PM

A Platform for Touristic Visualisation

MASTER'S DEGREE PROJECT

João Miguel Mendes Vasconcelos
MASTER IN INFORMATICS ENGINEERING

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

September | 2025

A Platform for Touristic Visualisation

MASTER'S DEGREE PROJECT

João Miguel Mendes Vasconcelos
MASTER IN INFORMATICS ENGINEERING

SUPERVISOR
Eduardo Miguel Dias Marques

CO-SUPERVISOR
Karolina Baras

Faculdade de Ciências Exatas e da Engenharia

Master of Science Degree in Informatics Engineering

A Platform for Touristic Visualisation

Supervised by: Eduardo Marques e Karolina Baras

Constituição do júri de provas públicas:

David Sardinha Andrade de Aveiro (categoria), Presidente

Mara Sofia Gomes Dionísio (categoria), Vogal

Eduardo Miguel Dias Marques (categoria), Vogal

Tuesday 17th March, 2026

Summary

In modern data analytics, visualisation platforms play a vital role in transforming raw data into useful insights. With the vast amount of data available online, these platforms make it easier to extract value by presenting complex information in clear, visual formats, enabling users to better understand the data and make well-informed decisions.

Focusing on touristic data, the SmartDest project, developed by the University of Madeira, created a platform to collect, analyse, and visualise tourism-related data. However, a formal analysis of the existing platform identified several major issues: a limited diversity of graphs, a complete lack of dashboards, and significant performance issues in data visualisation that hindered effective analysis.

To address these problems, the SmartDest platform architecture was systematically redesigned and implemented using a modern, open-source data stack comprising dbt Core for data transformation, Apache Airflow for orchestration, and Apache Superset for visualisation. The project followed a multi-phase methodology, beginning with a heuristic review of the original platform, followed by the design and implementation of the new architecture. The solution's effectiveness was then validated through four case studies targeting the identified problem areas.

The evaluation confirmed that the new architecture resolves the identified limitations, providing flexible visualisations, performant data integration, and a solid governance framework. However, the results also highlight that these increased capabilities come with increased technical complexity and specific architectural trade-offs, including the lack of built-in Column-Level Security and constraints in multi-tenant embedded scenarios, which must be considered for future implementation.

Keywords: Tourism Data Visualization · SmartDest Project · Platform Architecture · Dashboard Development · Visualization Performance · Data Analysis

Resumo

As plataformas de visualização transformam dados brutos em insights úteis ao apresentar informações em formatos visuais, permitindo que os utilizadores compreendam os dados e tomem decisões informadas.

Focando-se em dados turísticos, o projeto SmartDest, desenvolvido pela Universidade da Madeira, criou uma plataforma para recolher, analisar e visualizar dados relacionados com o turismo. No entanto, uma análise formal da plataforma existente identificou várias questões importantes: uma diversidade limitada de gráficos, a ausência completa de *dashboards* e problemas significativos de desempenho na visualização de dados que prejudicavam uma análise eficaz.

Para resolver estes problemas, a arquitetura da plataforma SmartDest foi sistematicamente redesenhada e implementada utilizando uma stack de dados moderna e de código aberto, composta pelo dbt Core para a transformação de dados, Apache Airflow para a orquestração e Apache Superset para a visualização. O projeto seguiu uma metodologia multifásica, começando com uma análise heurística da plataforma original, seguida pelo desenho e implementação da nova arquitetura. A eficácia da solução foi então validada através de quatro estudos de caso, direcionados para as áreas problemáticas identificadas.

A avaliação confirmou que a nova arquitetura resolve as limitações identificadas, fornecendo visualizações flexíveis, integração de dados performante e uma estrutura sólida de governação de dados. No entanto, os resultados também destacam que estas capacidades melhoradas implicam uma complexidade técnica acrescida e compromissos arquitetónicos específicos, incluindo a ausência de segurança a nível de coluna (*Column-Level Security*) nativa e restrições em cenários de incorporação *multi-tenant*, que devem ser considerados para futuras implementações.

Keywords: Visualização de Dados Turísticos · Projeto SmartDest · Arquitetura de Plataformas · Desenvolvimento de Dashboards · Desempenho de Visualização · Análise de Dados

Acknowledgments

I would like to express my sincere gratitude to everyone who supported me throughout the course of this dissertation.

