DM

Enhancing Biodiversity Documentation through an Integrated and Interoperable Web-based Software Framework

MASTER DISSERTATION

Sebastian José de Freitas Gonçalves
MASTER IN INFORMATICS ENGINEERING

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

September | 2025

Enhancing Biodiversity Documentation through an Integrated and Interoperable Web-based Software Framework

MASTER DISSERTATION

Sebastian José de Freitas Gonçalves
MASTER IN INFORMATICS ENGINEERING

SUPERVISOR
Marko Radeta

Faculdade de Ciências Exatas e da Engenharia

Enhancing Biodiversity Documentation through an Integrated and Interoperable Web-based Software Framework

Author:

Sebastian José de Freitas Gonçalves

Supervised by:

Professor Dr. Marko Radeta

Constitution of the public examination jury:

Professor Dr. David Sardinha Andrade de Aveiro, President

Professor Dr. Mara Sofia Gomes Dionísio, Member

Professor Dr. Marko Radeta, Member

Tuesday 9th December, 2025

Resumo

Embora a estimativa da biodiversidade aponte para cerca de 50 milhões de espécies em todo o mundo, apenas 4% se encontram documentadas. As bases de dados, estruturas e normas existentes para registar e fornecer a taxonomia das espécies de biodiversidade mundial continuam a ser de utilização complexa e exigem um elevado nível de especialização na área. Num esforço para aproximar a normalização terrestre e aquática no reporte da biodiversidade, esta dissertação apresenta uma interface de back-office baseada na web, que facilita o reporte e a taxonomia dessas espécies. A ferramenta proposta inclui Interfaces de Programação de Aplicações (APIs) interoperáveis, permitindo a integração centralizada de informação como espécies, ocorrências, taxonomia e referências, oferecendo ao utilizador final uma personalização de fácil utilização. O sistema é validado por um especialista na área da biologia, recorrendo a metodologias ágeis de desenvolvimento de software. O sistema contribui assim para a interoperabilidade global, facilitando o reporte e a taxonomia de todas as espécies de biodiversidade.

Keywords: Classificação da Biodiversidade, Taxonomia, Base de Dados Interoperável, Registo de Espécies, Interface Web

Abstract

While the estimate of biodiversity suggests 50 million species worldwide, only 4% of them remain documented. Existing databases, frameworks and standards to report and provide taxonomy of worldwide biodiversity species remain cumbersome to use and require significant need for domain expertise. In effort to bridge the terrestrial and aquatic standardization for reporting biodiversity, this dissertation provides the web-based back-office interface, facilitating the reporting and taxonomy of such species. Proposed tool comes with the interoperable Application Programming Interfaces (APIs), allowing the centralized integration of information such as species, occurrences, taxonomy and references, providing easy-to-use customization to the end user. The system is validated by domain expert in biology using the agile software development methodologies. The system contributes in providing the overall interoperability, facilitating the reporting and taxonomy of all biodiversity species.

Keywords: Biodiversity Classification, Taxonomy, Interoperable Database, Species Register, Web-Based Interface

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Marko Radeta, for his continuous guidance, encouragement, and invaluable support throughout the development of this thesis. His expertise, constructive feedback, and patience were fundamental in shaping the direction of this work and in helping me overcome the many challenges encountered along the way. I am also sincerely thankful to Prof. Dora Pombo and to the students of the Insect Collection of the University of Madeira (Faculty of Life Science)—Mafalda Abreu, Cesar Alves, Hugo Silva, Luena Soraya, Thais Coppen, and Rui Marques. Their expertise in biodiversity research, active engagement, and constructive feedback was invaluable in ensuring that this project remained closely aligned with real-world needs. Their willingness to share their expertise, answer questions, and provide continuous feedback was crucial for aligning the system with the requirements of biodiversity documentation and conservation. Finally, I wish to extend my heartfelt appreciation to all those who, directly or indirectly, contributed to the success of this project. Your support and encouragement have been invaluable, and this work would not have been possible without you.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ABCD Access to Biological Collection Data

ACID Atomicity, Consistency, Isolation, and Durability

AI Artificial Intelligence

APIs Application Programming Interfaces

ARIA Accessible Rich Internet Applications

BHL Biodiversity Heritage Library

BOLD Barcode of Life Data System

CITES Convention on International Trade in Endangered Species

CLI Command-Line Interface

CNNs Convolutional Neural Networks

COL Catalogue of Life

CSRF Cross-Site Request Forgery

CSS Cascading Style Sheets

CSV Comma-Separated Values

DOIs Digital Object Identifiers

DOM Document Object Model

DwC Darwin Core

DwC-A Darwin Core Archive

EJS Embedded JavaScript

EMODnet European Marine Observation and Data Network

ERMS European register of Marine Species

EoL Encyclopedia of Life

FTP File Transfer Protocol

GBIF Global Biodiversity Information Facility

GPS Global Positioning System

GSDs Global Species Databases

GUI Graphical User Interface

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IBEIS Image-Based Ecological Information System

IPBES Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services

IPT Integrated Publishing Toolkit

ITIS Integrated Taxonomic Information System

IUCN International Union for Conservation of Nature

JSON JavaScript Object Notation

LSIDs Life Science Identifiers

MBARI Monterey Bay Aquarium Research Institute

MIT Massachusetts Institute of Technology

MSA Microservices Architecture

MUI Material UI

MVC Model-View-Controller

MVCC Multiversion Concurrency Control

MVP Model-View-Presenter

MVVM Model-View-ViewModel

NOAA National Oceanic and Atmospheric Administration

NPM Node Package Manager

NoSQL Not Only SQL

OBIS Ocean Biodiversity Information System

OBIS-USA Ocean Biodiversity Information System – United States of America

ORM Object Relational Mapper

P2P Peer-to-Peer

PWAs Progressive Web Apps

RDBMS Relational Database Management System

SFCs Single File Components

SMTP Simple Mail Transfer Protocol

SPAs Single-Page Applications

SQL Structured Query Language

SUS System Usability Scale

TDWG Taxonomic Databases Working Group

UI User Interface

UN United Nations

URL Uniform Resource Locator

VARS Video Annotation and Reference System

VLIZ Flanders Marine Institute

WPD Workflow Process Description

WPF Windows Presentation Foundation

WoRMS World Register of Marine Species

XSS Cross-Site Scripting

1 Introduction

1.1 Current Context of the Research, Motivation, and Problem Specification

Biodiversity documentation of our planet´s species still remains very lacking, with estimates of total number of species on Earth ranging between 5 million to 50 million, and only about 1.7 to 2 million species formally described and cataloged to date [1]. This discrepancy highlights just how big and complex can life on Earth be, as well demonstrates how challenging can be stundying and documenting biodiversity. Also much of the world´s biodiversity is located in certain locations, such as the Amazon basin, Southeast Asia, and Central Africa, which are often hard to access, politically unstable, or a lack of appropriate research infrastructure. Traditionally the process of classifying species, known as taxonomy, relies on careful morphological analysis, extensive field-work, laboratory genetics, and anatomical verification. These methods, while very accurate and rigorous, are slow, resource intensive, and highly dependent on a small and decreasing number of taxonomic experts worldwide [2]. Many institutions have seen a decline in taxonomic experts, as less students enter the field and funding is cut to focus in molecular or high-throughput disciplines over classical taxonomy. Also with the impact of global warming, ecosystems are changing rapidly, with changes in temperature, rain patterns, and habitats, entire species are being disrupted or in the worst cases driven to extinction faster than they can be discovered and documented [3]. The Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services (IPBES) warns that up to a million species may be at risk of extinction within decades unless changes occur in the way humanity interacts with nature. This urgency highlights the need for tools and systems that can help streamline, scale, and improve biodiversity documentation. Emerging technologies like artificial intelligence (AI), cloud computing, mobile data collection, and citizen science platforms offer promising solutions, but unless these tools are made accessible, standardized, and interoperable across ecological and institutional boundaries, they risk replicating existing silos or reinforcing biases toward well-known taxa and regions.

1.2 Project Relevance

The decline in biodiversity documentation has significant implications for conservation biology, ecosystem management, and our understanding of life on Earth. So to try to address this issue,

several platforms have emerged in recent years to facilitate biodiversity documentation, making it more scalable, participatory, and data-rich. Every single one has its own approach and focus, some focus on mobily technology, others on clould infrastructure, others in artificial intelligence, and engaging citizen scientists to contribute. Some examples of platforms are INaturalist, for mobile technology and citizen science [4–6], eBird for bird observations [7], Pl@ntNet for plant identification using machine learning [8], Happy Whale for whale identification using computer vision [9], and CoralNet for coral reef monitoring [10]. A number of institutions have also emerged to provide standardization and aggregation of biodiversity data, such as Darwin Core [11], GBIF (Global Biodiversity Information Facility) [12], OBIS (Ocean Biodiversity Information System) [13], WoRMS (World Register of Marine Species) [14], and the Catalogue of Life [15]. These platforms and tools like Specify[1], Arctos [16], and Turboveg [17] have their strengths and limitations, but they represent a fragmented field, with solutions that are often very domain-specific or difficult to use and understand. What is needed is a cohesive and integrated framework that provides a simple platform that biologists can use to help them organize and document their findings helping improve their workflows, so that taxonomic verification can be done more efficiently.

1.3 Objective and Preview of the Proposed Work

The goal of this dissertation is to design and implement a web-based biodiversity data framework that helps address the problem of documenting biodiversity data. Existing platforms have made significant improvements in helping identify and handle biodiversity data, but they often have limitations in terms of scope, usability, or interoperability. So this project aims to create a unified platform that integrates both terrestrial and aquatic taxonomies within a single interoperable system, allowing for a better organization and management of biodiversity data across different domains. The proposed framework builds upon established standards such as Darwin Core to ensure data consistency and compatibility with global biodiversity infrastructures. A system that emphasizes usability, modularity, and scalability, making the platform easy to understand and user friendly. Such platform can help experts, by streamlining their workflows by organizing, simplifying, and improving the efficiency of their documentation process. This way improving the overall quality and accessibility of biodiversity data, which contributes to better conservation efforts improving the global understanding of biodiversity patterns and trends.

[1] https://www.specifysoftware.org/

1.4 Document structure

This dissertation is structured as follows:

Chapter 1 introduces the research problem, establishes its relevance, and outlines the dissertation goals and structure.

Chapter 2 surveys the existing landscape of biodiversity documentation tools and standards, highlighting their strengths and limitations.

Chapter 3 presents the design and architectural principles of the proposed system, including integration strategies for cross-domain taxonomies.

Chapter 4 details the implementation process, with attention to both technical components and user interface considerations.

Chapter 5 describes the validation methodology, reports on collaborative feedback, and evaluates system performance.

Chapter 6 concludes with a discussion of the project.

2 Related Work

Biodiversity systems are critical tools for the global effort to understand, conserve, and sustainably manage the planet's biodiversity. In an era where global warming is rapidly changing ecosystems, species are going extinct at alarming rates, so the need for biodiversity information systems that can provide the necessary infrastructure for scientific research, conservation, and ecological monitoring is more important than ever. With the evolution of biodiversity science into the modern era of digital information and big data, the need for scalable and interoperable digital platforms has become necessary. Over the past decades, a wide range of digital infrastructures and platforms have been developed to document and integrate biodiversity data into the digital information space. These systems vary widely in terms of their scope, scale, and user base. Large institutional repositories like the Global Biodiversity Information Facility (GBIF) [12, 18], the World Register of Marine Species (WoRMS) [14, 19], and the Ocean Biodiversity Information System (OBIS) [13, 20] focus on aggregating and standardizing occurrence and taxonomic data, providing an essential repository for biodiversity research. Platforms for citizen science like iNaturalist [4–6] have emerged to complement institutional efforts, expanding data collection through participatory monitoring. In more recent years, with the rise of artificial intelligence (AI), platforms like FathomNet [21, 22] have introduced recognition tools that facilitate image identification. All these systems and others have their own strengths and weaknesses, which will be explored in the following sections.

This literature review combines both traditional academic resources and AI-assisted research tools to ensure a in depth coverage. Peer-reviewed articles and conference proceedings were accessed through databases such as Google Scholar[2] and the ACM Digital Library[3]. Complementary searches were conducted using AI (Artificial Intelligence) -powered assistants including Phind[4] and Research Rabbit[5], which facilitated the identification of emerging themes and interconnections among publications. Reference management and citation integration were supported by Zotero[6].

[2] https://scholar.google.com/
[3] https://dl.acm.org/
[4] https://www.phind.com/
[5] https://www.researchrabbit.ai/
[6] https://www.zotero.org/

2.1 Specify

One of the most established software packages for managing biodiversity data is Specify[7] , an open-source platform developed by the Specify Collections Consortium, originating at the University of Kansas Biodiversity Institute. First released in the 1990s, Specify was designed to manage biological specimen collections in museums, herbaria, and similar institutions. It provides comprehensive functionality, including specimen tracking, hierarchical taxonomy management, custom reporting, and integration with external biodiversity networks such as GBIF. Its robust relational database design allows for handling large and complex datasets, making it essential for institutions with extensive collections. Specify supports curators, collection managers, and researchers in making biological collections more accessible, searchable, and scientifically valuable. Despite its technical strengths, Specify has notable limitations. Its steep learning curve and dependence on database knowledge can be barriers for smaller institutions or those without dedicated IT staff. Documentation is not always comprehensive, and real-time support may be limited outside the Specify Consortium. The user interface is considered outdated, lacking modern usability features, which can reduce efficiency when managing extensive datasets or complex taxonomies. Furthermore, Specify requires local installation or institutional servers, complicating deployment, maintenance, data backup, and upgrades, particularly for resource-limited institutions. Although supported by an open-source consortium of around 70 institutions, development can be slow due to resource constraints, community prioritization, and funding cycles, limiting responsiveness to evolving user needs or new standards. Nonetheless, Specify's architectural integrity, adherence to global biodiversity standards, and ability to manage large biological collections offer a valuable reference. For this dissertation, Specify serves both as a benchmark and a cautionary example. The proposed platform aims to preserve Specify's scientific rigor and data durability while improving usability, accessibility, and interoperability using modern web-native technologies, creating a solution suitable for institutional use and broader community engagement.

2.2 iNaturalist

iNaturalist is a prominent citizen science platform for global biodiversity monitoring, originally created in 2008 at the University of California, Berkeley, and adopted by the California Academy of Sciences in 2011 [4–6]. The platform allows users to upload images of organisms with metadata

[7] https://www.specifysoftware.org/

such as location and date. Identifications are generated through a hybrid AI-human system, and when consensus is reached, observations achieve "research-grade" status and are integrated into global databases like GBIF [5]. iNaturalist supports a broad taxonomic range—including birds, fungi, insects, plants, and indirect evidence like tracks—and facilitates collaborative identification, taxonomy discussions, and ecological learning [4, 5]. The platform is widely used for education, providing opportunities to teach biodiversity, ecological literacy, and digital skills. Projects and custom data filters allow flexible organization of observations for research, institutional, or personal goals [4]. However, iNaturalist faces challenges. Species identification accuracy depends on image quality, community expertise, and AI limitations, especially for obscure or hard-to-photograph taxa [4]. Geographic and taxonomic biases arise from uneven smartphone access and photogenic species preference [6]. AI models may over-represent common species, and indirect evidence remains difficult to classify [5]. Accessibility issues exist for users in under-resourced regions or those less familiar with mobile apps, and privacy concerns can arise from public geolocation. Pedagogically, over-reliance on AI may reduce learning of traditional identification skills, suggesting a blended approach combining digital tools with classical methods [4]. Despite these limitations, iNaturalist demonstrates a successful integration of intuitive interfaces, AI, and community validation for scalable biodiversity data. By 2022, it had accumulated approximately 129 million observations from 6 million users, with new species attaining research-grade status every 1.7 hours [4, 5]. Its open-data model and real-time integration with biodiversity systems make it both a scientific and educational resource, offering a replicable model for inclusive, user-centered biodiversity platforms.

2.3 Monicet

Monicet[8] is a long-term monitoring system for cetacean sightings in the Azores, initiated in 2009 with support from the Azores Government to leverage whale watching for scientific data collection [23]. The project partners scientists and tour operators to produce high-quality, standardized biodiversity data that meet regulatory, operational, and research needs. Between 2009 and 2020, Monicet compiled over 37,000 sightings of 25 marine species across four islands [23]. Data are collected systematically by trained observers following established protocols, starting with land-based spotters directing boats, and then guides recording species, behavior, group structure, environmental conditions, and GPS coordinates [23]. Data is uploaded to an online database,

[8] http://www.monicet.net/

validated by a centralized team, and structured according to Darwin Core Archive standards, with whale-watching trips as parent events and individual sightings as nested events. Then this dataset is shared with global repositories like GBIF, OBIS, EMODnet (European Marine Observation and Data Network), and IPT (Integrated Publishing Toolkit) [23]. Monicet's success lies in its ability to produce high-quality, geographically targeted data at low operational cost while integrating commercial, governmental, and scientific goals. However, limitations include the lack of quantifiable observation effort, potential misidentification of species, biases in behavioral observations, inaccurate group size estimates, possible duplicate sightings, and occasional errors during manual data entry [23]. Despite these constraints, Monicet exemplifies professionally facilitated, location-specific citizen science and provides a replicable model for incorporating tourism-based wildlife observations into long-term biodiversity monitoring.

2.4 Dive Reporter

Dive Reporter offers an innovative strategy combining structured protocols and crowd-sourcing aimed at the recreational dive industry. Created by Wave Labs [9], this mobile app aims to involve dive guides and SCUBA divers in the systematic collection of marine species occurrence data using a semi-structured approach [24]. Region-specific checklists of ecologically important taxa are initially created by marine biologists and afterwards utilized by dive operators to aid post-dive interviews. Divers note the abundance of various species on a five-step abundance scale, reducing subjectivity and enabling consistency between dives [24]. The application also records metadata like dive sites, number of divers, and dive time, enabling normalization of effort at observation—a factor usually missing from conventional citizen science platforms. All observations that are recorded are kept in Wave Labs' centralized database, enabling visualization, analysis, and integration into larger ecological research. The platform enables temporal and spatial analysis, quality control processes, and the capacity to detect biodiversity hot spots or temporal trends [24]. There are, nevertheless, limitations to Dive Reporter. Misidentification of species is a significant issue, especially since the divers do not need to provide photographic proof, as it is usually impossible under water conditions. Information quality mostly depends on the expertise of the diver and the interpretive abilities of the dive guide. Furthermore, reporting bias may arise where divers provide selective reports of charismatic or notable species (e.g., sharks or groupers) and ignore less well-known taxa

[9] https://wave-labs.org/

like invertebrates or algae. While early user feedback has been good, the site remains reliant on the ongoing enthusiasm of dive operators, who may have conflicting operational responsibilities. Such factors may result in variability in both the amount and frequency of data over time. Yet, Dive Reporter illustrates the potential of rigorous methodology and targeted engagement strategies to enhance biodiversity data collection in long-overlooked marine ecosystems. This case is significant in that it represents an example of trading off methodological rigor with user ease, a fundamental rule of system design that is both scientifically relevant and widely accessible.

