in a certain cognitive assessment. CDR Table
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Memory	number		Range 0-3
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Orientation	number		Range 0-3
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Judgement and Problem Solving	number		Range 0-3
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Activities in the Communities	number		Range 0-3
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Home and Hobbies	number		Range 0-3
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Personal Care	number		Range 0-3
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Total	number		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**CDR**	Clinical Interpretation	category	Suspect	
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**CDR**	Observations	text		

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
	Patient Intrinsic Concepts (5 attributes)						
Patient Intrinsic Concepts		**Patient**		Reqs12/2022	Patient Registration		
Patient Intrinsic Concepts	Reqs12/2022	**Patient**	Full Name	Reqs12/2022	Patient Registration		
Patient Intrinsic Concepts	Reqs12/2022	**Patient**	Email	Reqs12/2022	Patient Registration		
Patient Intrinsic Concepts	Reqs12/2022	**Patient**	Address	Reqs12/2022	Patient Registration		
Patient Intrinsic Concepts	Reqs12/2022	**Patient**	Phone Contact	Reqs12/2022	Patient Registration		
Patient Intrinsic Concepts	Reqs12/2022	**Patient**	Health Professional	Reqs12/2022	Patient Registration	Regs	Editing of Patient association to the HP
	Patient Related Concepts (11 attributes)						
Patient Related Concepts		**Sociodemographic Data**		Reqs12/2022	Patient Registration	Reqs12/2022	Sociodemographic Data Update
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Education Level	Reqs12/2022	Patient Registration	Reqs12/2022	Sociodemographic Data Update
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Date of Birth	Reqs12/2022	Patient Registration	Reqs12/2022	Sociodemographic Data Update
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Sex	Reqs12/2022	Patient Registration	Reqs12/2022	Sociodemographic Data Update
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Laterality	Reqs12/2022	Patient Registration	Reqs12/2022	Sociodemographic Data Update
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Professional Situation	Reqs12/2022	Patient Registration	Reqs12/2022	Sociodemographic Data Update
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Profession	Reqs12/2022	Patient Registration	Reqs12/2022	Sociodemographic Data Update
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Household	Reqs12/2022	Patient Registration	Reqs12/2022	Sociodemographic Data Update