To my family, thank you for your unwavering support, understanding, and encouragement. Your patience during this demanding period was greatly appreciated and provided a strong foundation for my work.

I am also grateful to my friends for their companionship and for providing a much-needed sense of balance. Your encouragement was a constant source of motivation.

I owe special thanks to my supervisor, Professor Eduardo Marques. His insightful guidance, methodological rigour, and consistent availability were invaluable in shaping this research. His contributions were essential to the completion of this thesis.

I would also like to thank my co-supervisor, Professor Karolina Baras, for her valuable feedback and the constructive discussions that helped refine the direction of this project. Her perspectives significantly improved the quality of this work.

Finally, my appreciation goes to the colleagues and professors in the program. The academic environment and the exchange of ideas were instrumental in my learning and development.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AI Attractiveness Index

API Application Programming Interface

BI Business Intelligence

CLS Column-Level Security

CORS Cross-Origin Resource Sharing

CSS Cascading Style Sheets

DAG Directed Acyclic Graph

DVP Data visualisation Platform

ELT Extract, Load, Transform

ERP Enterprise Resource Planning

ETL Extract, Transform, Load

GDP Gross Domestic Product

GIS Geographic Information System

GPS Global Positioning System

HTML HyperText Markup Language

IDATD Integrated Data Analytic Tourism Dashboard

IaC Infrastructure-as-Code

JWT JSON Web Token

MIUT Madeira Island Ultra-Trail

MSMEs Micro, Small, and Medium Enterprises

OLAP Online Analytical Processing

OSM OpenStreetMap

PoIs Points of Interest

RBAC Role-Based Access Control

RLS Row-Level Security

SDK Software Development Kit

SQL Structured Query Language

SSO Single Sign-On

TTDI Travel and Tourism Development Index

WI Walkability Index

YAML Yet Ain't Markup Language

dbt data build tool

glTF Graphics Library Transmission Format

1 Introduction

In modern data analytics, visualisation platforms play a role in transforming raw data into useful insights in the tourism sector. These platforms present complex information in clear, visual formats, enabling users to better understand data and make well-informed decisions [1]. However, existing tourism data has often been limited in scope, aggregated over long periods, and dispersed across various sources [2]. For instance, data has typically originated from "tourist agencies, tourist facilities, and consumers of tourism services on the Internet" [3], leading to issues of "massiveness, heterogeneity, sometimes weak structuring, incompleteness, and inconsistency" [3]. This fragmentation has historically made it difficult for stakeholders to gain detailed insights and make data-driven decisions. The objective of this project is to address these challenges by redesigning a platform to overcome these limitations.

The SmartDest project, developed by the University of Madeira, was part of a broader initiative to convert islands into Smart Tourist Destinations. This was achieved through a two-phase strategic plan involving a diagnostic study and the subsequent implementation of key initiatives in areas such as data management and intelligent information systems. As part of this effort, a platform was created to collect, analyse, and visualise tourism-related data, capable of generating both simple and complex graphics. The project aimed to improve tourism on Madeira Island through smart technologies and data-driven approaches.

However, a formal analysis of the existing SmartDest platform revealed several key issues: a limited diversity of existing graphs, a lack of dashboards, and performance issues in data visualisation. These shortcomings hindered the platform's ability to effectively support tourism research and decision-making, particularly as "the importance of predictive analytics is also on the rise" [4], which required more advanced capabilities than the previous platform offered.

To address these identified problems, the redesign was guided by a primary technical objective: design an architecture capable of integrating and automatically transforming data from multiple heterogeneous sources into a unified, governed analytical layer. This required an automated ETL pipeline, separation of raw and serving data stores, data quality validation, and role-based access controls. Dashboards and Visualisation was a main requirement, and Embeddable analytics into external workflows was a defined secondary requirement.

To achieve these objectives, the work will be structured into multiple phases: a formal analysis of the existing platform, the proposal of a new architecture, and its subsequent development and validation. This approach establishes a framework for data visualisation, built on a data stack designed to address shortcomings at each stage of the data lifecycle. The suitability of the proposed solution will be empirically validated through four case studies, each designed to test a specific objective.