2.5 HappyWhale

HappyWhale[10] is a web-based platform for photographic identification and long-term tracking of whales and dolphins, founded in 2015 with the dual goals of engaging citizen scientists and providing high-quality, low-cost data for marine conservation [9]. Contributors—including researchers, tourists, and naturalists—submit photos of whale flukes or dorsal fins, which are processed by an automatic image recognition algorithm adapted from the IBEIS (Image-Based Ecological Information System) project. Images are preprocessed for optimal recognition, then compared against a curated global reference catalog [9]. Matches trigger automatic notifications summarizing the whale's history, migration, and behavior. Unmatched images undergo manual verification by regional specialists, which ensures data integrity but constitutes a workflow bottleneck [9]. By April 2017, over 41,000 images had been submitted by more than 1,000 contributors, with numbers growing steadily [9]. The platform enables long-distance matching between geographically dispersed datasets, linking sightings across regions like the U.S. West Coast, Alaska, Baja California, and Hawaii. Researchers can access curated encounter records under flexible, mostly open-access licensing [9]. HappyWhale's rapid identification capability also supports time-sensitive interventions, such as whale entanglement responses. Limitations include dependency on image quality, narrow taxonomic scope, uneven contributor distribution, and biases toward charismatic species. Despite these constraints, HappyWhale demonstrates the value of combining algorithmic processing, human verification, and public engagement to create a globally accessible, scientifically robust citizen science platform [9].

[10] `https://happywhale.com/about`

2.6 FathomNet

FathomNet is an open-access, machine learning–enableable image database designed to leverage AI for ocean exploration, marine ecology monitoring, and conservation [21, 22]. It addresses the challenge of processing the large volumes of underwater imagery collected by research cruises by normalizing, annotating, and curating datasets for AI training in tasks such as organism detection, classification, and behavioral analysis. Developed collaboratively by MBARI (Monterey Bay Aquarium Research Institute), MIT Media Lab, CVision AI Inc., and NOAA (National Oceanic and Atmospheric Administration), FathomNet uses a relational database structured along Darwin Core standards, ensuring interoperability with international biodiversity repositories such as WoRMS [22]. Data is accessible via a REST API and web interface[11]. The platform employs a hierarchical concept tree adapted from MBARI's Video Annotation and Reference System (VARS), organizing data by taxonomy, equipment, substrate, and anthropogenic materials [21]. This structure supports diverse applications including habitat mapping, pollution monitoring, and species inventories. FathomNet incorporates over 30 years of imagery from MBARI, National Geographic, NOAA, and autonomous platforms, augmented with expert annotations and bounding boxes to facilitate AI training and ecological interpretation [21, 22]. Metadata—such as timestamps, depth, geolocation, imaging modality, and bounding boxes—enhances machine learning and broad ecological analyses. Challenges remain, including obtaining detailed, taxonomically accurate annotations and managing variability in underwater imagery (e.g., lighting, species density, and angles). Despite these limitations, FathomNet exemplifies scalable, cross-disciplinary ocean observation infrastructure, integrating expert-curated data with global standards to support AI-driven biodiversity monitoring [15, 25, 26].

2.7 Darwin Core

Darwin Core[12] (DwC) is a widely adopted metadata standard for sharing, discovering, and integrating biodiversity data, particularly species occurrence records [11,27]. It provides a structured vocabulary with clear semantics, documenting the who, what, when, and where of observations and specimens. Originating in 1999 through the Darwin Core Task Group under TDWG (Taxonomic Databases Working Group), the standard was formally ratified in 2009 after iterative community

[11] `https://database.fathomnet.org/fathomnet/#/`
[12] `https://dwc.tdwg.org/`

development [11]. DwC organizes terms into conceptual categories such as event, location, occurrence, taxon, and identification. Each term has semantic clarity to ensure consistency across datasets and platforms, promoting interoperability [11]. Its design is minimal yet flexible, allowing data publication in formats like CSV (Comma-Separated Values), XML, JSON, and RDF. The Darwin Core Archive (DwC-A) is the most common implementation, packaging core and extension files in a star schema for modular, extensible data [11]. Tools like GBIF's Integrated Publishing Toolkit (IPT) facilitate publication, validation, and registration, while custom applications such as Scratchpads extend the standard for domain-specific needs [11]. Limitations include the absence of a formal ontology defining relationships between terms, which hinders integration with complex datasets, linked data, or semantic web applications [27]. Data heterogeneity, inconsistent term usage, and varying DwC-A implementations also pose challenges. Specialized domains may require non-standard extensions, which are not universally supported. Despite these issues, Darwin Core's broad adoption by GBIF, iNaturalist, and the Biodiversity Heritage Library ensures compatibility, interoperability, and reusability of biodiversity data [27]. Integrating DwC into information systems enables global dataset alignment, facilitating scientific discovery and the development of a connected biodiversity informatics ecosystem.

2.8 Global Biodiversity Information Facility

The Global Biodiversity Information Facility (GBIF) is the world's largest open-access platform for biodiversity occurrence data, established in 2001 to provide universally accessible data for research, conservation, and policy [12, 18]. Operating as an international intergovernmental organization, GBIF coordinates a distributed network of national and thematic data publishers, aggregating records from museums, herbaria, research institutions, government agencies, and citizen science platforms. Its web portal[13] allows users to search, filter, map, and download species occurrence records, taxonomic checklists, specimen metadata, and observational datasets, supporting applications such as species distribution modeling, ecological forecasting, and conservation planning [18]. GBIF relies on open data standards, primarily Darwin Core and ABCD (Access to Biological Collection Data), ensuring harmonization and integration of diverse datasets [18]. Taxonomic consistency is maintained through links to authoritative backbones like the Catalogue of Life, Species 2000, and ITIS. GBIF emphasizes inclusivity, engaging over 60 countries and hundreds

[13] `https://www.gbif.org/`

of institutions, mobilizing both historical and contemporary biodiversity data, including records from underrepresented regions [18]. Challenges include uneven taxonomic and geographic coverage, data quality issues in identification and geolocation, integration complexities due to varied sources, and sensitive data management [12]. Legacy records and taxonomic concept discrepancies complicate analyses, while sustainable operation depends on stable funding and continued institutional support. Despite these limitations, GBIF provides foundational infrastructure for biodiversity research.

2.9 Ocean Biodiversity Information System

The Ocean Biodiversity Information System (OBIS) is a global, open-access platform aggregating marine biodiversity data, initiated in 1997 under the Sloan Foundation's Census of Marine Life [13, 20]. Designed as an electronic atlas of ocean life, OBIS integrates species occurrence data with environmental parameters, supporting marine research, conservation, and resource management. The platform links datasets from international partners adhering to common metadata, taxonomic, and structural standards. Users can explore, visualize, and analyze marine biodiversity patterns through a web interface[14] , enabling research on species distributions, habitat shifts, invasive species, and biodiversity hotspots. OBIS emphasizes data quality, employing expert-vetted taxonomic identifications, controlled vocabularies, Darwin Core standards, and references like WoRMS [13]. Its architecture supports dynamic data exploration, integration of datasets, spatial queries, and tailored outputs. OBIS also contributes to global initiatives, including the Global Ocean Observing System and the UN Decade of Ocean Science, and integrates with related platforms like GBIF and OBIS-USA. Challenges include gaps in species-level geographic coverage, particularly for benthic and deep-sea ecosystems, limited taxonomic resolution in some datasets, and data fragmentation due to disparate formats or legacy storage [13, 20]. Data contributions rely on voluntary submissions, which vary with funding and staffing. OBIS addresses this through standardized metadata, persistent contributor recognition, and integration with other biodiversity systems. Despite limitations, OBIS remains essential for marine biodiversity research, providing a georeferenced and taxonomically rigorous foundation.

[14] `https://obis.org/`

2.10 World Register of Marine Species

The World Register of Marine Species (WoRMS) is an authoritative, open-access database providing a standardized inventory of all published marine species names. Launched in 1997 as the European Register of Marine Species (ERMS) and expanded globally in 2007, WoRMS is hosted by the Flanders Marine Institute (VLIZ) [14,19]. Its mission is to maintain expert-validated taxonomy, including accepted names and synonyms, supporting historical and comparative biodiversity analyses. As of 2024, WoRMS contains over 460,000 taxonomic names, more than 368,000 species-level combinations, and over 215,000 accepted marine species [14]. WoRMS uses a relational database, Aphia, assigning each taxon a unique identifier (AphiaID) and integrating with over 100 global, 12 regional, and 4 thematic species databases [14]. The platform provides ecological, geographic, literature, and image metadata, and connects with major biodiversity initiatives such as COL, EoL (Encyclopedia of Life), GBIF, and OBIS [28]. Tools include the Taxon Match Tool, search engine, and APIs for programmatic access. WoRMS also supports UN Ocean Decade goals, data rescue for legacy databases, and standardization across platforms [19]. Challenges remain, including filling taxonomic gaps, verifying publication metadata, georeferencing type localities, expanding images, and sustaining community contributions [19]. Despite these, WoRMS serves as a global taxonomic reference, for marine biodiversity and modeling initiatives, improving data reliability, and supporting large-scale ecological analyses [15].

2.11 Catalogue of Life

The Catalogue of Life (COL) is a comprehensive, authoritative index of known species, established in 2001 through a partnership between Species 2000 and ITIS [15, 26]. As of 2022, it contained over two million accepted species names, compiled from taxonomic experts worldwide. COL integrates Global Species Databases (GSDs), each curated by specialists, and serves as a central taxonomic backbone for biodiversity research, environmental monitoring, and regulatory frameworks [15, 26]. Species 2000 (1996) and ITIS (Integrated Taxonomic Information System 1996) laid the foundation for COL, providing standardized species indices and hierarchical classifications [15]. The 2020 launch of ChecklistBank, developed with GBIF and Naturalis, introduced automation, stable identifiers, and modern editorial tools [15, 26]. COL underpins platforms such as GBIF, iNaturalist, EoL, GenBank, and BHL (Biodiversity Heritage Library), and supports le-

gal instruments including CITES and the IUCN (International Union for Conservation of Nature) Red List [15,26]. Species entries include persistent identifiers (DOIs , LSIDs), enabling integration with BOLD (Barcode of Life Data System), UNITE, FishBase, and SeaLifeBase, as well as services for name matching, checklist comparison, and taxonomic validation [25,26]. Key challenges include incomplete coverage of hyperdiverse taxa, dynamic and subjective taxonomic concepts, limited incentives for contributors, legacy data, and inconsistent identifier assignment [15,25,26]. Despite these, COL remains a cornerstone of global biodiversity information, providing standardized, expert-curated species data for scientific and conservation applications.

2.12 Pl@ntNet

Pl@ntNet is an image-based plant identification platform available on Android, iOS, and the web, designed to facilitate botanical data collection through participatory sensing [8, 29, 30]. Often called the "Shazam of plants" [29, 31], it allows non-experts to contribute plant observations via photographs. Launched in 2010 as Pl@ntScan, it evolved through Pl@ntNet-ID (2011) and Pl@ntNet-Identify (2012), progressively supporting multiple plant organs (leaves, flowers, fruits, bark, whole plant) and social-network integration [30]. Mobile apps launched in 2013 (iOS) and 2014 (Android) expanded species coverage and introduced multi-image and metadata-based identification strategies. By 2017, Pl@ntNet supported 10,000 species with over 332,000 training images and had been downloaded more than 3 million times in 170+ countries [30]. The system processes user-submitted images using Convolutional Neural Networks (CNNs) trained on validated observations, combining multi-view predictions with geolocation-based species checklists [29]. Observations are stored in a CouchDB NoSQL database, and a hashing-based similarity search allows retrieval of visually similar images [8, 29]. Multi-organ submissions improve accuracy, while data validation relies on a collaborative workflow where community members and tools like IdentiPlante and ThePlantGame confirm observations before inclusion in the training index [8, 29, 30]. Limitations include low sharing rates, uneven image quantity per observation, spatial biases, and dependency on a small expert group for validation [30]. Despite these, Pl@ntNet exemplifies the integration of AI, citizen science, and botanical research, providing a scalable model for plant biodiversity monitoring and public engagement [8, 29, 30, 32].

2.13 Key Takeaways

The reviewed system demonstrates several approaches to biodiversity data management, each with unique strengths and limitations. Common challenges were identified as usability issues, data quality problems, interoperability gaps, and features focused on non-domain experts makes this systems not suitable for professional use. But strengths like modular architectures with very detailed taxonomic management, and features on how to handle large datasets was also identified. These insights will help inform and design a new biodiversity data management system that is focused on usability, interoperability, and specifically on professional use. This way, the new system can utilize strengths of existing platforms while addressing their shortcomings to better meet the needs of biodiversity researchers and professionals.

3 Methodology

For the creation of such biodiversity data management system, the knowledge, experience, and domain expertise of biologists was essential. So stakeholders for the project were identified, these were the members of the Insect Collection of the University of Madeira, housed at the Faculty of Life Sciences. Withing this group three main roles were identified: the interns that conduct smaller research projects only adding new data to the collection, the researcher that also have research projects, but they also handle data that already exists in the collection, and the senior researcher/collection manager that basically are involved with almost all of the research projects of the collection and handle the collection data in a daily basis. Each of the roles has different permissions within the collection so it was important to understand their needs and requirements for the system. Meetings were held with some of the stakeholders to try to understand which were the main problems that they faced using their current workflows. The main problem reported was the decentralization of the data, because each member of the collection stored their data locally on their own devices using excel spreadsheets. This approach resulted in several recurring problems, like data loss when members left the department or no longer could access their files, and difficulty to retrieve data from the files due to inconsistent storage practices. Also the lack of standardization caused that some reports were incomplete or missing critical metadata, reducing both the reliability and reusability of the information. These problems caused loss of data integraty, time and resources as trying to locate missing files or correct incomplete records caused delays in ongoing research projects. After these first meetings it was defined that the system should be a centralized repository for all the data handled in the collection, so that data loss was less likely to happen, data entry standardization being enforced so that the time and resources spent managing the data are reduced. In the next meeting with the stakeholders the requirements for the system were refined and prioritized, to ensure that the proposed solution would address their needs.

3.1 Requirements

To ensure the development of a system that effectively meets user needs and project goals, a set of functional requirements were defined, throughout the conversation with the domain experts. These requirements were collected through meetings with stakeholders and refined throughout the planning phase. Functional requirements describe what the system should do, clearly defining what

functionalities and the behavior that the system should have. The following sections will describe the main functional requirements of the system.

3.1.1 Login and Registration

The Login and Registration model is responsible for managing user access to the platform. It ensures secure account creation and authentication while handling the initial assignment of the basic user role, wich could be attributed to an intern within the collection.

ID	Description	Priority
FR.1	The system shall require users to register with a username.	High
FR.2	The system shall require users to register with an email.	High
FR.3	The system shall require users to register with a password.	High
FR.4	The system shall require users to authenticate with email before accessing functionalities.	High
FR.5	The system shall require users to authenticate with password before accessing functionalities.	High
FR.6	The system shall provide a password reset mechanism for users who have forgotten their password.	Low
FR.7	The system shall allow users to log out at any time.	High
FR.8	The system shall automatically assign the "user" role to users upon registration.	High

Table 1: Functional Requirements (Authentication Features)

3.1.2 Dashboard

The Dashboard serves as the system's landing page, offering users a concise overview of essential information. It presents key metrics such as the total number of occurrences and projects, along with recent activity and the latest submitted reports. This centralized view enhances user experience by providing quick access to relevant and up-to-date data upon login.

ID	Description	Priority
FR.9	The system shall display a graph of daily occurrences on the user's dashboard.	Low
FR.10	The system shall display the latest reports on the user's dashboard.	Low

Table 2: Functional Requirements (Dashboard Features)

3.1.3 Nomenclature

The Nomenclature section is one of the core components of the system, dedicated to the management of taxonomic records. It enables users to create, update, and search for detailed species information. This module serves as the central repository for occurrence data registration, ensuring consistency and accessibility data for all users.

ID	Description	Priority
FR.11	The system shall allow users to create new nomenclature records.	High
FR.12	The system shall allow users to link multiple bibliographies to a single nomenclature.	High
FR.13	The system shall allow users to remove linked bibliographies from a nomenclature.	High
FR.14	The system shall require users to fill all necessary taxonomic fields (e.g., Kingdom, Family, Genus) when creating a nomenclature.	High
FR.15	The system shall allow users to import nomenclature data from an Excel file.	Medium
FR.16	The system shall allow users to import taxonomic information from an external API when creating nomenclature.	Low
FR.17	The system shall provide a search function for nomenclature records.	High
FR.18	The system shall display all occurrences linked to a nomenclature.	High

Table 3: Functional Requirements (Nomenclature Management)

3.1.4 Bibliography

The Bibliography section complements the Nomenclature module by serving as the repository for scientific literature that supports and validates taxonomic names. It allows users to manage bibliographic references, import citation data, and efficiently search or filter through existing records. This section is essential for maintaining the scientific rigor of the nomenclature system, as it ensures that each taxonomic entry is grounded in verified literature and aligned with accepted standards of classification.

ID	Description	Priority
FR.19	The system shall allow users to create new bibliography records.	High
FR.20	The system shall require users to provide all necessary information when creating a bibliography (e.g., title, authors, publication year).	High
FR.21	The system shall allow users to import bibliography data from an Excel file.	Medium
FR.22	The system shall display detailed information for each bibliography.	High
FR.23	The system shall allow users to filter bibliographies by title.	Low
FR.24	The system shall allow users to filter bibliographies by author.	Low
FR.25	The system shall allow users to filter bibliographies by publication title.	Low
FR.26	The system shall allow users to filter bibliographies by publication year.	Low
FR.27	The system shall display a list of all bibliographies.	High

Table 4: Functional Requirements (Bibliography Management)

3.1.5 Occurrences

The Occurrences section enables users to register, manage, and export detailed records of species sightings or field data. Each occurrence is linked to a specific taxonomic entry from the Nomenclature module, ensuring scientific consistency. Users can associate occurrences with ongoing projects, attach relevant files such as images or field notes, and ensure standardized data entry.