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Patient Related Concepts	Reqs12/2022	**Sociodemographic Data**	Civil Status	Reqs12/2022	Patient Registration	Reqs12/2022	Sociodemographic Data Update
Patient Related Concepts		**Clinical Information**		Reqs12/2022	Clinical Information Registration	Reqs12/2022	Clinical Information Edition
Patient Related Concepts	Reqs12/2022	**Clinical Information**	Patient	Reqs12/2022	Clinical Information Registration	Reqs12/2022	Clinical Information Edition
Patient Related Concepts	Reqs12/2022	**Clinical Information**	Type	Reqs12/2022	Clinical Information Registration	Reqs12/2022	Clinical Information Edition
Patient Related Concepts	Reqs12/2022	**Clinical Information**	Description	Reqs12/2022	Clinical Information Registration	Reqs12/2022	Clinical Information Edition
	Health Professional Intrinsic Concepts (3 attributes)						
Health Professional Intrinsic Concepts		**Health Professional**		Reqs12/2022	Health Professional Details Registration	UC	Health Professional Details Edition
Health Professional Intrinsic Concepts	Reqs12/2022	**Health Professional**	Phone Contact	Reqs12/2022	Health Professional Details Registration	UC	Health Professional Details Edition
Health Professional Intrinsic Concepts	Reqs12/2022	**Health Professional**	Professional Cell	Reqs12/2022	Health Professional Details Registration	UC	Health Professional Details Edition
Health Professional Intrinsic Concepts	Reqs12/2022	**Health Professional**	Joining Date	Reqs12/2022	Health Professional Details Registration	UC	Health Professional Details Edition
	Administrator Intrinsic Concepts (1 attributes)						
Administrator Intrinsic Concepts		**Administrator**		RaNT Prototype	Administrator Details Registration	RaNT Prototype	Administrator Details Edition
Administrator Intrinsic Concepts	NexusBRaNT Prototype	**Administrator**	Phone Contact	NexusBRaNT Prototype	Administrator Details Registration	NexusBRaNT Prototype	Administrator Details Edition
	Cognitive Training Program Instrinsic Concepts (9 attributes)						
Cognitive Training Program Instrinsic Concepts		**Cognitive Training Program**		Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Patient	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Start Date	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Number of Training Sessions	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Weekly Recurrence	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Days of the Week	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	Training Session Duration	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	Reqs12/2022	**Cognitive Training Program**	State	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	BdNexus	**Cognitive Training Program**	Minimum Adaptation Challenge	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
Cognitive Training Program Instrinsic Concepts	BdNexus	**Cognitive Training Program**	Maximum Adaptation Challenge	Reqs12/2022	Cognitive Training Program Creation	Reqs12/2022	Cognitive Training Program Edition
	Neuropsychological Assessment Session Intrinsic Concepts (4 attributes)						
Neuropsychological Assessment Session Intrinsic Concepts		**Neuropsychological Assessment**		Reqs12/2022	Neuropsychological Assessment Session Creation	Reqs12/2022	Neuropsychological Assessment Session Edition
Neuropsychological Assessment Session Intrinsic Concepts	BdNexus	**Neuropsychological Assessment Session**	Patient	Reqs12/2022	Neuropsychological Assessment Session Creation	Reqs12/2022	Neuropsychological Assessment Session Edition
Neuropsychological Assessment Session Intrinsic Concepts	BdNexus	**Neuropsychological Assessment Session**	Health Professional	Reqs12/2022	Neuropsychological Assessment Session Creation	Reqs12/2022	Neuropsychological Assessment Session Edition
Neuropsychological Assessment Session Intrinsic Concepts	Reqs12/2022	**Neuropsychological Assessment Session**	Session Number	Reqs12/2022	Neuropsychological Assessment Session Creation	Reqs12/2022	Neuropsychological Assessment Session Edition
Neuropsychological Assessment Session Intrinsic Concepts	Reqs12/2022	**Neuropsychological Assessment Session**	Execution Date	Reqs12/2022	Neuropsychological Assessment Session Creation	Reqs12/2022	Neuropsychological Assessment Session Edition
	Neuropsychological Assessment Session Related Concepts (160 attributes)						
Neuropsychological Assessment Session Related Concepts		**Clinical Interpretation**		Reqs12/2022	Clinical Interpretation Calculation		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Clinical Interpretation**	General Cognition	Reqs12/2022	Clinical Interpretation Calculation		

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Neuropsychological Assessment Session Related Concepts	BdNexus	**Clinical Interpretation**	Memory	Regs12/2022	Clinical Interpretation Calculation		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Clinical Interpretation**	Executive Functions	Regs12/2022	Clinical Interpretation Calculation		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Clinical Interpretation**	Language	Regs12/2022	Clinical Interpretation Calculation		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Clinical Interpretation**	Attention	Regs12/2022	Clinical Interpretation Calculation		
Neuropsychological Assessment Session Related Concepts		**Cognitive Profile**		Regs12/2022	Cognitive Profile Calculation		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Cognitive Profile**	General Cognition	Regs12/2022	Cognitive Profile Calculation		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Cognitive Profile**	Memory	Regs12/2022	Cognitive Profile Calculation		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Cognitive Profile**	Executive Functions	Regs12/2022	Cognitive Profile Calculation		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Cognitive Profile**	Language	Regs12/2022	Cognitive Profile Calculation		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Cognitive Profile**	Attention	Regs12/2022	Cognitive Profile Calculation		
Neuropsychological Assessment Session Related Concepts		**Cognitive Normative Data Interpr**			Cognitive Normative Data Interpretation Creation		Cognitive Normative Data Interpretation Edition
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Cognitive Normative Data Interpretation**	Interpretation Level		Cognitive Normative Data Interpretation Creation		Cognitive Normative Data Interpretation Edition
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Cognitive Normative Data Interpretation**	Standard Deviation from the Average		Cognitive Normative Data Interpretation Creation		Cognitive Normative Data Interpretation Edition
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Cognitive Normative Data Interpretation**	Scale		Cognitive Normative Data Interpretation Creation		Cognitive Normative Data Interpretation Edition
Neuropsychological Assessment Session Related Concepts		**Scalar Score Interpretation**			Scalar Score Interpretation Creation		Scalar Score Interpretation Edition
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Scalar Score Interpretation**	Interpretation Level		Scalar Score Interpretation Creation		Scalar Score Interpretation Edition