The thesis is structured to present this plan for improving a visualisation platform like SmartDest. It commences with an Introduction (Chapter 1) that highlights the role of data visualisation in tourism and details the SmartDest platform's limitations. The State of the Art (Chapter 2) explores current visualisation solutions, tourism data types, specific touristic platforms, and Business Intelligence tools. The Problem Description (Chapter 3) outlines the specific challenges faced by SmartDest. The Methodology (Chapter 4) details the systematic approach, including the evaluation of the existing platform and the plan for the validation case studies. The Development (Chapter 5) section describes the practical steps of implementation, and the Evaluation (Chapter 6) will present the findings from the case studies. Finally, the Conclusion (Chapter 7) will summarise the project's achievements and how the improved platform transforms complex data into a strategic asset.

2 State of the Art

This chapter reviews current visualisation solutions and their application in the tourism sector. It begins by examining the core functionalities of visualisation platforms and their ability to handle heterogeneous data sources. Following this, it analyses techniques for visualising specific types of tourism data, such as festival metrics, national performance indicators, and environmental data. Finally, the chapter evaluates widespread Business Intelligence (BI) tools used to build these platforms, comparing their technical requirements, costs, and integration capabilities.

To identify relevant articles for this research, systematic searches were conducted on Google Scholar[1] and the ACM Digital Library[2] between September 2024 and June 2025.

This selection process resulted in a corpus of 34 contributions that form the foundation for the analysis in this chapter. The analysis is structured to cover the state of the art in a logical progression, beginning with broad Visualisation Platforms [7-10], narrowing to the specific types of Data Visualisation [11-16], then focusing on dedicated Touristic Platforms [3, 17-25], and concluding with an in-depth review of the Visualisation Tooling used to build these systems [1, 26-39].

2.1 Visualisation Platforms

Modern visualisation platforms provide an interface between computational power and human comprehension through smart visual interfaces [5].

The contemporary design philosophy of visualisation software is an extensibility and modularity one, whereby diverse data sources and analysis tools can be plugged in. This kind of flexibility in design is important when one is operating in the heterogenous data spaces common in tourism use cases, where one gets data from booking systems, social networking websites, GPS tracking devices, and customer review websites [6].

Platform architects need to accommodate the various formats, shapes, and sizes of these data sources and use data transformation and normalisation features to deliver integrated visualisation experiences [6, 7].

[1] https://scholar.google.com/
[2] https://dl.acm.org/

Building upon this foundation of dynamic interaction, extensibility and modularity are equally central. A platform called the Data visualisation Platform (DVP) is taking on an architecture that incorporates computational power directly into visualisation methods in one platform [5].

Interactive data visualisation has the potential to help solve problems of exploring and understanding data by allowing researchers and analysts to work with data in ways that are unencumbered by static displays [8].

Such frameworks have become essential for world trend analysis and developing large-scale visualisations intended for research and educational use [9].

Figure 1 serves as a visual outline of the systematic approach undertaken to build the visualisation platform [9]. This figure is a direct illustration of the method by which the platform was conceived, developed, and deployed. The process began with acquiring datasets from various sources, such as the Global Terrorism Database, Global Air Quality Dataset, and Global Population Dataset. Following data acquisition, an important step involved preprocessing the data, which included cleaning, handling missing values, and correcting errors to prepare it for visualisation. The methodology then progressed to the development of three dynamic visualisation web pages, each focusing on a specific global trend (terrorism, air quality, population), and these were linked together through a central homepage. The figure also encompasses the deployment strategy, noting that the homepage was deployed on Vercel, while the individual visualisations were deployed on Render, which like Vercel is a cloud hosting server, due to specific deployment limitations. This systematic process also involved exploring and utilising different visualisation libraries like Plotly Express, Plotly.js, and D3.js simultaneously, which necessitated separate deployments to manage compatibility issues.

These frameworks help to develop complex visualisations, but they can even be used on other devices and operating systems [9]. This is especially true for tourism applications because users may gather information from other environments and places [6].

The development of custom web applications such as Wiz reflects the increasing maturity of interactive visualisation tools for big data contexts [8].