ID	Description	Priority
FR.28	The system shall allow users to create new occurrence records.	High
FR.29	The system shall require users to provide all necessary information for an occurrence.	High
FR.30	The system shall allow users to link a project to an occurrence.	High
FR.31	The system shall allow users to link a nomenclature to an occurrence.	High
FR.32	The system shall allow users to attach files (e.g., images, PDFs) to an occurrence.	High
FR.33	The system shall allow users to filter occurrences by scientific name.	Low
FR.34	The system shall allow users to filter occurrences by locality.	Low
FR.35	The system shall allow users to filter occurrences by date.	Low
FR.36	The system shall display detailed information for each occurrence.	High
FR.37	The system shall allow users to download files linked to an occurrence.	High
FR.38	The system shall allow users to export occurrences in Darwin Core (CSV) individually.	Low
FR.39	The system shall allow users to export occurrences in Excel format individually.	High
FR.40	The system shall allow users to export occurrences in Excel format in bulk.	Low
FR.41	The system shall allow users to edit occurrences they have created.	Low

Table 5: Functional Requirements (Occurrence Management)

3.1.6 Projects

The Projects section allows users to create, manage, and occurences within the system. Although not directly related to biodiversity data itself, this module plays a key role in structuring occurrences by linking them to specific projects. It imporves traceability, enables users to retrieve occurrences by project, and supports better data management across different research efforts.

ID	Description	Priority
FR.42	The system shall allow users to create new project records.	High
FR.43	The system shall require users to provide all necessary information for a project (e.g., title, research type, advisor).	High
FR.44	The system shall display detailed information for each project.	High
FR.45	The system shall display all occurrences linked to a project.	High
FR.46	The system shall allow users to filter projects by title.	Low
FR.47	The system shall allow users to filter projects by advisor name.	Low
FR.48	The system shall allow users to filter projects by department name.	Low
FR.49	The system shall allow users to filter projects by project type.	Low

Table 6: Functional Requirements (Project Management)

3.1.7 Profile Page

The Profile section allows users to manage their personal account information, including updating their name, email, and password.

ID	Description	Priority
FR.50	The system shall allow users to change their email address.	Medium
FR.51	The system shall allow users to change their password.	Medium

Table 7: Functional Requirements (User Profile)

3.1.8 Manager and Admin

The Manager and Administrator roles offer advanced control over platform operations and data governance. Managers, witch could be attributed to a normal researcher within the collection, possess extended permissions beyond those of regular users, allowing them to delete records across the Nomenclature, Bibliography, Occurrences, and Projects modules, while still retaining all the capabilities of the basic user role. Administrators, witch are the senior researchers/collection manager, on the other hand, have full system-wide authority. In addition to all manager privileges, they can manage user accounts, assign or revoke roles, and customize the structure of occurrence records by modifying field configurations.

ID	Description	Priority
FR.52	The system shall allow users with the "manager" role to perform all actions available to the "user" role.	High
FR.53	The system shall allow managers to delete occurrences.	High
FR.54	The system shall allow managers to delete nomenclature.	High
FR.55	The system shall allow managers to delete bibliographies.	High
FR.56	The system shall allow managers to delete projects.	High
FR.57	The system shall allow administrators to perform all actions available to managers.	Low
FR.58	The system shall allow administrators to delete user accounts.	Low
FR.59	The system shall allow administrators to edit user roles.	Low
FR.60	The system shall allow administrators to filter users by username.	Low
FR.61	The system shall allow administrators to filter users by email.	Low
FR.62	The system shall allow administrators to filter users by role.	Low
FR.63	The system shall allow administrators create occurrence form fields.	High
FR.64	The system shall allow administrators edit occurrence form fields.	High
FR.65	The system shall allow administrators activate for occurrence form fields.	High
FR.66	The system shall allow administrators deactivate occurrence form fields.	High
FR.67	The system shall allow administrators set mandatory status to occurrence form fields.	High
FR.68	The system shall allow administrators set optional status to occurrence form fields.	High
FR.69	The system shall allow administrators to filter occurrence form fields by field name.	Low
FR.70	The system shall allow administrators to filter occurrence form fields by the status of the field (e.g., Mandatory, optional).	Low
FR.71	The system shall allow administrators to filter occurrence form fields by the status of the field (e.g., active, deactivated).	Low

Table 8: Functional Requirements (Roles and Administration)

3.2 Design Decisions

With the functional requirements defined the development process moved into the system design phase. In this phase a reference architecture, architecture pattern, and a design pattern were selected as overall guides how should the system should be structured and implemented.

3.2.1 Reference Architecture

As for the reference architecture, three alternative were considered: web application, desktop application with a thick client, and mobile application. Each option was evaluated and in the end the web application architecture was selected as it provides accessibility, maintainability, ease of development and deployment, and most importantly makes the creation of a very usable and

understandable platform possible. As most people are very familiar with the interaction with websites, making the learning curve of the system much smaller.

3.2.2 Architecture Pattern

Having established the reference architecture as web application, the next decision involved selecting between three different architecture patterns, Client-Server, Peer to Peer, and Microservices. The architectural pattern defines the high level organization of the system, such as how the components interact, communicate and are deployed. So considering that a web application was being developed, the Client-Server pattern was selected as the one to follow, as it is one of the most widely used with web applications. This pattern provides separation between data management and user interface, and ensures scalability and ease of maintainability [33, 34].

3.2.3 Design Pattern

With both reference architecture and architecture pattern defined, only a design pattern was missing. Three patterns were considered: Model-View-Controller (MVC), Model-View-ViewModel (MVVM), and Model-View-Presenter (MVP). Each one of these patterns offer different advantages and disadvantages, MVC separates the application into three components, the models that manage the data and business logic, the views responsible for the user interaction with the system, and the controllers that process the users inputs [35–38]. MVVM separates the user interface, presentation logic and domain data, where the model contains the core business logic and domain data, the view that renders the UI and handles inputs, and the view model that maintains the UI state and mediatesuser interactions [35, 39]. MVP tries to improve the separation of concerns by replacing the controller with the presenter that actively mediates between the view and model [35, 40]. After evaluating the three patterns, MVC was selected as the most suitable for the project, by providing clear separation of concerns between data and logic, and a modular development.

3.3 Technologies

With the design decisions made, the next phase of the project evolved to the selection of the technologies to be used in the implementation of the system. The system is structured into three main layers: the front-end, the back-end, and the database, each one being responsible for specific functionalities and tasks. Front-end being responsible for the user interface and client-side interactions, the back-end manages the core business logic, API endpoints, and server-side

processing, and the database is responsible for storing and managing the system's data. This separation of concerns aligns with the Client-Server architecture and the Model-View-Controller (MVC) pattern defined previously.

3.3.1 Front-end

The front-end of a web application refers to the interface that the user interacts with directly. It contains the design, structure, behavior, and content of a web application, so it plays a critical role in ensuring great usability, accessibility, and user experience. So four technologies were considered: React, Angular, Vue.js, and Svelte. React is an open-source library developed by Meta (previously Facebook) that implements a component-based architecture that promotes modularity, maintainability, and scalability [41–43]. Angular is a open-source framework developed by Google that also folllows component-based architecture and provides strong typing through TypeScript [44–47]. Vue.js is a progressive JavaScript framework that combines declarative templates and component-based [41–43]. Svelte is an open-source JavaScript framework that collects components into efficient imperative code at build time, instead of using a virtual DOM [48–50]. In the end React was selected due to its strengths of reusable components, that promotes an intuitive and responsive user interface, also the existence of a large community and ecosystem of tools and libraries that improve development speed and efficiency.

3.3.2 Back-end

The back-end is responsible for the server-side logic, database interactions, and API endpoints of a web application. Handling requests from the front-end, processing data, and managing business logic. So three technologies were considered: Laravel, Ruby on Rails and Express. Laravel is a open-source PHP framework that follows the MVC pattern that provides a robust set of tools already integrated into the framework [51–53]. Ruby on Rails is an open-source full-stack web framework that also follows the MVC pattern that offers convention over configuration [54–56]. Express is a minimalist Node.js framework that provides a lightweight core with middleware support [57, 58]. After evaluating each option Laravel was selected due to its implementation of MVC, robust features, modular architecture, maintainability, and strong community support.

3.3.3 Database

Databases provide persistent storage and manage data operations such as creation, modification, querying, and deletion. They are generally categorized as relational (SQL) or non-relational (NoSQL). Three database management systems were considered for this project: MySQL, MongoDB, and PostgreSQL. MySQL is a wydely-used open-source RDBMS (Relational Database Management System), that supports ACID (Atomicity, Consistency, Isolation, and Durability) compliance, replication, clustering, stored procedures, triggers, and indexing [59–61]. MongoDB is an open-source document-oriented NoSQL database that stores data in JSON-like documents with dynamic schemas, supports horizontal scaling via sharding, replication for high availability, and aggregation pipelines for advanced querying [62, 63]. PostgreSQL is an open-source, object-relational database system that supports advanced data types, ACID compliance, MVCC, stored procedures, triggers, window functions, and JSON/JSONB [64–66]. MySQL was selected due to its already integrated support in Laravel, robust relational model that´s more suited for hierarchical biodiversity data, and its great support and scalability.

3.4 System Architecture

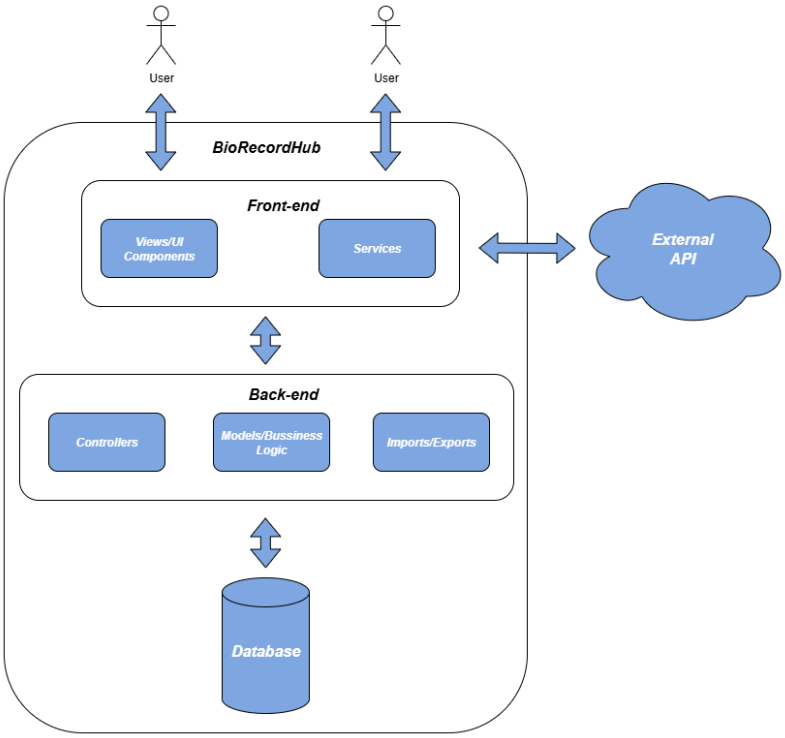


Figure. 1: High Level View of the system

The system architecture adopts a layered approach, structured into three domains: user interface, application logic, and data management. This separation of concerns aligns with the Client–Server architecture and the Model–View–Controller (MVC) pattern, also promotes modularity, maintainability, and scalability, enabling features to be extended or replaced with minimal impact on other components. A high-level overview is presented in Figure 1.

3.5 Figma Prototype

With the overall structure of the system defined, the next phase of the project involved the creation of a prototype to help visualize and understand how the stakeholders would like to interact with the system. So a Figma prototype was created to have a high-fidelity representation of the user interface and user experience, this prototype has created following the requirements previously defined and using feedback collected from several meetings with the stakeholders. The process started with the definition of a base theme and color palette that ensured visual consistency and was visually appealing to the users. So after discussing with the stakeholders a color palette of a primary color #468585, a secondary color #50B498, and white #FFFFFF for backgrounds and highlights was selected. Now the overall structure and layout of the system needed to be defined, the need of a simple UI that made usability high has defined with the creation of side menu that provided quick, easy navigation, a top bar that displayed the authenticated user, and on the right of the navigation menu, the relevant contextual information of the system would be displayed 2.

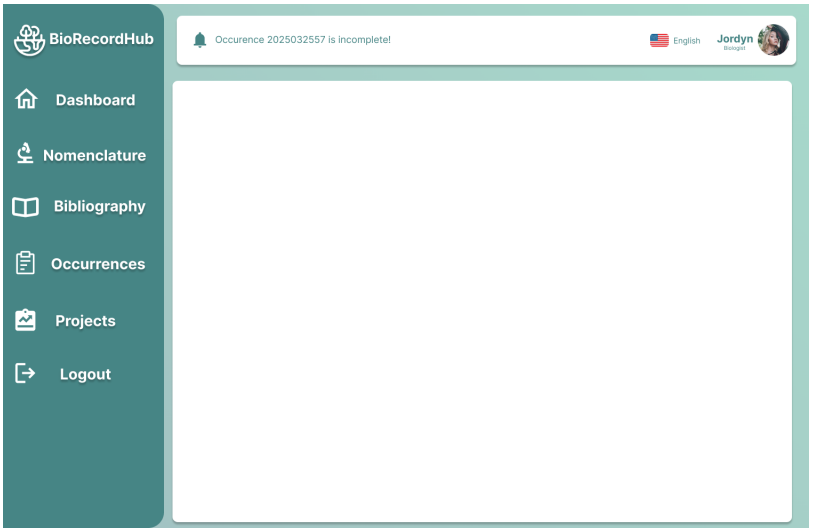


Figure. 2: Layout Prototype

Having defined the main points of the system design, the process evolved to the creation of the different pages, starting with the login and registration pages, that followed a simple form design for the best usability 3.

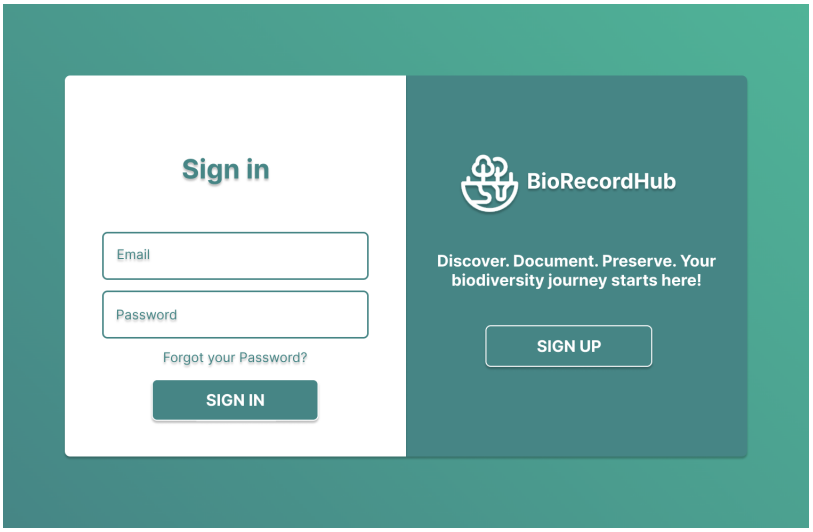
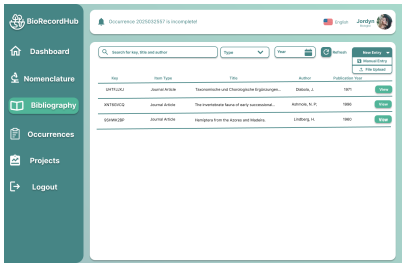


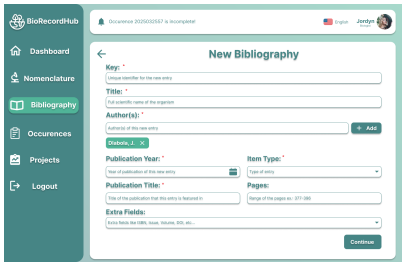
Figure. 3: Login Prototype

Then the pages for bibliographies and nomenclatures were created, as these pages were defined as the backbone of the system. Both pages followed the same structure, a multi-record view that displayed all the records 4a, a single view that displayed detailed information of a record 4b, a form view for creating and editing records 4c, and finally the import data page for bulk data creation using Excel files 4d.

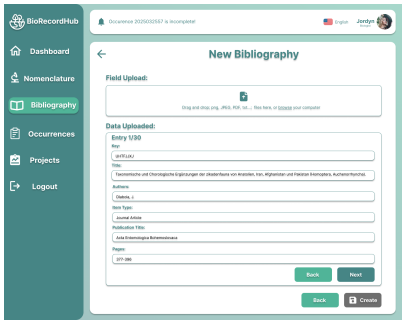


(a) Multi View Prototype

(b) Single View Prototype Bibliography



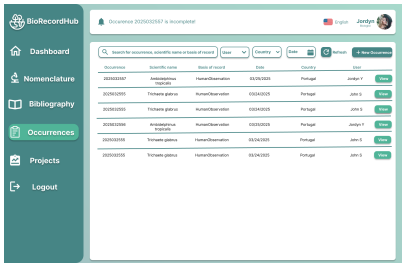
(c) Form Prototype



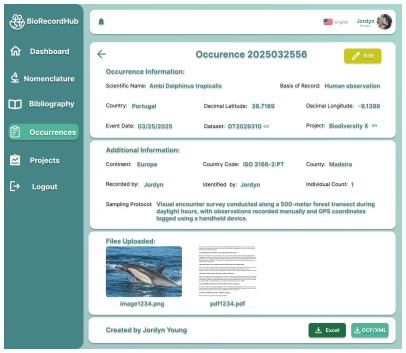
(d) Import Data Prototype Bibliography

Figure. 4: Bibliography Prototype.