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Neuropsychological Assessment Session Related Concepts	BRaNT NexusBRaNT Prototype	**Scalar Score Interpretation**	Scalar Score		Scalar Score Interpretation Creation		Scalar Score Interpretation Edition
Neuropsychological Assessment Session Related Concepts	BRaNT NexusBRaNT Prototype	**Scalar Score Interpretation**	Scale		Scalar Score Interpretation Creation		Scalar Score Interpretation Edition
Neuropsychological Assessment Instruments							
Neuropsychological Assessment Session Related Concepts		**MOCA**			MOCA Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Visuospatial / Executive		MOCA Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Nomination		MOCA Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Attention		MOCA Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Language		MOCA Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Abstraction		MOCA Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Deferred Recall		MOCA Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Orientation		MOCA Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**MOCA**	Total		MOCA Registration		
Neuropsychological Assessment Session Related Concepts	BRaNT NexusBRaNT Prototype	**MOCA**	Total - Average		MOCA Registration		
Neuropsychological Assessment Session Related Concepts	BRaNT NexusBRaNT Prototype	**MOCA**	Total - Standard Deviation		MOCA Registration		
Neuropsychological Assessment Session Related Concepts	BRaNT NexusBRaNT Prototype	**MOCA**	Total - Clinical Interpretation		MOCA Registration		
Neuropsychological Assessment Session Related Concepts	BRaNT NexusBRaNT Prototype	**MOCA**	Observations		MOCA Registration		

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Neuropsychological Assessment Session Related Concepts		**WAIS-III**			WAIS-III Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WAIS-III**	Symbol Search		WAIS-III Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Symbol Search - Scalar Score		WAIS-III Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Symbol Search - Clinical Interpretation		WAIS-III Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Code		WAIS-III Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Code - Scalar Score		WAIS-III Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Code - Clinical Interpretation		WAIS-III Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Vocabulary		WAIS-III Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Vocabulary - Scalar Score		WAIS-III Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Vocabulary - Clinical Interpreation		WAIS-III Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WAIS-III**	Observations		WAIS-III Registration		
Neuropsychological Assessment Session Related Concepts		**GDS-15**			GDS-15 Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**GDS-15**	GDS-15		GDS-15 Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**GDS-15**	Clinical Interpretation		GDS-15 Registration		

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**GDS-15**	Observations		GDS-15 Registration		
Neuropsychological Assessment Session Related Concepts		**GDS-30**			GDS-30 Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**GDS-30**	GDS-30		GDS-30 Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**GDS-30**	Clinical Interpretation		GDS-30 Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**GDS-30**	Observations		GDS-30 Registration		
Neuropsychological Assessment Session Related Concepts		**GATSB**			GATSB Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**GATSB**	GATSB		GATSB Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**GATSB**	Observations		GATSB Registration		
Neuropsychological Assessment Session Related Concepts		**Semantic and Phonemic Verbal Fluency**			Semantic and Phonemic Verbal Fluency Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Semantic and Phonemic Verbal Fluency**	Semantic Verbal Fluency - Animals		Semantic and Phonemic Verbal Fluency Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Semantic and Phonemic Verbal Fluency**	Semantic Verbal Fluency - Animals - Scalar Score		Semantic and Phonemic Verbal Fluency Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Semantic and Phonemic Verbal Fluency**	Semantic Verbal Fluency - Animals - Clinical Interpretation		Semantic and Phonemic Verbal Fluency Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Semantic and Phonemic Verbal Fluency**	Semantic Verbal Fluency - Professions (FVSP)		Semantic and Phonemic Verbal Fluency Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Semantic and Phonemic Verbal Fluency**	Phonemic Verbal Fluency M (FVFM)		Semantic and Phonemic Verbal Fluency Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Semantic and Phonemic Verbal Fluency**	Phonemic Verbal Fluency R (FVFR)		Semantic and Phonemic Verbal Fluency Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Semantic and Phonemic Verbal Fluency**	Phonemic Verbal Fluency P (FVFP)		Semantic and Phonemic Verbal Fluency Registration		