The evolution of these systems reflects an advancement in data analysis and understanding [6,8]. Extensibility and modularity allow diverse data sources and analysis tools to be integrated, which is

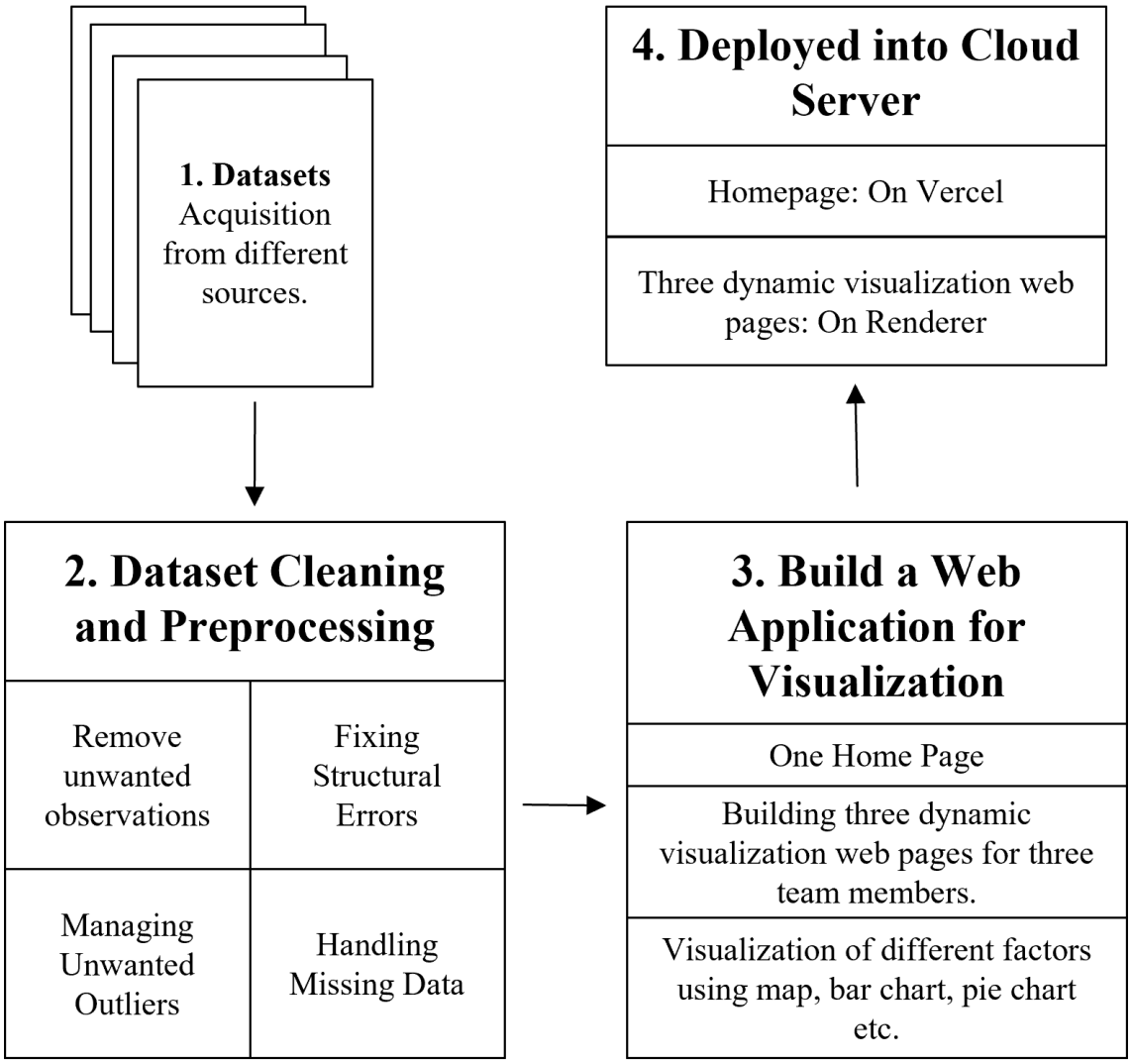

Fig. 1: Methodology example of visualisation platform [9].

particularly vital for handling heterogeneous data spaces found in tourism, where data originates from varied systems like booking platforms, social media, and GPS devices [6]. The shift from traditional static approaches to interactive data visualisation has proven important for exploring intricate data with numerous relationships between variables [8]. Custom web applications like Wiz illustrate the increasing maturity of interactive tools for big data contexts; they are adaptable across devices and operating systems, making them especially beneficial for tourism applications where information may be gathered from diverse environments and locations [8, 9].