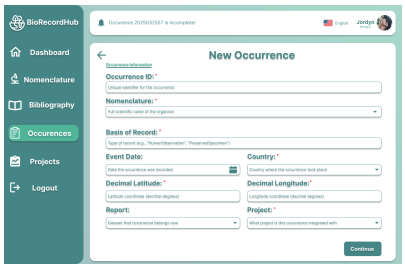
After these two core pages were created and approved by the stakeholders, the next phase evolved the creation of the occurrences, projects, and dashboard pages. The occurrences and projects pages followed the same structure as the nomenclature and bibliography pages with multi-record views 5a, single views 5b, and form views 5c. As for the dashboard page it was created as the landing page of the system where the user could visualize the most recent occurrences and how many occurrences have been added into the system 5d.



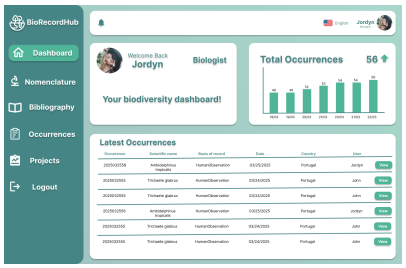
(a) Multi View Prototype Occurrences



(b) Single View Prototype Occurrences



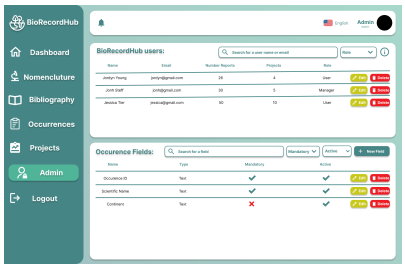
(c) Form Prototype Occurrences



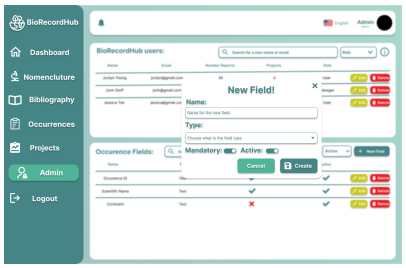
(d) Dashboard Prototype

Figure. 5: Occurrences and Dashboard Prototypes.

With the occurrences, projects, and dashboard pages approved by the stakeholders, the only section of the system missing a prototype was the admin page, where the administrators could manage users and occurrences fields. This page followed a simple design where both user management and occurrence fields management had a multi record view, where the administrator could verify, search, and interact with users or occurrence fields 6a, and pop up form for creating and managing data 6b.



(a) Prototype Admin



(b) Popup Prototype Admin

Figure. 6: Occurrences and Dashboard Prototypes.

3.6 Data Flow

With the structure and design of the system defined, the next step of the project involved mapping and defining how the data flow of the system should be for the main components. Data flows for the main functionalities of the system were created to help in the understanding of how the parts of the system should interact making the development process easier. The four main data flows created where for the creation/import of new data, the edition of existing data, the deletion of data, and the capability of exporting data from the system.

3.6.1 Create/Import data flow

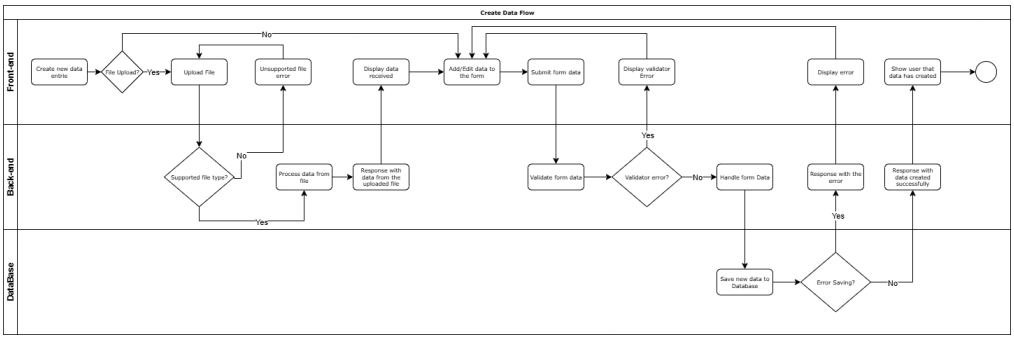
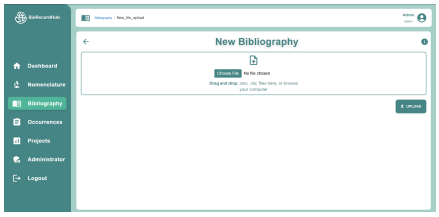
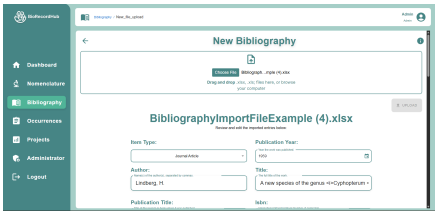


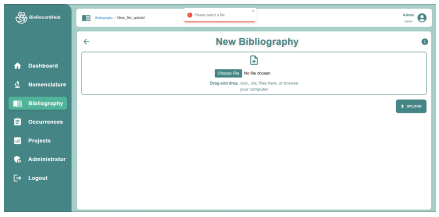
Figure. 7: Create Data Flow.



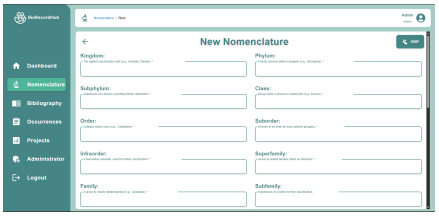
(a) File Upload



(b) File Uploaded Data



(c) File Upload Error



(d) Manual Form

Figure. 8: Create/Import Data.

Figure 7 illustrates the generic flow of creating new data inside the system. The process can be initiated in two ways: either by uploading an Excel file containing the data or by manually filling a form. If the user chooses to upload a file, they are redirected to the designated upload page 8a, where the file is submitted to the back-end. Then the back-end first validates whether the uploaded file is of a supported type (Excel). If the file type is not supported, a validation error is returned and displayed by the front-end 8c. If the file passes validation, it is processed and the data it contains is extracted. This data is then returned to the front-end, where it is displayed for user confirmation and possible modification before submission 8b. Alternatively, if the user decides to input the data manually, the front-end redirects them to the appropriate form page 8d. Here, the user provides the required information directly. Regardless of the data source—file upload or manual input—once the user submits the form through the defined API. Then the back-end controllers validate the received data to ensures that the business rules are applied, such as the presence of mandatory fields and correct data formatting. If validation errors are detected, the back-end generates an error response, which is displayed on the front-end for correction 8c. If validation succeeds, the controller delegates the operation to the relevant models. The database confirms the operation, either by storing the new record successfully or returning an error if a error occurs. This result is sent back to the back-end, which prepares the appropriate response. Finally, the front-end presents the response to the user as either a success message or an error notification.

3.6.2 Edit data flow

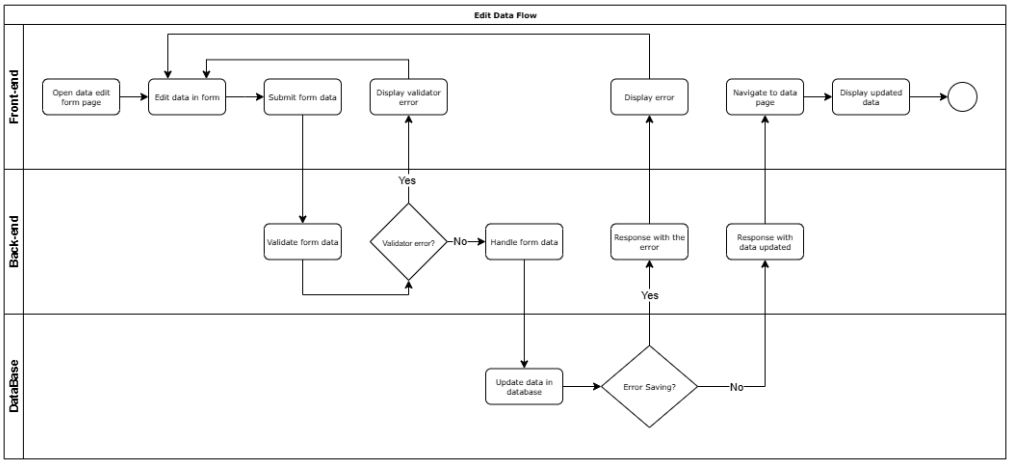
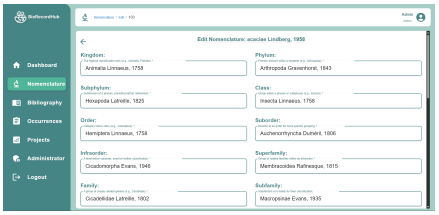


Figure. 9: Edit Data Flow.

(a) Edit Data



(b) Edit Form Data

Figure. 10: Edit Data Flow Prototype Images.

Figure (9) illustrates the generic flow of editing data within the system. The process begins when a user interacts with the front-end 10a, where the user is redirected to the form page 10b and can modify the necessary values. Once the changes are submitted, the front-end collects the updated input and sends it to the back-end through a request to the defined API. Then the controllers, after receiving the request, validate the data to ensure compliance with the business rules of the entity. If validation errors are detected, the back-end generates an error response, which is sent to the front-end and displayed to the user. This allows the user to make the necessary corrections before resubmitting. If validation is successful, the controller submits the operation to the relevant models. Then the database confirms the operation by either updating the record successfully or returning an error if a error occurs. This result is passed back to the back-end, which prepares the appropriate response. Finally, the front-end communicates the outcome to the user in the form of a success message or an error notification.

3.6.3 Delete data flow

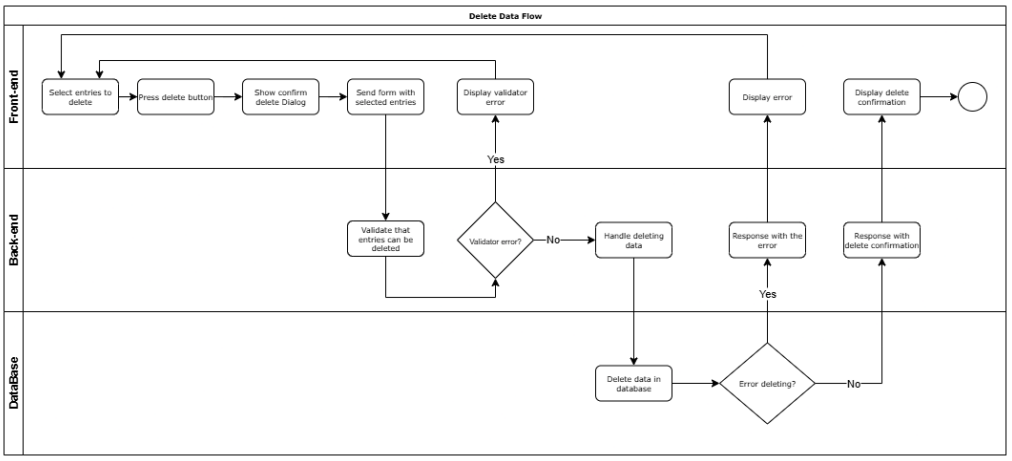
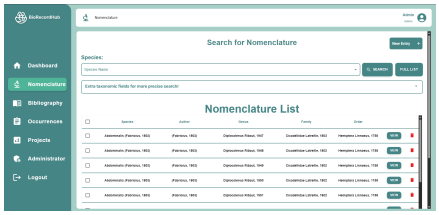
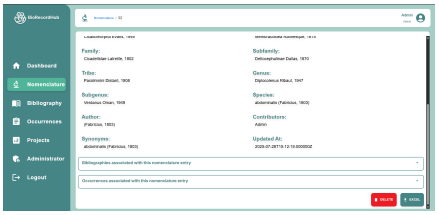


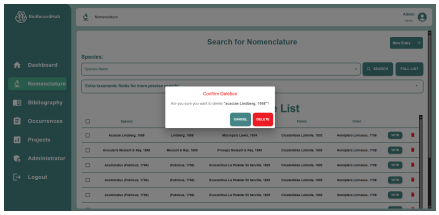
Figure. 11: Delete Data Flow.



(a) Data List



(b) Delete Data Button



(c) Confirm Deletion

Figure. 12: Delete Data Flow Prototype Images.

Figure 11 illustrates the generic flow of deleting data within the system. The process begins when a user interacts with the front-end to delete data (Fig.12a, Fig.12b). The front-end then prompts a confirmation dialog 12c to ensure that the user is fully aware of the action. Once confirmed, the front-end collects the identifiers of the selected entries and sends them to the back-end. When the back-end receives the requets, the controllers validate whether the selected entries are eligible for deletion, ensuring that they are not referenced by other records or bound by other constraints. If a validation error occurs, the back-end generates an error response, which is returned to the front-end and displayed to the user. This informs the user that the deletion cannot proceed. If validation is successful, the necessary models are used. Then the database confirms the operation by either deleting the record successfully or returning an error. The result is sent back to the back-end, which prepares the appropriate response. Finally, the front-end communicates the outcome to the user as either a success message or an error notification.

3.6.4 Export data flow

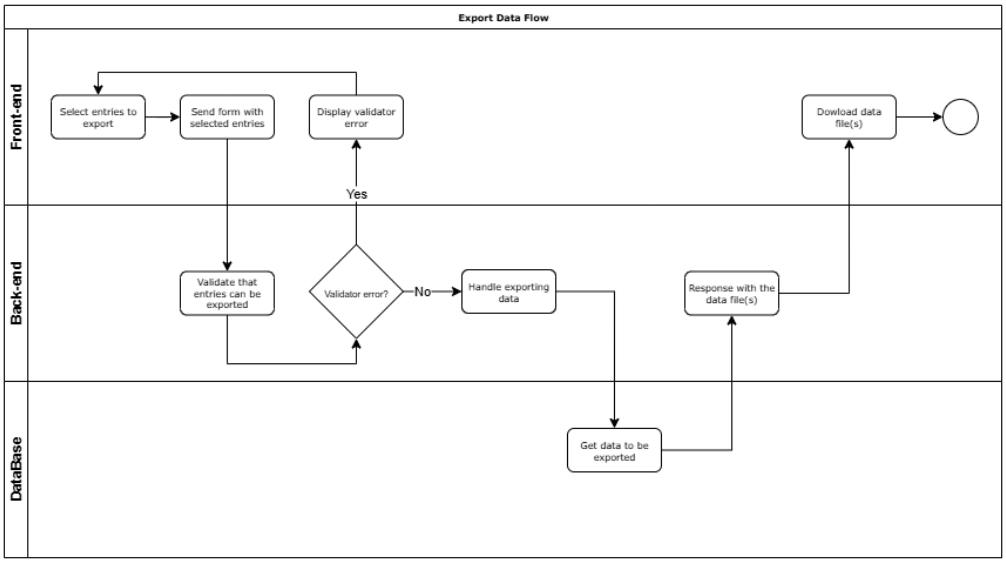
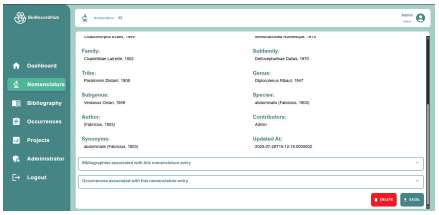


Figure. 13: Export Data Flow.

(a) Export Multiple Data



(b) Export Single Data

Figure. 14: Export Data Flow Prototype Images.

Figure 13 illustrates the generic flow of exporting data within the system. The process begins when a user interacts with the front-end to export data. In multiple-entry views, the user selects one or more records and presses the export button 14a. The front-end collects the identifiers of the selected entries and sends them to the back-end through a request to the defined API. For single-entry views, the process is the same, except no selection step is required 14b. When the back-end receives the request, the controllers will validate whether the selected entries are eligible for export. If validation fails, the back-end generates an error response, which is returned to the front-end and displayed to the user, indicating that the export cannot be performed. If validation is successful, the controller calls the relevant models, that make the data proceed to the retrieval phase. The database retrieves the relevant records and returns them to the back-end. The back-

end then generates the export files in the appropriate format and prepares the response with the files attached. Finally, the front-end receives the response and prompts the user with a download interface, allowing them to save the files to their desired location.

4 Development Process

The development of the system followed an agile method approach, where the development process was divided in several phases. Each phase focused on specific parts of the system, allowing for an iterative process, where the stakeholders could provide feedback and adjustments so that the solution evolved according to their expectations and needs. The development process can be divided into 6 major phases, with work being made between the three main parts of the system, the database, the back-end, and the front-end: the creation of the main tables of the database, then the creation of front-end and back-end modules responsible for the authentication and user management. Followed by the modules responsible for the management of bibliographies and nomenclature as they are the backbone of the system, then the modules for the administrator pages because the occurrence fields are need for the creation of occurrences. Following up with the creation of the modules for the management of projects then occurrences. Finaly the modules responsible for the dashboard and data export were created. Between each phase, meetings with stakeholders were held to confirm that the system was working as expected and get feedback if anything needed to be changed or added. The sections will explain in deatil how each main part of the system is implemented, their modeules and their functionalities. The complete source code and implementation details are available in the project's GitHub repositories [67, 68].

4.1 Database

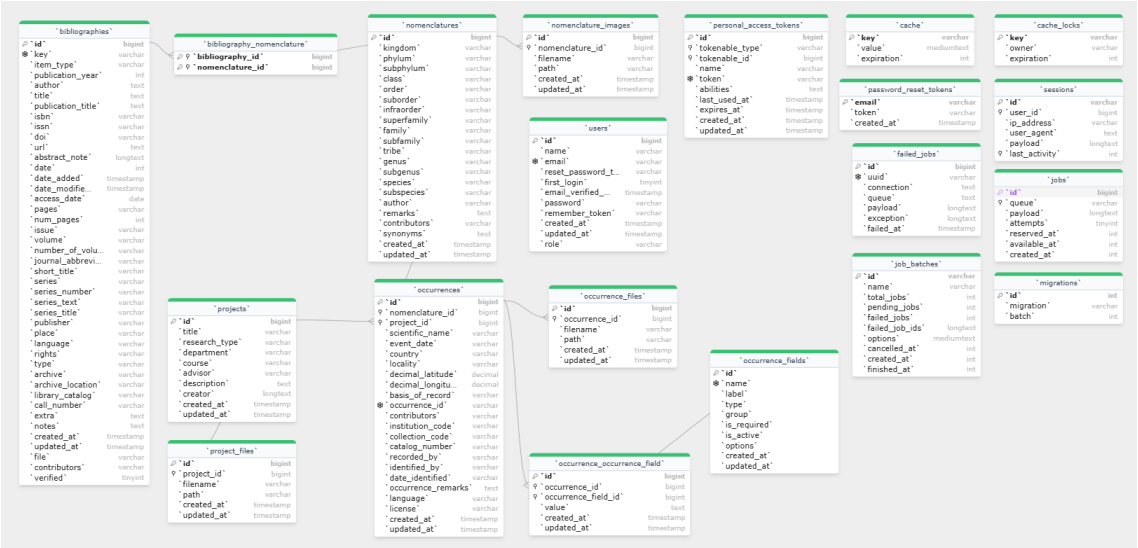


Figure. 15: Database ERD.