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Neuropsychological Assessment Session Related Concepts	BdNexus	**Semantic and Phonemic Verbal Fluency**	Phonemic Verbal Fluency - Total		Semantic and Phonemic Verbal Fluency Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Semantic and Phonemic Verbal Fluency**	Phonemic Verbal Fluency - Scalar Score		Semantic and Phonemic Verbal Fluency Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Semantic and Phonemic Verbal Fluency**	Phonemic Verbal Fluency - Clinical Interpretation		Semantic and Phonemic Verbal Fluency Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Semantic and Phonemic Verbal Fluency**	Observations		Semantic and Phonemic Verbal Fluency Registration		
Neuropsychological Assessment Session Related Concepts		**FCSRT**			FCSRT Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**FCSRT**	Immediate Free Recall		FCSRT Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**FCSRT**	Immediate Guided Recall		FCSRT Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**FCSRT**	Total Immediate Recall (Free + Guided)		FCSRT Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**FCSRT**	Deferred Free Recall		FCSRT Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**FCSRT**	Deferred Guided Recall		FCSRT Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**FCSRT**	Total Deferred Recall (Free + Guided)		FCSRT Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**FCSRT**	Observations		FCSRT Registration		
Neuropsychological Assessment Session Related Concepts		**IAFAI**			IAFAI Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**IAFAI**	ABVD		IAFAI Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**IAFAI**	AIVD-F		IAFAI Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**IAFAI**	AIVD-A		IAFAI Registration		

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Neuropsychological Assessment Session Related Concepts	BdNexus	**IAFAI**	Total		IAFAI Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**IAFAI**	Observations		IAFAI Registration		
Neuropsychological Assessment Session Related Concepts		**PRECiS**			PRECiS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**PRECiS**	Total Score		PRECiS Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**PRECiS**	Observations		PRECiS Registration		
Neuropsychological Assessment Session Related Concepts		**QSM**			QSM Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**QSM**	Total Score		QSM Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**QSM**	Observations		QSM Registration		
Neuropsychological Assessment Session Related Concepts		**Toulouse-Piéron**			Toulouse-Piéron Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Toulouse-Piéron**	Realization Capacity		Toulouse-Piéron Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Toulouse-Piéron**	Dispersion Index		Toulouse-Piéron Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Dispersion Index - Average		Toulouse-Piéron Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Dispersion Index - Standard Deviation		Toulouse-Piéron Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Dispersion Index - Clinical Interpretation		Toulouse-Piéron Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Toulouse-Piéron**	Work Efficiency		Toulouse-Piéron Registration		

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Work Efficiency - Average		Toulouse-Piéron Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Work Efficiency - Standard Deviation		Toulouse-Piéron Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Work Efficiency - Clinical Interpretation		Toulouse-Piéron Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Toulouse-Piéron**	Observations		Toulouse-Piéron Registration		
Neuropsychological Assessment Session Related Concepts		**SIS**			SIS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Strength		SIS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Hand Strength		SIS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Mobility		SIS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Daily Life Activities		SIS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Emotional		SIS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Memory		SIS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Communication		SIS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Social Engagement		SIS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**SIS**	Stroke Recovery		SIS Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**SIS**	Observations		SIS Registration		
Neuropsychological Assessment Session Related Concepts		**WMS**			WMS Registration		