2.2 Data Visualisation in tourism

This subsection adopts a *data-centric* perspective: it examines which visualisation techniques and chart types are best suited to different categories of tourism data (e.g., festival metrics, envi-

ronmental indicators, satisfaction feedback). The focus is on the mapping between data types and visualisation approaches, rather than on specific platforms or systems.

The interpretation and application of data are essential for strategic planning and operational efficiency in the tourism sector [10, 11].

The Optimisation of Banyuwangi Festival Management through ERP-Tourism Using Dashboard primarily deals with festival management data, aspects like event coordination, performance metrics, financial details (expenditures and income), and information on participating Micro, Small, and Medium Enterprises (MSMEs). For this type of data, an interactive dashboard was highlighted as the most suitable visualisation. This dashboard offered a succinct presentation of key metrics, graphs, and tables for overseeing system operations. By providing clear overviews of current situations, developments, and trends, this visualisation enables managers to monitor advancements, detect problems, and make well-informed judgments regarding festival quality and efficiency [10].

In the context of Developing a Tourism Performance Dashboard for a National Tourism Organisation in Indonesia, data was categorised into micro and macro types. Macrodata, such as foreign exchange value from tourism, its contribution to GDP, workforce size, Travel and Tourism Development Index (TTDI) ratings, and numbers of foreign and domestic tourist visits, is best visualised through graphs and tables that emphasize comparisons of previous, current, and target data. A particularly effective visualisation for macro-level performance is the head-to-head comparison feature for international tourist arrivals and TTDI values against competitor countries, allowing for rapid benchmarking. For microdata, which includes detailed tourist demographics, travel patterns, spending habits, and perceptions, the dashboard aims to provide full analytical capabilities, often implying detailed charts and filtered views to explore specific segments. Additionally, a public dashboard module provides real-time alerts and statistics for dynamic information like COVID-19 status and earthquake alerts, ensuring immediate awareness for public safety and travel advisory [11].

For Optimising Sustainable Tourism Capacity in Tanjung Bira, including tourist visit numbers, detailed waste management data (types, volume, frequency of plastic, glass, metal, paper, styrofoam, and organic waste), and environmental metrics like rainfall and water quality, alongside staffing information. The proposed visualisation is a Real-Time Environmental Management Dashboard. This dashboard integrates various elements for clarity: Real-Time Statistics are presented

for total accumulated waste and bin fullness indicators, often accompanied by pie or bar charts to display waste types by percentage, enabling quick understanding of composition. Notably, an Interactive Map is employed to visualise trash bin conditions using colour-coded indicators (green for empty, yellow for half-full, red for full) and to mark waste hotspots, enabling prompt and targeted responses for environmental action. Reports and Analysis sections, which would use trend lines and comparative graphs, provide daily, weekly, and monthly waste trends, alongside feedback analysis, crucial for sustainable management strategies [12].

As seen in Figure 2, the visualisation showcases the user interface of the Real-Time Environmental Management Dashboard [12]. The central feature is an interactive map of the Tanjung Bira area, overlaid with various points of interest and navigational icons. The interface is structured to provide access to different categories of information. On the left, an 'Environmental Quality Infographics' panel lists the key metrics the platform is designed to monitor, including Physical Carrying Capacity (PCC), water pH levels, waste management (Pengelolaan Sampah), and rainfall. This panel indicates the types of data the system tracks. Above it, separate modules are available for mapping tourist attractions (peta sebaran wisata) and accommodations. The figure effectively illustrates the dashboard's layout and intended functionalities rather than displaying specific real-time data points or analytical charts.