4.1.1 Database Design Rationale

The database schema (Figure 15) was designed to effectively handle the complexity of the biodiversity data, by keeping referential integrity and adaptability between the different entities. A key design decision was the separation of bibliographies and nomenclatures into independent entities connected through a many-to-many relation table called bibliography_nomenclature. This approach prevents redundancy and represents the reality as one nomenclature can be support by multiple bibliographies and a single bibliography can support multiple nomenclatures. Occurrences implementation also displays this design principle, as this module is connected with 3 other modules: nomenclatures, projects, and occurrence_fields. This one-to-many relation was done because an occurrence needs to have a nomenclature associated with so that the taxonomic information of the species is linked to the occurrence and nomenclature can have multiple occurrences associated with. The project relation follows the same logic, as the nomenclature relation, as an occurrence needs to be linked to a project that a better management of the occurrences can be done. Finaly the many-to-many relation with occurrence_fields was done so that the system can adapt to the necessary fields that each occurrence may need, as depending on the type of occurrence different fields may be necessary. This approach to the design of the database allows for a flexible and scalable system that can handle the always evolving needs of biodiversity systems. Data integrity and consistency were ensured by the utilization of primary and foreign keys, normal procedure in relational databases. As these mechanisms enforce integrity between related tables, preventing data from becoming inconsistent and incomplete. Other rules like ON DELETE CASCADE make sure that when records are deleted information that is linked between each other are also deleted, once again preventing inconsistencies in the data. Security considerations were also implemented as some tables are responsible for the users authentication and permissions, making sure that only authorized users can access or handle data. In the next section, each table will be described in detail, highlighting its role within the database and the rationale behind its design.

4.1.2 Bibliographies

Table: `bibliographies`

Purpose: Stores bibliographic references such as journal articles, books, or reports.

Key Fields:

- `id`: Unique identifier.

- `key`: Unique reference key created by the system.
- `author`, `title`, `publication_year`, `journal_abbreviation`, etc.
- `verified`: Indicates if the entry is verified.

Relations: Many-to-Many with `nomenclatures` via `bibliography_nomenclature`. Each bibliography can reference multiple taxonomic entities.

4.1.3 Nomenclatures

Table: `nomenclatures`

Purpose: Stores taxonomic hierarchy and species data.

Key Fields:

- Taxonomic levels: `kingdom`, `phylum`, `class`, `order`, `family`, `genus`, `species`.
- `author`: Person who described the taxon.
- `remarks`, `synonyms`, `contributors`.

Relations:

- Many-to-Many with `bibliographies` through `bibliography_nomenclature`, each nomenclature can reference multiple bibliographies.
- One-to-Many with `nomenclature_images`.
- One-to-Many with `occurrences`.

4.1.4 Occurrences

Table: `occurrences`

Purpose: Stores individual species observations or records.

Key Fields:

- `occurrence_id`, `scientific_name`, `event_date`, `country`, `locality`.
- Geographical coordinates: `decimal_latitude`, `decimal_longitude`.
- `basis_of_record`, `recorded_by`, `identified_by`.

Relations:

- Belongs to `nomenclatures` via `nomenclature_id`.

- Belongs to `projects` via `project_id`.
- One-to-Many with `occurrence_files`.
- Many-to-Many with `occurrence_fields` through `occurrence_occurrence_field`.

4.1.5 Projects

Table: `projects`

Purpose: Groups occurrences under research projects.

Key Fields: `title`, `research_type`, `department`, `course`, `advisor`, `description`, `creator`.

Relations:

- One-to-Many with `occurrences`.
- One-to-Many with `project_files`.

4.1.6 Occurrence Fields

Table: `occurrence_fields`

Purpose: Defines dynamic/custom fields for occurrences.

Relations: Many-to-Many with `occurrences` via `occurrence_occurrence_field`.

4.1.7 Users and Access

Table: `users`

Purpose: Stores user information for authentication and roles.

Key Fields: `name`, `email`, `password`, `role`.

Relations: Users create or contribute to projects, occurrences, and bibliographies.

4.1.8 Auxiliary Tables

In addition to the core entities, several auxiliary tables were introduced to extend functionality and manage specific relationships within the database:

- `nomenclature_images`: Stores the file paths of images associated with a given nomenclature entry, enabling visual documentation of taxonomic records.
- `project_files`: Stores the file paths of documents or resources linked to projects, supporting the management of supplementary project data.

- `occurrence_files`: Stores the file paths of files related to occurrences, such as specimen images, field notes, or environmental data.
- `bibliography_nomenclature`: Implements the Many-to-Many relationship between bibliographies and nomenclatures by storing their respective unique identifiers.
- `occurrence_occurrence_field`: Manages dynamic occurrence attributes by linking non-mandatory occurrence fields to their corresponding values, ensuring schema flexibility.

4.1.9 System Tables

- `jobs, failed_jobs, job_batches`: Background task management.
- `cache, cache_locks, sessions`: Caching, locking, and session handling.
- `migrations`: Tracks database schema changes.

4.1.10 Summary of Relationships

- Bibliographies <-> Nomenclatures: Many-to-Many (bibliography_nomenclature)
- Nomenclatures -> Nomenclature Images: One-to-Many
- Nomenclatures -> Occurrences: One-to-Many
- Projects -> Occurrences: One-to-Many
- Occurrences <-> Occurrence Fields: Many-to-Many (occurrence_occurrence_field)
- Projects -> Project Files: One-to-Many
- Occurrences -> Occurrence Files: One-to-Many

4.1.11 Key Takeaways

Creating a database that could handle complex biodiversity data with integrity, scalability, and interoperability was a significant challenge, because if these relations and tables were not well defined the system should be unusable and completely inefficient. Validating and enforcing certain rules on the data was essential to maintain high-quality records, also making sure that fast queries and exports were possible. Making sure that the database was flexible so that could adapt to the continues evolution of the biodiversity data was also a key point of focus. Considering all the needs

and requirements of the system for the database, it's possible to say that the database design and implementation was successful.

4.2 Back-end

As defined before the back-end of the system was developed using Laravel, a PHP framework that provides several tools already built in to improve development speed and implements the Model-View-Controller pattern. For this project two packages were utilized to help with specific functionalities. For authentication and user management, Laravel Breeze was used, this starter kit provides a simple and secure implementation of login, registration, and session handling functions. To address some necessities involving the different types of users with different sets of permissions, the table users that the Breeze package creates was modified with the addition of two new columns: role and first_login. Where the role is responsible for saving the type of user, meaning what permissions that user has, and the first_login column is used to identify if the user is logging in for the first time. This information is used in the front-end to know if is necessary to show a popup window with relevant information for a first time user. The second package used was the maatwebsite/excel package, this one is responsible for the handling of excel files within the system, meaning that this package is used to import data from excel files and export data to excel files. Also the architecture of the back-end follows the principle of modularity and separation of concerns. This means that each model is responsible for handling specific data or requests within the system, making the implementation easier to maintain and extend in the future.

4.2.1 Main Modules

Within the back-end, modules for Bibliographies, Nomenclatures, Occurrences, Occurrences Fields, and Projects were implemented as the main modules of the system. Each module is responsible for handling data related to each entity, where they operate within the business rules related and enforced to that entity.

For bibliographies, a request to create or update a bibliography, the following fields are mandatory: item_type, publication_year, author, title, and publication_title. Additionally any file that is attached to the a bibliography must have the PDF format. This restrictions were implemented to ensure standardization and consistency within the bibliography entries.

Nomenclature records follow similar rules as the bibliographic records, as some fields are required to maintain data consistency and integrity. The mandatory fields are: kingdom, phylum, subphylum, class, order, suborder, infraorder, superfamily, family, author, and species. In addition to the mandatory fields any nomenclature record must be linked to a minimum of one bibliographic record, making sure that every nomenclature representation is backed by a accepted and published reference.

As for nomenclature fields each record must contain the following attributes: name, label, type (which is restricted to text, textarea, select, number or date), and group. By default the parameter is_required is set to false while is_active is set to true. These rules ensure that occurence fields can be flexible and adaptable to any type of occurrence, making sure that the system can evolve with biodiversity data.

Occurrences as the entity within the system that can contain the most amount of data, it must follow several rules, so that their integraty and consistency is maintained. This means that every occurrence must be linked to a valid nomenclature and project. In addition to this the following fields are mandatory: scientific_name, event_date, contributors, institution_code, collection_code, catalog_number, recorded_by, identified_by, date_identified, occurrence_remarks, language and license. Files that are attached to occurrences can have the following formats: PDF, Excel or image formats. Also if necessary occurrences can contain an array of occurrence_fields, which link back to the dynamically defined fields, making sure that the system can adapt to different biodiversity research.

Finally projects are responsible for grouping occurrences together, so that contextual information can be provided for the biodiversity studies. This means that every occurrence can be linked to a project or a studied, so for these reasons every project must contain a title and a research_type. Any file that is attached to a project must have the PDF format, this restriction was implemented to ensure consistency and long-term readability of the files.

4.2.2 Back-end structure

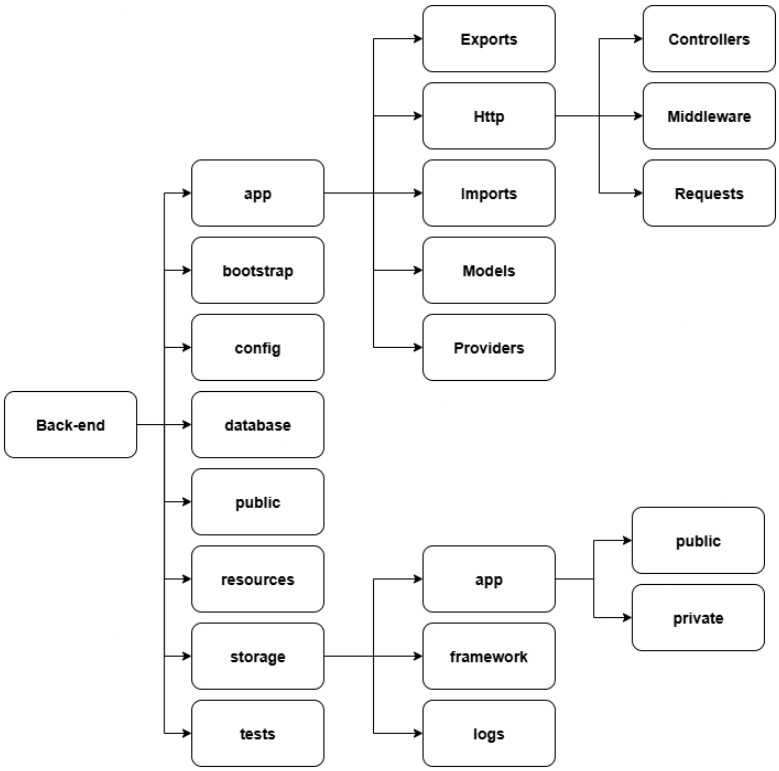


Figure. 16: Back-end tree diagram.

The figure (16) represents the structure of the system back-end. In the next section, each user-relevant module will be described in detail.

4.2.2.1 app/Exports

The Exports module is responsible for handling the system's data export functionality (Figure: 17). Its primary purpose is to enable users to generate and download biodiversity data in Excel files, improving data sharing, reporting, and integration with other research tools. The maatwebsite/excel package is based on the concept of export classes which can handle both the dataset to be exported and the configuration needed to transform it into the correct representation. These classes typically implement specific interfaces (called "concerns" in the package), such as FromCollection, WithHeadings, or WithMapping, each defining a different aspect of the export process. For example, FromCollection specifies the data source, WithHeadings defines the column headers, and WithMapping that allows data to changed before is written to the file. When a user initiates an export, the system uses the appropriate export class, which retrieves the requested data from

the database and prepares it for formatting. The package then processes this data row by row, applying mappings and headings as defined, before generating a downloadable spreadsheet and sending it back to the user.

```php
class ProjectExport implements FromCollection, WithHeadings, WithMapping
{
    2 usages
    protected $projects;

    1 usage  Sebastian *
    public function __construct(array $ids)
    {
        $this->projects = Project::with( relations: 'occurrences')
            ->whereIn( column: 'id', $ids)
            ->get();
    }

    no usages  Sebastian
    public function collection()
    {
        return $this->projects;
    }

    no usages  Sebastian
    public function map($project): array
    {
        return [
            $project->id,
            $project->title,
            $project->research_type,
            $project->department,
            $project->course,
            $project->advisor,
            $project->description,
            $project->creator,
            $project->occurrences->pluck('occurrence_id')->join(', '),
        ];
    }

    no usages  Sebastian
    public function headings(): array
    {
        return [
            'ID',
            'Title',
            'Research Type',
            'Department',
            'Course',
            'Advisor',
            'Description',
            'Creator',
            'Occurrence IDs',
        ];
    }
}
```

Figure. 17: Example of the Export function using the Maatwebsite/Excel package.

4.2.2.2 app/Http/Controllers

The Controllers module is responsible for handling the requests received from the front-end. Its primary purpose is to handle different types of HTTP requests, such as GET requests for retrieving data or POST requests for storing new information. Each controller defines functions (Figure: 18) corresponding to the various request types, ensuring that the back-end can process incoming requests, and business rules are enforced.

```php
public function store(StoreNomenclatureRequest $request)
{
    $validated = $request->validated();

    $bibliographyIds = $validated['bibliographies'] ?? [];
    unset($validated['bibliographies']);

    $imageFiles = $validated['images'] ?? null;
    unset($validated['images']);

    $nomenclature = Nomenclature::create($validated);

    if (!empty($bibliographyIds)) {
        $nomenclature->bibliographies()->sync($bibliographyIds);
    }

    $folderPath = storage_path( path: "app/public/nomenclature_images/{$nomenclature->id}");
    if (!file_exists($folderPath)) {
        mkdir($folderPath, permissions: 0755, recursive: true);
    }

    if ($request->hasFile( key: 'images')) {
        foreach ($request->file( key: 'images') as $image) {
            $filename = 'nomenclature' . $nomenclature->id . '_' . uniqid() . '.jpg';
            $path = "{$folderPath}/{$filename}";

            $image->move($folderPath, $filename);

            $nomenclature->images()->create([
                'filename' => $filename,
                'path' => $path,
            ]);
        }
    }

    return response()->json($nomenclature->load('bibliographies', 'images'), status: 201);
}
```

Figure. 18: Example of a Controller function in the back-end.

4.2.2.3 app/Http/MiddleWare

The Middleware module is used to ensure that only users with verified email addresses can access certain routes by checking if the user is authenticated and has verified their email. If not, it returns a JSON message with a 409 status code. If the user is verified, the request continues to the next step. This ensures that unverified users cannot access protected parts of the system, improving security and data integrity.

4.2.2.4 app/Http/Requests

The Requests module is responsible for validating incoming data. Each request class defines rules (Figure: 19) that the data must follow, such as required fields, data types, or specific formats. Controllers call these request classes to ensure that only valid and correctly formatted data is processed, enforcing business rules.

```php
public function rules(): array
{
    return [
        'title' => 'required|string|max:255',
        'research_type' => 'required|string|max:255',
        'department' => 'nullable|string|max:255',
        'course' => 'nullable|string|max:255',
        'advisor' => 'nullable|string|max:255',
        'description' => 'nullable|string',
        'creator' => 'nullable|string',
        'files.*' => 'file|mimes:pdf|max:10240',
    ];
}
```

Figure. 19: Example of a Request class in the back-end.

4.2.2.5 app/Imports

The Imports module is responsible for processing spreadsheet files uploaded by users (Figure: 20) and retrive their data. Import functionality is included in the dedicated import classes which define how rows from a file are handled and converted into data. Depending on the use case, different types can be applied: for example, ToCollection provides raw rows for custom processing, ToModel maps rows directly into database records, and WithHeadingRow allows headers to be matched automatically with attributes in the schema. When a file is uploaded, the appropriate import class is called, then the spreadsheet is procced and its data retrived. This process can include matching headers, applying default values for missing fields, and enforcing validation rules to ensure consistency. After the the files are proceeded, the data is sent to the user, so that they can review and makes corrections if necessary before saving it to the database.

```php
class BibliographyImport implements ToCollection
{
    2 usages
    protected $mappedData;

    no usages  Sebastian
    public function collection(Collection $rows)
    {
        $headers = [
            'item_type', 'publication_year', 'author', 'title',
            'publication_title', 'isbn', 'issn', 'doi', 'url',
            'abstract_note', 'date', 'access_date',
            'pages', 'num_pages', 'issue', 'volume', 'number_of_volumes',
            'journal_abbreviation', 'short_title', 'series', 'series_number',
            'series_text', 'series_title', 'publisher', 'place', 'language',
            'rights', 'type', 'archive', 'archive_location', 'library_catalog',
            'call_number', 'extra', 'notes','verified',
        ];

        $this->mappedData = $rows->skip( count: 1)->map(function ($row) use ($headers) {
            $rowArray = $row->toArray();

            if (count($rowArray) !== count($headers)) {
                return null;
            }

            $combined = array_combine($headers, $rowArray);

            if (is_null($combined['verified'])) {
                $combined['verified'] = false;
            }

            return $combined;
        })->filter()->values();
    }

    1 usage  Sebastian
    public function getMappedData()
    {
        return $this->mappedData;
    }
}
```

Figure. 20: Example of an Import function using the Maatwebsite/Excel package.

4.2.2.6 app/Models

Each Model module (Figure: 21) corresponds to a database table that defines its attributes, relationships, and any business logic associated with that entity. Models are responsible for querying the database, creating, updating, and deleting records. They also define relationships such as one-to-many or many-to-many, ensuring that related data can be accessed and manipulated easily. Controllers call these model methods to handle data, enforce business rules, and maintain data integrity.

```php
class Project extends Model
{
    no usages
    protected $fillable = [
        'title',
        'research_type',
        'department',
        'course',
        'advisor',
        'description',
        'creator',
    ];

    no usages  Sebastian
    public function files()
    {
        return $this->hasMany( related: ProjectFile::class);
    }

     Sebastian
    public function occurrences()
    {
        return $this->hasMany( related: Occurrence::class);
    }
}
```

Figure. 21: Example of a Model class in the back-end.