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Neuropsychological Assessment Session Related Concepts	BdNexus	**WMS**	Immediate Recall		WMS Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Immediate Recall - Scalar Score		WMS Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Immediate Recall - Clinical Interpretation		WMS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WMS**	Long-term Recall		WMS Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Long-term Recall - Scalar Score		WMS Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Long-term Recall - Clinical Interpretation		WMS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WMS**	Copy		WMS Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Copy - Scalar Score		WMS Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Copy - Clinical Interpretation		WMS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WMS**	Discrimination		WMS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WMS**	Recognition		WMS Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Recognition - Scalar Score		WMS Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Recognition - Clinical Interpretation		WMS Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WMS**	Retention Percentage		WMS Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Retention Percentage - Scalar Score		WMS Registration		

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Retention Percentage - Clinical Interpretation		WMS Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WMS**	Observations		WMS Registration		
Neuropsychological Assessment Session Related Concepts		**WHOQOL-BREF**			WHOQOL-BREF Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-BREF**	Domain 1		WHOQOL-BREF Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-BREF**	Domain 2		WHOQOL-BREF Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-BREF**	Domain 3		WHOQOL-BREF Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-BREF**	Domain 4		WHOQOL-BREF Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WHOQOL-BREF**	Observações		WHOQOL-BREF Registration		
Neuropsychological Assessment Session Related Concepts		**WHOQOL-OLD**			WHOQOL-OLD Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Sensory Functioning		WHOQOL-OLD Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Autonomy		WHOQOL-OLD Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Past / Present / Future Activities		WHOQOL-OLD Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Social Engagement		WHOQOL-OLD Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Death / Dying		WHOQOL-OLD Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Intimacy		WHOQOL-OLD Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Family and Family Life		WHOQOL-OLD Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**WHOQOL-OLD**	Total		WHOQOL-OLD Registration		

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**WHOQOL-OLD**	Observations		WHOQOL-OLD Registration		
Neuropsychological Assessment Session Related Concepts		**Rey's Complex Figure**			Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Rey's Complex Figure**	Copy		Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Copy - Average		Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Copy - Standard Deviation		Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Copy - Clinical Interpretation		Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Immediate Memory		Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Immediate Memory - Average		Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Immediate Memory - Standard Deviation		Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Immediate Memory - Clinical Interpretation		Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Deferred Memory		Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Deferred Memory - Average		Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Deferred Memory - Standard Deviation		Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Deferred Memory - Clinical Interpretation		Rey's Complex Figure Registration		

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey's Complex Figure**	Observations		Rey's Complex Figure Registration		
Neuropsychological Assessment Session Related Concepts		**Rey15**			Rey15 Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Rey15**	Immediate Recall Test		Rey15 Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**Rey15**	Combined Recognition Result		Rey15 Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**Rey15**	Observations		Rey15 Registration		
Neuropsychological Assessment Session Related Concepts		**BDI-II**			BDI-II Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**BDI-II**	BDI-II		BDI-II Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**BDI-II**	Clinical Interpretation		BDI-II Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**BDI-II**	Observations		BDI-II Registration		
Neuropsychological Assessment Session Related Concepts		**QOLIBRI**			QOLIBRI Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	Total		QOLIBRI Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	Cognitive		QOLIBRI Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	Self		QOLIBRI Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	AVDs		QOLIBRI Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	Social Relationships		QOLIBRI Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	Emotions		QOLIBRI Registration		

Scope	Source	Concept	Attribute Name	Task 1's Source	Task 1 creating / modifying the attribute	Task 2's Source	Task 2 creating / modifying the attribute
Neuropsychological Assessment Session Related Concepts	BdNexus	**QOLIBRI**	Physical Condition		QOLIBRI Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**QOLIBRI**	Observations		QOLIBRI Registration		
Neuropsychological Assessment Session Related Concepts		**CDR**			CDR Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Memory		CDR Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Orientation		CDR Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Judgement and Problem Solving		CDR Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Activities in the Communities		CDR Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Home and Hobbies		CDR Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Personal Care		CDR Registration		
Neuropsychological Assessment Session Related Concepts	BdNexus	**CDR**	Total		CDR Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**CDR**	Clinical Interpretation		CDR Registration		
Neuropsychological Assessment Session Related Concepts	NexusBRaNT Prototype	**CDR**	Observations		CDR Registration		