4.2.2.7 config

The config module contains configuration files that define the behavior and settings of the Laravel application. These files are used for parameters such as database connections, mail server settings, authentication options, caching mechanisms, and file storage paths.

4.2.2.8 database

The database module is responsible for managing the structure and initial testing data of the application's database. It contains three main components: factories, migrations, and seeders. Where the factories are used to generate fake data for testing, migrations define the database shema, and seeders populate the database with testing data.

4.2.2.9 routes

The routes module defines how the back-end responds to incoming HTTP requests. Each route specifies (Figure: 22) the URL endpoint, the request type (e.g., GET, POST, PUT, DELETE), and the corresponding controller function that should be executed. In this way, routes ensure that requests are directed to the correct part of the application logic. To improve maintainability and clarity, the project's routes were separated into multiple files, each corresponding to a specific model, such as nomenclature, bibliography, or occurrence.

```
Route::middleware( middleware: 'auth:sanctum')->prefix( prefix: 'project')->group(function () {
    Route::get( uri: '/getAutoComplete', [ProjectController::class, 'getProjectsAutoComplete']);
    Route::get( uri: '/', [ProjectController::class, 'index']);
    Route::post( uri: '/', [ProjectController::class, 'store']);
    Route::put( uri: '/{id}', [ProjectController::class, 'update']);
    Route::delete( uri: '/{projectId}/file/{fileId}', [ProjectController::class, 'destroyFile']);
    Route::delete( uri: '/{id}', [ProjectController::class, 'destroy']);
    Route::get( uri: '/{id}', [ProjectController::class, 'show']);
});
```

Figure. 22: Example of a Route definition in the back-end.

4.2.2.10 storage

The storage module is responsible for managing filesused by the back-end. This directory is further divided into two subfolders: storage/app/private and storage/app/public, the private folder cannot be accessed directly from outside the system, ensuring security, and public folder is intended for resources that need to be available to the users, such as images and documents related to nomenclatures, occurrences, bibliographies, and projects. This is done by executing the command: php artisan storage:link, which creates a link from public/storage to storage/app/public. The storage/logs directory contains the application log files where Laravel records errors, warnings, and system events.

4.2.3 Key Takeaways

The development of the back-end followed the best approach by applying the necessary decisons so that the MVC pattern was followed and create a system that is modular, secure, and scalable. By using already existing packages like Laravel Breeze and maatwebsite/excel the development speed was improved and the system could focus on the specific needs of biodiversity data management. The capabilitie of handling excel files was the most challenging part of the implementation of the back-end as a lot of data processing and validation is needed to handle the complex biodiversity data. Some constrains and requirements emerge from this implementation as the version of PHP must be 8 or higher and three PHP extensions must be enabled: php-zip, php-xml, and php-gd as they are necessary for the maatwebsite/excel package. The complete implementation and source code are publicly available in the project's GitHub repository [67].

4.3 Front-end

Built using React and Typescript, the front-end provides the interface that the user interacts with when using the system. As in the back-end some packages and libraries were used to improve and speed up development, these were Material UI (MUI), MUI Icons, React Router DOM, Axios, React Toastify, and Chart.js 2. Material UI was used as the main component library, as it provides a wide variety of finished and customizable components, this way consistency can be maintained throughout the system. MUI Icons as the name indicates was used as the main library for icons within the system, making the development easier as it improves better organization and consistency. React Router DOM as responsible for handling the navigation between the different pages of the front-end without the need to reload the page it self. Axios had the role of communicating with the back-end API, by handling the HTTP requests and responses. React Toastify was used to provide ready to go notifications to the user in the case of errors or successful operations 23. Finally Chart.js 2 was used to create the chart present in the system, this way allowing for a more dynamic data visualization. As expected from a React application, the front-end follows a component based architecture, where each component is responsible for a specific function within the system. This approach allows for a better organization of code, as components can be reused throughout the system, improving maintainability, scalability, and interfaces consistency.

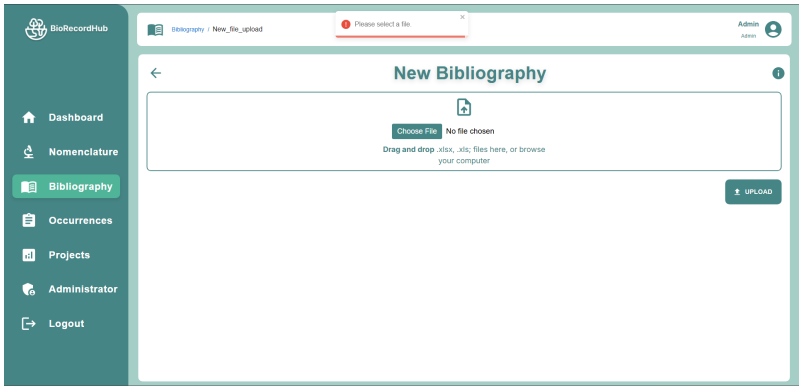
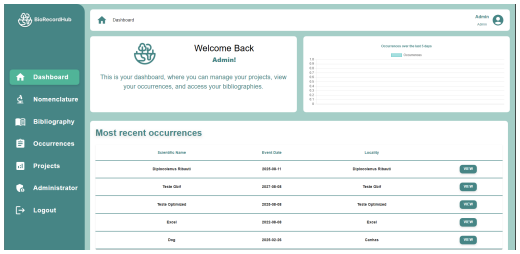


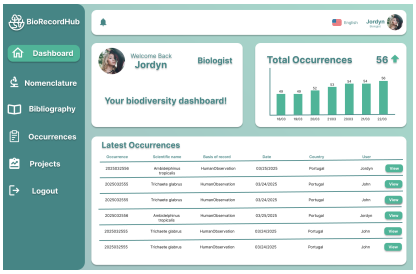
Figure. 23: Error notification example

As defined during the creation of the prototype the system follows a simple structure of having a side bar for navigation and a landing page that is a simple dashboard where the user can see the most recent occurrences created and some statistics about the number of occurrences created in the last five days. When compared with the prototype, this pages includes some changes were

made to improve usability, these where the replacement of the notification icons with breadcrumbs for better navigation and the removal of the language selector as the system is only available in English. This can be seen when comparing the implemented page 24a and the prototype 24b.



(a) Landing Dashboard Page.

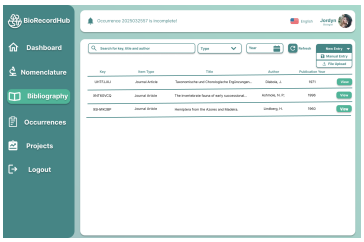


(b) Prototype Dashboard Page.

Figure. 24: Dashboard Page vs Prototype Dashboard Page.

Then the system provides list views, that basically display all the data available for each entity by using a table. For entities like bibliographies, occurrences, and projects this page follows a very similar structure, only changing the actual fields display in the table and the filters provided. When compared with the prototype the only change that was made was the replacement of the refresh button in the filters section for a clear filters button 25b, this way improving usability of the system 25a.

(a) Bibliography List View Page.

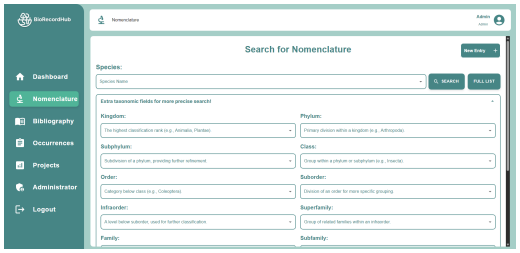


(b) Prototype Bibliography List View Page.

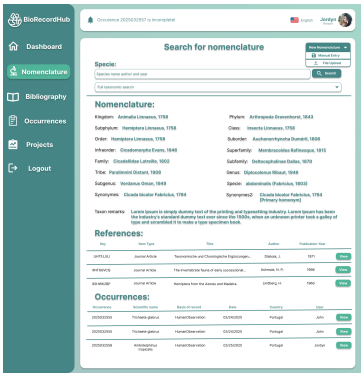
Figure. 25: Bibliography List View Page vs Prototype Bibliography List View Page.

The only list view that is different is the nomenclature list view, as this entity can have a large amount of data making it difficult and not efficient to display all the data on page load, so it provides specific filter so that the user can search for a specific nomenclature. This search can be done using a specific taxonomic level or a species specific name. But in the case where it is

necessary to display all the nomenclatures the user can just click the "Full List" button and the system will load all the available nomenclature 26a. This page also changed when compared with the prototype, as in the prototype, the searched nomenclature would be displayed in the same page 26b, but this was changed so that now the searched nomenclature is displayed in a different page, this way improving usability and navigation.



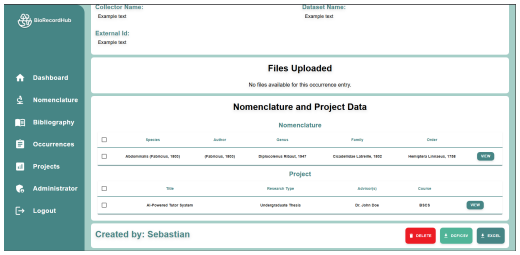
(a) Nomenclature List View Page.



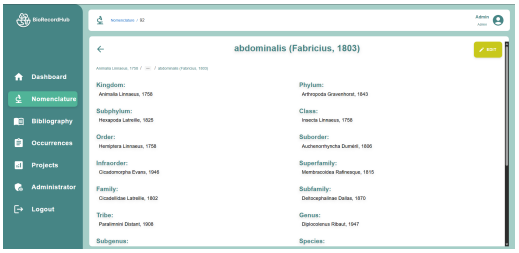
(b) Prototype Nomenclature Page.

Figure. 26: Nomenclature List View Page vs Prototype Nomenclature Page.

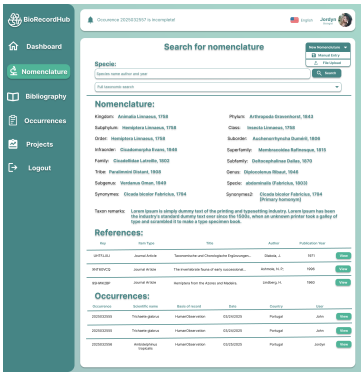
Now for the single views, these pages display the data of a specific record, providing all the relevant information and actions that the user can perform in that record. These also follow a similar structure for bibliographies, occurrences, nomenclatures, and projects, just changing the actual fields displayed 27a. As explained before the only change that occurred within these types of pages was for the nomenclature single view, where before the searched nomenclature would be displayed on the same page as the list view 27c, but now it is displayed in a different page 27b.



(a) Occurrence Single View Page.



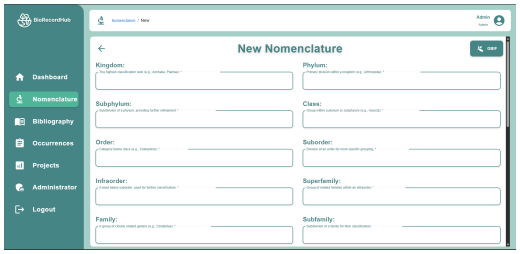
(b) Nomenclature Single Page.



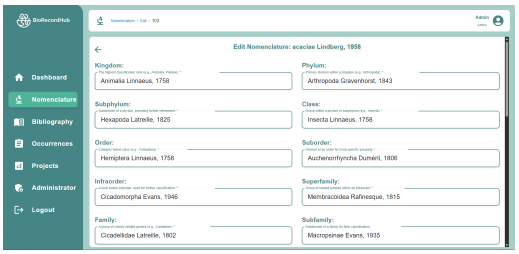
(c) Prototype Nomenclature Page.

Figure. 27: Nomenclature Single View Page vs Prototype Nomenclature Page.

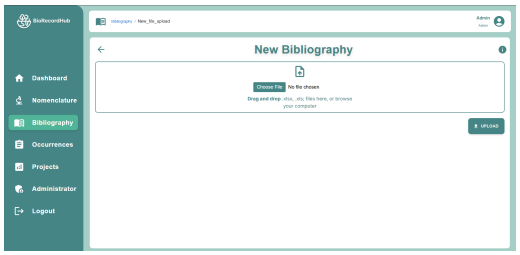
The pages for manually creating 28a and updating 28b records follow the same structure for all the entities, by using forms with the necessary fields for each entity. In the case of importing data from Excel files, the pages have the structure of having a field where the user can upload and submit the file 28c. Then when the file is processed a form is displayed 28d with the data, where the user can review and make necessary corrections if needed before saving the data.



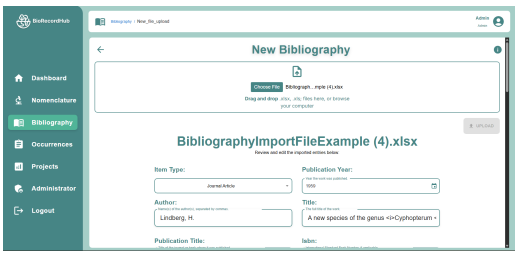
(a) Nomenclature Create Page.



(b) Nomenclature Edit Page.



(c) Upload Page.



(d) Bibliography File Uploaded Page.

Figure. 28: Create, Edit and Import Pages.

The only missing page is the admin page, where the admin can manage the users of the system and the occurrences fields. This page followed the prototype with no changes made, were there is a section with a list of the users and a section with the list of occurrences fields. In these sections the admin has filters and the buttons to initiate the necessary actions 29.

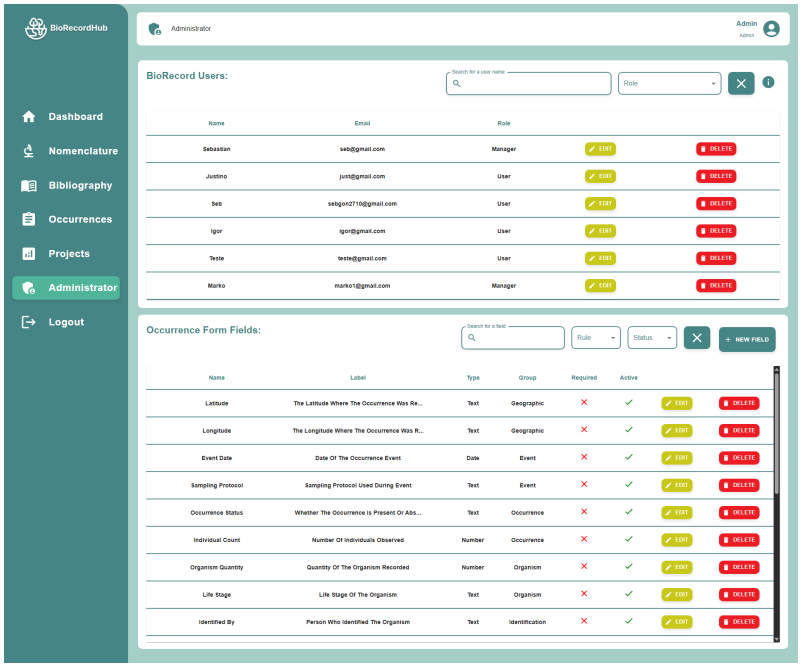
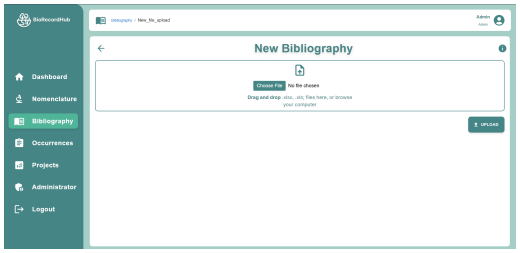


Figure. 29: Admin Page.

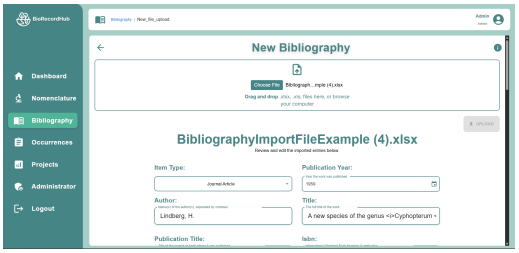
4.3.1 Interoperability with Biodiversity Platforms

One of the key features of this system is the capability of exchanging data with external biodiversity platforms. This interoperability is achieved in this system by the capability of importing and exporting data using Excel files and CSV files in the case of exporting occurrences. Also the system has the capability of retrieving nomenclature data from the Global Biodiversity Information Facility (GBIF) platform using their public API.

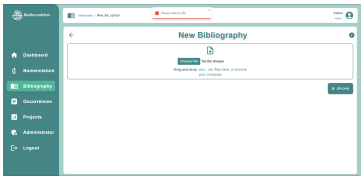
The process of retriving data from the files has been already explained in the previous back-end section, so now the user interaction will be explained. When the user wants to import data using Excel files, they must navigate to the specific page for that entity, for example, bibliographies or nomenclatures. Then on this page the user can upload the file using the provided field and then submit this file 30a. If any error occurs the user will be notified 30c, otherwise the file is processed and the data is sent back to the user 30b. This data is displayed in a form where the user can review it and make any necessary changes before saving it.



(a) Upload File page



(b) File Upload Form

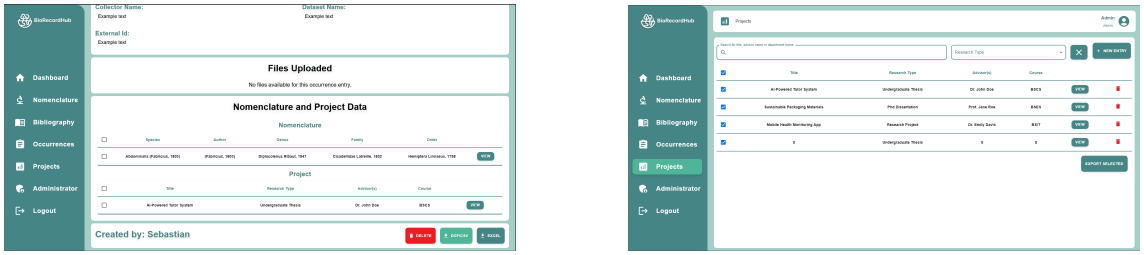


(c) Error notification example

Figure. 30: Import Data Pages

Now the process of exporting data can be done in both single-view and multi-view pages. In single-view pages the export buttons are located at the bottom of the page, allowing the user to export the current record in Excel format in CSV format in the case of occurrences 31a. In multi-view pages the user can select one or multiple records using the checkboxes in the table, then a button will be available under the table to export the selected records 31b. When these export

buttons are clicked the back-end will do all the processing and send the file back to the user for download.

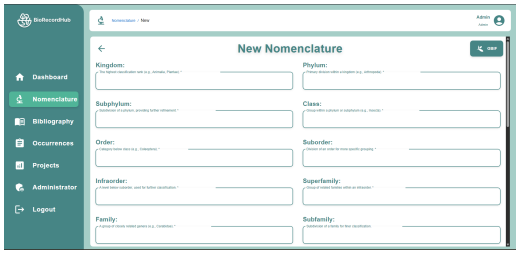


(a) Export Occurrence options

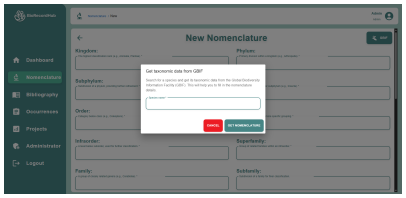
(b) Export List view

Figure. 31: Export Data Pages

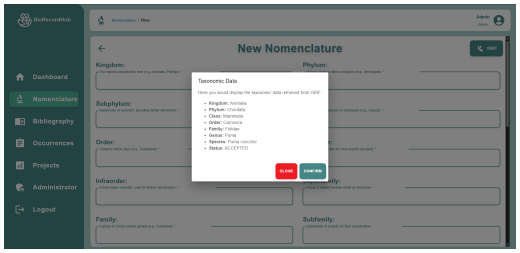
The process of retrieving nomenclature data from GBIF is started when the user chooses to create a new nomenclature manually. Then on the form page there is a button that the user can click to open a dialog where they can enter the species name to be searched 32a.Next when the user submits this request to the GBIF API, two outcomes are possible, depending on whether the species exists in the GBIF database or not 32b. If the species sent dosen´t exist, an error notification is displayed informing the user that no information could be retrieved 32e. In the case that the species exists, the response data will be displayed in the dialog, where the user can review and confirm the information or cancel to leave the dialog 32c. Once the user confirms the data, they can click a button that will populate the new nomenclature form with the retrieved data 32d.



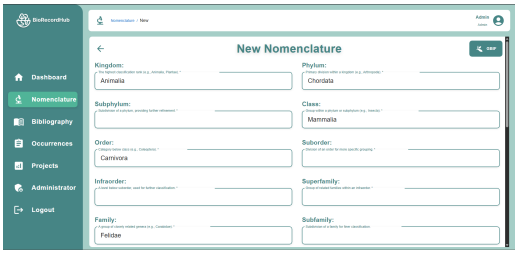
(a) Create Nomenclature page



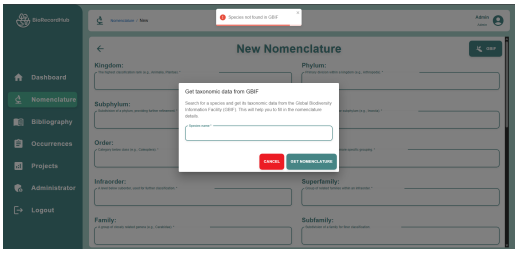
(b) Nomenclature GBIF Dialog



(c) Nomenclature GBIF Dialog Result



(d) Nomenclature Create Form Populated



(e) Nomenclature GBIF Dialog Error

Figure. 32: Nomenclature Interoperability

4.3.2 Front-End Structure

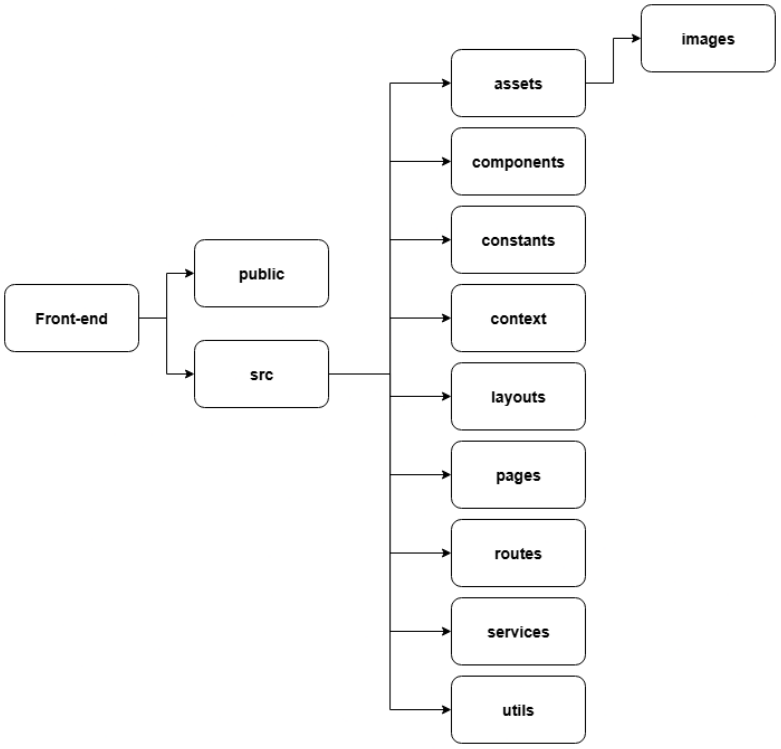


Figure. 33: Front-end tree diagram

The figure (33) illustrates the structure of the system's front-end. In the following section, each user-relevant module will be described in detail.

4.3.2.1 public

The public directory in the front-end is primarily used to store static assets that can be accessed directly by the application. In this project, it contains example Excel files that demonstrate the correct structure required for importing data into the system. These examples serve as user guidance, ensuring that files uploaded for Nomenclature and Bibliography imports follow the expected column format and data organization.

4.3.2.2 src/assets

The src/assets directory is responsible for storing static resources used throughout the front-end application. Inside this module, there are two main subdirectories, the images forlder that contains images such as logos, icons, and other graphical elements. The other subdirectorie are the SVG files, were vector graphics are stored directly.

4.3.2.3 src/components

The src/components directory contains all the reusable components used across the system's interface. These components are designed to keep the code modular, maintainable, and consistent, since common UI elements only need to be implemented once and can then be reused in multiple views.

4.3.2.4 src/constants

The src/constants directory contains configuration files that contain values, options, and UI-related constants that are used throughout the front-end.

4.3.2.5 src/context

The src/context directory is responsible for handling global state management related to user authentication. It contains a single file that defines the User Context, which provides way to manage user information and authentication state across the entire application.

4.3.2.6 src/layouts

The src/layouts directory is responsible for defining the overall structure of the user interface. It ensures consistency in how pages are displayed, while also centralizing common UI elements such as the header and side navigation menu.

4.3.2.7 src/pages

The src/pages directory contains the top-level views of the application. Each file or subfolder in this module represents a complete page within the system, corresponding directly to the routes defined in the application. Unlike components, which are smaller reusable building blocks, pages provide the full layout and business logic for specific features.

4.3.2.8 src/routes

The src/routes directory defines all the navigation and API communication paths of the application. It is split into two files, clearly separating front-end application routes (used by React Router for navigation) and back-end API routes (used by Axios to send requests to the Laravel server).

4.3.2.9 src/services

The src/services directory is responsible for handling all communication between the front-end and the back-end API. It provides the logic where HTTP requests are defined and managed, ensuring that React components and contexts remain focused on presentation and state management. Each service file typically groups together functions for a specific domain (e.g., users, projects, bibliographies, nomenclature), with operations such as retrieving, creating, updating, and deleting records.

4.3.2.10 src/utils

The src/utils directory contains helper functions and shared utilities that support common tasks across the application. These abstractions reduce duplication, enforce consistency, and keep the business logic focused.

4.3.3 Key Takeaways

The front-end created provides a user friendly interface that is easy to use, navigate, and provides all the necessary functions to the users. The implementation of such interface presented some challenges, as the capability of creating an interface that is usable and accurate for the complex biodiversity data was challenging. Also the interoperability with external platform like GBIF and the import/export functionality required special attention to detail to ensure the best user experience. One constraint that this implementation has is that it requieres modern browsers with ES6 support to function correctly. The full implementation details and source code are accessible in the project's GitHub repository [68].

5 Validation

Validation for this project followed a process that combined several methods, ensuring that the system met the need and expectations of its users. The combined methods were SCRUM meetings for iterative feedback from the stakeholders, an agile approach for development where constant feedback from the domain experts was used to improve the system, and a final user testing phase where two of the stakeholders tested the system, and provided feedback. Unfortunately, due to time that the test was conducted, only two users could participate, as most of the stakeholder were in vacation or not available to participate. The profile of the users that participated in the test were both researchers with experience in biodiversity data management but one of them was a senior researcher and collection manager while the other was a junior researcher. The presence of the senior researcher was important to validated the admin and manager functionalities as they are the one that will be using these roles the most. As for the junior researcher helped to validate that the general workflow of the system was easy to use and understand for a less experienced user. Even though, only two users participated in the test, it was possible to test the system as a whole, and get important feedback to improve the system. The next sections will describe the user testing process, the results obtained and the changes made to the system based on the results.

5.1 User Testing

The user testing followed a strategy that made the evaluation process focused on the user experience and usability of the system. So, the test was divided into three main stages: a pre-test questionnaire to collect background information on the participants, the test execution itself where the participants were asked to perform a series of tasks using the system, and a post-test questionnaire to collect feedback and evaluate the usability of the system using the System Usability Scale (SUS). During the test execution, the participants were observed and their interactions, errors, and feedback were recorded, so that it could be analyzed later.

5.1.1 Pre-Test Questionnaire

Questions:

1. What is your primary role? (e.g., researcher, student, administrator, other)

2. How often do you work with biodiversity-related data or information systems? (e.g., Never, Rarely, Sometimes, Often, Daily)
3. How would you rate your general computer skills? (e.g., Beginner, Intermediate, Advanced)
4. How familiar are you with using web-based platforms for data management? (e.g., Not familiar, Somewhat familiar, Familiar, Very familiar)
5. What do you expect this system to help you with?
6. What would you consider most important in a system like this? (e.g., speed, usability, data accuracy, design, etc.)

5.1.2 Test Execution Tasks

1. Create an account in the system.
2. Say how many occurrences were created in the last 5 days.
3. Create a new bibliography record, using the supplied data, in the system using the manual entry option.
4. Import new bibliographies from the supplied Excel file, using the file import option.
5. Visualize in the system the bibliography added in step 2.
6. Create a new nomenclature record using the supplied data.
7. Import new nomenclature from the supplied Excel file.
8. Search for the nomenclature added in step 6 and view it.
9. Create a new project record, using the supplied data, in the system.
10. Filter the projects in the system by the name of the project created in step 9.
11. Create a new occurrence, using the supplied data, in the system.
12. Visualize the occurrence created in step 11.
13. Edit the occurrence created in step 11 and change its country.
14. Export the occurrence (edited in step 13) in CSV format.
15. Export the two first bibliographies.

16. Logout of the system.
17. Login using the administrator credentials provided.
18. Add a new occurrence text field using the supplied data.
19. Change the role of the user created in step 1 to Manager.
20. Delete the project with the title BioRecordHub Thesis.
21. Logout of the system.

5.1.3 Post-Test Questionnaire

Part A - General Questions:

1. What aspects of the system did you like the most?
2. What aspects of the system did you find most difficult or frustrating?
3. Were there any features you expected but did not find in the system?
4. Do you have suggestions for improving the system?

Part B - System Usability Scale (SUS):

Each participant rated the following ten statements on a 5-point Likert scale (1 = Strongly Disagree, 5 = Strongly Agree):

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

The scoring for SUS is calculated following the standard method: For odd-numbered items, the score is the user response minus 1. For even-numbered items, the score is 5 minus the user response. The total is summed and multiplied by 2.5 to get a SUS score between 0 and 100. A SUS score above 68 is considered above average usability, while scores above 80 indicate excellent usability.

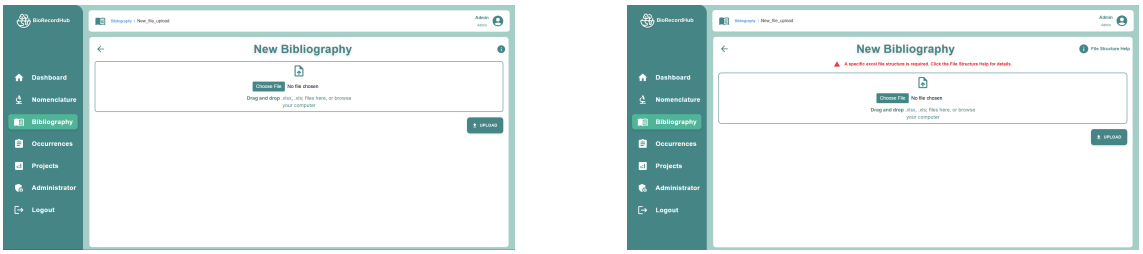
5.2 Validation Results

All the participants identified themselves as researchers with daily experience working with biodiversity data, they also rated their computer skills as advanced as they utilize computers in their daily work. As for their familiarity with web-based platforms, one of the participants considered themselves as familiar while the other only considered as somewhat familiar. Both participants expected the system to help them organize data and make it more accessible then, when asked what was the most important aspect of a system like this both they focused on data accuracy and system usability. As for the test execution, all the tasks were completed successfully by both participants, the only difficulty during the test was that both participants struggled to find the correct format requirements for importing Excel files during task 4. This means that the option that provides this information was not easily locatable within the interface. Other than this issue, both participants indicated that the overall task flow was clear, and the system functionality met their expectations. Participants in the post-test questionnaire expressed overall positive impressions of the system, highlighting its simple user interface and ease of use of the system. The main challenges reported were: understanding the location of the information about the required format for the Excel import as seen during the test execution, and the lack a dedicated page for importing occurrences using Excel files, as well the need of a bibliographic field for occurrences. As for the SUS results, both participants, scored the system with 90 and 85 points, which are well above the generally accepted 68 points for good usability. These results reinforce the prior evaluation and conclusions of the system as a user friendly platform.

5.2.1 Changes Based on the Results

Based on the feedback received from the users, and the results from the test, some changes were made to the system to improve its usability and functionality. The main problem identified during the test was the confusion regarding the Excel import format, as users struggled to identify where to find this information 34a. So to address this issue a new subtitle, in red was added to

highlight the file format requirements, as well as a label for the information icon was added to improve clarity 34b.



(a) Import Bibliography page before changes

(b) Import Bibliography page after changes

Figure. 34: Side-by-side comparison of Import Bibliography pages before and after changes.

Also, a new page for the Excel import of occurrences was developed and added to the system, as the users requested the addition of this feature to facilitate the import of occurrence records. This new page 35 works in the same way as the existing pages for bibliography and nomenclature imports, with the only difference being that it handles occurrence data.

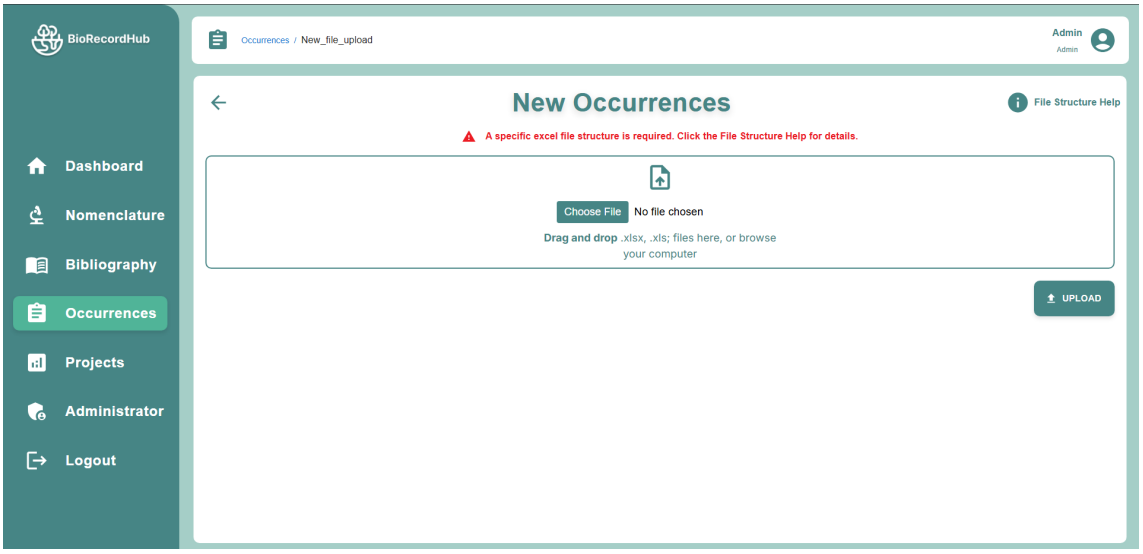
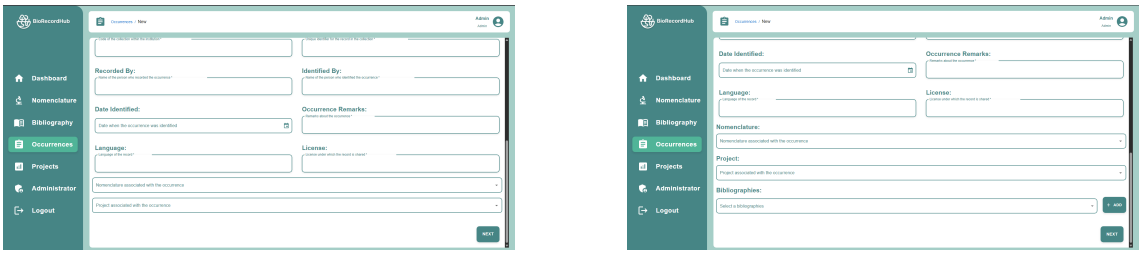


Figure. 35: New page for Excel file upload for occurrences.

The final requested change made to the system, was the addition of a bibliography field to the occurrence creation form 36a. This addition was made so that the users could provide bibliographic references that support the occurrence data, enhancing its data quality and scientific value 36b.

This field was added in the same way as other fields in the occurrence creation form, allowing users to input multiple bibliographic references as needed.



(a) New occurrence creation form before changes

(b) New occurrence creation form after changes

Figure. 36: Side-by-side comparison of New occurrence creation form before and after changes.

5.3 Key Takeaways

The validation process demonstrated that the system can effectively handle the needs of its target users, especially in terms of usability, simplicity, and support for biodiversity data management, but these points where only proven for terrestrial biodiversity data. As it was not possible to test the system with marine biodiversity data and domain experts due to time constraints and availability of such users. Either way, the results obtained from the testing and SUS scores indicate that the system provides a robust and user-friendly platform for managing biodiversity-related data. Considering the current state of the system, some future work tasks, can be identified to further improve the system:

- Expanding interoperability with biodiversity platforms: Currently the system, integrates with GBIF for new entries of nomenclatures. Future work could expand this functionality to include more data types or integration with other biodiversity platforms, such as Catalog of Life or WORMS.
- Improving the supported files: While the current implementation supports some types of files such as Excel, CSV, PDF and images, future work could focus on expanding the range of supported file formats for data import and export, to accommodate a wider variety of user needs.
- Scaling up validation with real-life biodiversity data: While the current validation verified the system's usability and basic functionality, future work could involve large-scale validation with

real biodiversity datasets, to assess performance, scalability, and robustness. Also validation with marine biodiversity experts to ensure the system meets their specific needs.

- Enhancing the user experience: As the system evolves future work could also focus on enhancing the user experience by improving the interface design making the system more appealing.

Overall, the validation results indicate that the system is on the right track to becoming a valuable tool for biodiversity data management, but there are still opportunities for improvement and expansion to better serve its users.

6 Conclusion

Biodiversity research and documentation remains a field that faces several challenges, particularly in terms of data management and accessibility. Existing platforms present several problems from complex user interfaces, low usability, to lack of interoperability between terrestrial and aquatic biodiversity data, and lack of platforms aimed at professionals. This thesis aimed to resolve these issues by developing an ease to use, flexible, interoperable, and professionally aimed web-based back office system for biodiversity data management.

The developed system, supervised by domain experts, provides a structured system, where users can manage species data, occurrences, bibliographies, projects, and taxonomy. This modular architecture provides essential functionalities such as authentication with role-based management, import and export of data sets, and a user-friendly interface based on modern design principles. The system also provides interoperability with external biodiversity infrastructures through APIs. Also the utilization of standardized frameworks such as Laravel Breeze and Maatwebsite/Excel together with the libraries such as Material UI and React Toastify ensures maintainability and scalability.

Validated by domain experts, the system successfully handled the necessary tasks across the needs of its users. Even though the tests were only coducted with two people, due to availability constraints from the domain experts, both experts confirmed that the system aligned with what they need from a system like this. All the validation participants confirmed that the system, even thought it was created for professionals, can be used without extensive technical knowledge as well. The feedback provided from tests helped improve the system further, as some problems were indentified, such as the location of the excel import instructions was not clear enough, also the lack of a dedicated import page for occurrences, and missing bibliographic field in occurrences. By addressing these issues, and with the high SUS scores of 90 and 85 obtained, it´s clear that the system achieved its goal of being both effective and user friendly. Also some possible future work was identified, such as a large scale validation with large scale testing with large biodiversity datasets, and improving the interoperability of the system with the integration with other biodiversity platforms.

Considering all the aspects of this project, the developed system fulfills its initial proposition of being a unified, interoperable, and easy to use system. Lowering the technical barriers, improves

usability, and supports standardized data exchange, the system lays a foundation for a possible comprehensive and collaborative biodiversity research and conservation in the future. Making this project a success in achieving its goals and objectives, which by consequence makes this thesis a success as well.

References

[1] R. M. May, "How many species are there on earth?" *Science*, vol. 241, no. 4872, pp. 1441–1449, 1988. [Online]. Available: https://www.montana.edu/screel/teaching/bioe-440r-521/documents/May1988.pdf

[2] L. Holden, R. G. Lee, L. Orsini, N. Eastwood, J. Zhou, and A. Čavoški, "Biodiversity management challenges: A policy brief," *Environmental Law Review*, vol. 26, no. 2, pp. 141–150, 2024. [Online]. Available: https://journals.sagepub.com/doi/10.1177/14614529241247361

[3] A. E. Cahill, M. E. Aiello-Lammens, M. C. Fisher-Reid, X. Hua, C. J. Karanewsky, H. Yeong Ryu, G. C. Sbeglia, F. Spagnolo, J. B. Waldron, O. Warsi *et al.*, "How does climate change cause extinction?" *Proceedings of the Royal Society B: Biological Sciences*, vol. 280, no. 1750, p. 20121890, 2013. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rspb.2012.1890

[4] Ž. Rode and G. Torkar, "The inaturalist application in biology education: A systematic review." *International Journal of Educational Methodology*, vol. 9, no. 4, pp. 725–744, 2023. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00219266.2020.1739114

[5] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The inaturalist species classification and detection dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8769–8778. [Online]. Available: https://ieeexplore.ieee.org/document/8579012/

[6] S. Unger, M. Rollins, A. Tietz, and H. Dumais, "inaturalist as an engaging tool for identifying organisms in outdoor activities," *Journal of Biological Education*, vol. 55, no. 5, pp. 537–547, 2021. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00219266.2020.1739114

[7] B. L. Sullivan, C. L. Wood, M. J. Iliff, R. E. Bonney, D. Fink, and S. Kelling, "ebird: A citizen-based bird observation network in the biological sciences," *Biological conservation*, vol. 142, no. 10, pp. 2282–2292, 2009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S000632070900216X

[8] H. Goëau, P. Bonnet, A. Joly, V. Bakić, J. Barbe, I. Yahiaoui, S. Selmi, J. Carré, D. Barthélémy, N. Boujemaa, J.-F. Molino, G. Duché, and A. Péronnet, "Pl@ntNet mobile app," in *Proceedings of the 21st ACM International Conference on Multimedia*. ACM, pp. 423–424. [Online]. Available: https://dl.acm.org/doi/10.1145/2502081.2502251

[9] T. Cheeseman, T. Johnson, K. Southerland, and N. Muldavin, "Happywhale: Globalizing marine mammal photo identification via a citizen science web platform," *Happywhale, Santa Cruz, CA, USA, Rep. SC/67b/PH/02*, 2017. [Online]. Available: https://cabowhaletrek.com/wp-content/uploads/2017/06/Cheeseman-et-al-2017-Happywhale-for-IWC-SC_67A_PH_02.pdf

[10] Q. Chen, O. Beijbom, S. Chan, J. Bouwmeester, and D. Kriegman, "A new deep learning engine for coralnet," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 3693–3702. [Online]. Available: https://ieeexplore.ieee.org/document/9607450/

[11] J. Wieczorek, D. Bloom, R. Guralnick, S. Blum, M. Döring, R. Giovanni, T. Robertson, and D. Vieglais, "Darwin core: an evolving community-developed biodiversity data standard," *PloS one*, vol. 7, no. 1, p. e29715, 2012. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0029715

[12] C. Yesson, P. W. Brewer, T. Sutton, N. Caithness, J. S. Pahwa, M. Burgess, W. A. Gray, R. J. White, A. C. Jones, F. A. Bisby *et al.*, "How global is the global biodiversity information facility?" *PloS one*, vol. 2, no. 11, p. e1124, 2007. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0001124

[13] M. Fornwall, "Planning for obis: examining relationships with existing national and international biodiversity information systems," *Oceanography*, vol. 13, no. 3, pp. 31–38, 2000. [Online]. Available: https://tos.org/oceanography/article/planning-for-obis-examining-relationships-with-existing-national-and-intern

[14] M. J. Costello, P. Bouchet, G. Boxshall, K. Fauchald, D. Gordon, B. W. Hoeksema, G. C. Poore, R. W. van Soest, S. Stöhr, T. C. Walter *et al.*, "Global coordination and standardisation in marine biodiversity through the world register of marine species (worms) and related databases," *PloS one*, vol. 8, no. 1, p. e51629, 2013. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0051629

[15] D. Hobern, S. K. Barik, L. Christidis, S. T. Garnett, P. Kirk, T. M. Orrell, T. Pape, R. L. Pyle, K. R. Thiele, F. E. Zachos *et al.*, "Towards a global list of accepted species vi: The catalogue of life checklist," *Organisms Diversity & Evolution*, vol. 21, no. 4, pp. 677–690, 2021. [Online]. Available: https://link.springer.com/10.1007/s13127-021-00516-w

[16] C. Cicero, M. S. Koo, E. Braker, J. Abbott, D. Bloom, M. Campbell, J. A. Cook, J. R. Demboski, A. C. Doll, L. M. Frederick *et al.*, "Arctos: Community-driven innovations for managing natural and cultural history collections," *Plos one*, vol. 19, no. 5, p. e0296478, 2024. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0296478

[17] S. M. Hennekens and J. H. Schaminée, "Turboveg, a comprehensive data base management system for vegetation data," *Journal of vegetation science*, vol. 12, no. 4, pp. 589–591, 2001. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.2307/3237010

[18] M. A. Lane and J. L. Edwards, "The global biodiversity information facility (gbif)," *Systematics Association special volume*, vol. 73, p. 1, 2007.

[19] L. Vandepitte, S. Dekeyzer, W. Decock, L. Delgat, B. Boydens, L. Tyberghein, B. Vanhoorne, F. Hernandez, J. Mees, T. Horton *et al.*, "The world register of marine species (worms) through the looking glass: insights from the data management team in light of the crystal anniversary of worms," *Hydrobiologia*, pp. 1–22, 2024. [Online]. Available: https://link.springer.com/10.1007/s10750-024-05688-y

[20] J. F. Grassle and K. I. Stocks, "A global ocean biogeographic information system (obis) for the census of marine life," *Oceanography*, vol. 12, no. 3, pp. 12–14, 1999. [Online]. Available: https://tos.org/oceanography/article/a-global-ocean-biogeographic-information-system-obis-for-the-census-of-mari

[21] K. Katija, E. Orenstein, B. Schlining, L. Lundsten, K. Barnard, G. Sainz, O. Boulais, M. Cromwell, E. Butler, B. Woodward *et al.*, "Fathomnet: A global image database for enabling artificial intelligence in the ocean," *Scientific reports*, vol. 12, no. 1, p. 15914, 2022. [Online]. Available: https://www.nature.com/articles/s41598-022-19939-2

[22] O. Boulais, B. Woodward, B. Schlining, L. Lundsten, K. Barnard, K. C. Bell, and K. Katija, "Fathomnet: An underwater image training database for ocean exploration and discovery,"

arXiv preprint arXiv:2007.00114, 2020. [Online]. Available: http://arxiv.org/abs/2007.00114

[23] L. G. García, M. Fernández, and J. M. Azevedo, "Monicet: The azores whale watching contribution to cetacean monitoring," *Biodiversity Data Journal*, vol. 11, 2023. [Online]. Available: https://bdj.pensoft.net/article/106991/

[24] M. Buzinkai, M. Radeta, C. Rodrigues, F. Silva, R. Freitas, S. Chebaane, P. Parretti, S. Schäfer, R. Silva, F. Gizzi *et al.*, "Crowdsourcing biodiversity data from recreational scuba divers using dive reporter," *Ecological Informatics*, vol. 77, p. 102191, 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1574954123002200

[25] A. C. Jones, R. J. White, and E. R. Orme, "Identifying and relating biological concepts in the Catalogue of Life," vol. 2, no. 1, p. 7. [Online]. Available: http://jbiomedsem.biomedcentral.com/articles/10.1186/2041-1480-2-7

[26] O. Bánki, "Catalogue of Life: From a list to a service," vol. 6, p. e94040. [Online]. Available: https://biss.pensoft.net/article/94040/

[27] M. E. Baker, S. Rycroft, and V. S. Smith, "Linking multiple biodiversity informatics platforms with darwin core archives," *Biodiversity data journal*, no. 2, 2014. [Online]. Available: http://bdj.pensoft.net/articles.php?id=1039

[28] L. Vandepitte, B. Vanhoorne, W. Decock, S. Vranken, T. Lanssens, S. Dekeyzer, K. Verfaille, T. Horton, A. Kroh, F. Hernandez *et al.*, "A decade of the world register of marine species–general insights and experiences from the data management team: Where are we, what have we learned and how can we continue?" *PLoS One*, vol. 13, no. 4, p. e0194599, 2018. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0194599

[29] A. Affouard, H. Goëau, P. Bonnet, J.-C. Lombardo, and A. Joly, "Pl@ ntnet app in the era of deep learning," in *ICLR: International Conference on Learning Representations*, 2017. [Online]. Available: https://hal.science/hal-01629195/document

[30] A. Joly, P. Bonnet, H. Goëau, J. Barbe, S. Selmi, J. Champ, S. Dufour-Kowalski, A. Affouard, J. Carré, J.-F. Molino, N. Boujemaa, and D. Barthélémy, "A look inside the Pl@ntNet experience: The good, the bias and the hope," vol. 22, no. 6, pp. 751–766. [Online]. Available: http://link.springer.com/10.1007/s00530-015-0462-9

[31] A. Wang, "The shazam music recognition service," *Communications of the ACM*, vol. 49, no. 8, pp. 44–48, 2006.

[32] V. Lang and A. Šorgo, "Added value of the pl@ ntnet smartphone application for the motivation and performance of lower secondary school students in species identification," in *ICERI2022 Proceedings*. IATED, 2022, pp. 4534–4540.

[33] H. S. Oluwatosin, "Client-server model," *IOSR Journal of Computer Engineering*, vol. 16, no. 1, pp. 67–71, 2014. [Online]. Available: http://www.iosrjournals.org/iosr-jce/papers/Vol16-issue1/Version-9/J016195771.pdf

[34] M. G. M. Nyabuto, V. Mony, and S. Mbugua, "Architectural review of client-server models," *International journal of scientific research and engineering trends*, vol. 10, no. 1, pp. 139–143, 2024.

[35] A. Syromiatnikov and D. Weyns, "A Journey through the Land of Model-View-Design Patterns," in *2014 IEEE/IFIP Conference on Software Architecture*. IEEE, pp. 21–30. [Online]. Available: http://ieeexplore.ieee.org/document/6827095/

[36] D.-P. Pop and A. Altar, "Designing an MVC Model for Rapid Web Application Development," vol. 69, pp. 1172–1179. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S187770581400352X

[37] A. Leff and J. Rayfield, "Web-application development using the Model/View/Controller design pattern," in *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*. IEEE Comput. Soc, pp. 118–127. [Online]. Available: http://ieeexplore.ieee.org/document/950428/

[38] S. Necula. Exploring The Model-View-Controller (MVC) Architecture: A Broad Analysis of Market and Technological Applications. [Online]. Available: https://www.preprints.org/manuscript/202404.1860/v1

[39] C. Anderson, "The model-view-viewmodel (mvvm) design pattern," in *Pro Business Applications with Silverlight 5*. Springer, 2012, pp. 461–499.

[40] R. J. Qureshi, "A COMPARISON OF MODEL VIEW CONTROLLER AND MODEL VIEW PRESENTER."

[41] S. Chen, U. R. Thaduri, and V. K. R. Ballamudi, "Front-end development in react: an overview," *Engineering International*, vol. 7, no. 2, pp. 117–126, 2019. [Online]. Available: https://abc.us.org/ojs/index.php/ei/article/view/662

[42] C. Gackenheimer, *Introduction to React.* Apress, 2015.

[43] A. Fedosejev, *React. js essentials.* Packt Publishing Ltd, 2015.

[44] A. Freeman, *Pro Angular*, 5th ed. Apress, 2022.

[45] P. Deeleman, N. Murray, F. Coury, A. Lerner, and C. Miranda, *ng-book: The Complete Guide to Angular*, 2nd ed. Fullstack.io, 2020.

[46] A. Team, "Angular documentation," 2023. [Online]. Available: https://angular.io/docs

[47] G. Geetha, M. Mittal, K. M. Prasad, and J. G. Ponsam, "Interpretation and analysis of angular framework," in *2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS).* IEEE, 2022, pp. 1–6.

[48] R. Harris, "Rethinking reactivity," 2019, accessed: 2025-05-26. [Online]. Available: https://svelte.dev/blog/rich-harris-rethinking-reactivity

[49] M. Holmes, *Svelte and Sapper in Action.* Manning Publications, 2021.

[50] S. Roberts, *Svelte 3 Up and Running.* O'Reilly Media, 2020.

[51] M. Stauffer, *Laravel: Up & running: A framework for building modern php apps.* " O'Reilly Media, Inc.", 2019.

[52] R. Y. He, "Design and implementation of web based on laravel framework," in *2014 International Conference on Computer Science and Electronic Technology (ICCSET 2014).* Atlantis Press, 2015, pp. 301–304. [Online]. Available: https://www.atlantis-press.com/article/16334

[53] M. Bean, *Laravel 5 essentials.* Packt Publishing Ltd, 2015.

[54] J. Plekhanova, "Evaluating web development frameworks: Django, ruby on rails and cakephp," *Institute for Business and Information Technology*, vol. 20, p. 2009, 2009.

[55] D. H. Hansson and R. C. Team, "Ruby on rails," 2009.

[56] B. Tate and C. Hibbs, *Ruby on Rails: Up and Running: Up and Running.* " O'Reilly Media, Inc.", 2006.

[57] S. Holmes, *Getting MEAN with Mongo, Express, Angular, and Node.* Manning Publications, 2014.

[58] S. Tilkov and S. Vinoski, "Node.js: Using javascript to build high-performance network programs," *IEEE Internet Computing*, vol. 14, no. 6, pp. 80–83, 2010.

[59] P. Duffy, *MySQL.* Sams Publishing, 2003.

[60] S. Kozlov, *MySQL 8 Cookbook.* Packt Publishing Ltd, 2019.

[61] J. Murach and S. Urban, *Murach's MySQL.* Mike Murach and Associates, Inc., 2005.

[62] K. Chodorow, *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage.* O'Reilly Media, Inc., 2013.

[63] K. Banker, *MongoDB in Action.* Manning Publications, 2011.

[64] B. Momjian, *PostgreSQL: Introduction and Concepts.* Addison-Wesley, 2001.

[65] R. O. Kirkpatrick and L. Hsu, *PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database.* O'Reilly Media, Inc., 2015.

[66] R. S. Douglas and Matthew, *Beginning PostgreSQL.* Apress, 2011.

[67] S. Gonçalves, "Github. biorecordhub back-end source code." 2025. [Online]. Available: https://github.com/Sebastian2702/BioRecordHub_Backend

[68] ——, "Github. biorecordhub front-end source code." 2025. [Online]. Available: https://github.com/Sebastian2702/BioRecordHub_Frontend