The Smart Dashboard and Slow Tourism in Santa Barbara Walk uses a blend of intrinsic characteristics (length, difficulty, duration of tracks), extrinsic characteristics (points of interest like mining sites, cultural sites, natural sites, urban areas within a buffer), and important smart, user-generated data from social networks (likes, followers, reviews, ratings, visitor demographics, and travel periods). To effectively integrate these diverse datasets, a circular smart dashboard is designed. This dashboard displays georeferenced tracks and points of interest directly on Google Maps, providing a spatial context for the intrinsic and extrinsic data. It also presents a physical profile for each track, coupled with calculated Walkability Index (WI) and Attractiveness Index (AI), offering qualitative and quantitative insights into the routes. Furthermore, it prominently features ratings from social networks, transforming user perceptions into quantifiable metrics for decision-makers and potential visitors. The design emphasises constant updates on performance indicators, including dynamic information like weather and social network ratings, reflecting a commitment to real-time, user-informed governance [13].

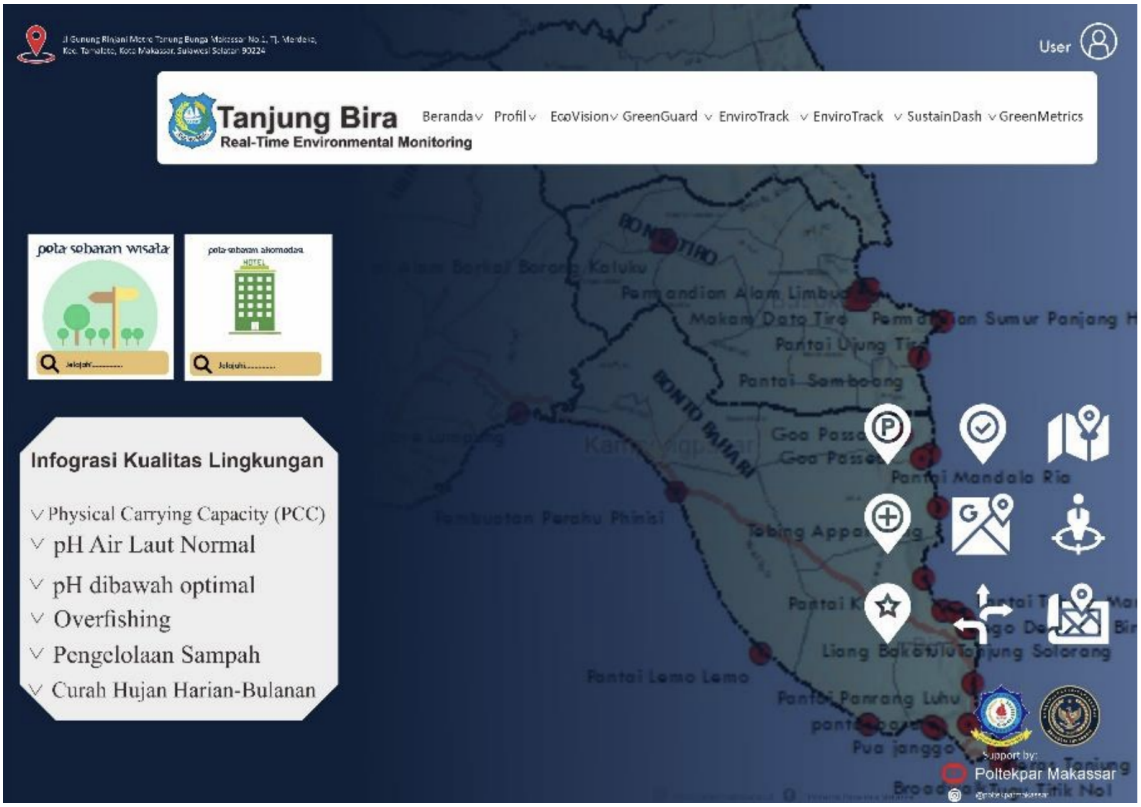

Fig. 2: Dashboard created with Real-Time Statistics [12].

Figure 3, exemplifies the dashboard design of the platform, illustrating how georeferenced tracks and points of interest are presented on Google Maps, along with the physical profile of the track, its Walkability Index (WI) and Attractiveness Index (AI), and ratings from main social networks [13].

The Integrated Data Analytic Tourism Dashboard (IDATD) project, primarily focuses on measuring tourist satisfaction in Oman. The core data for this is tourist feedback, gathered through various digital and physical tools, including electronic rating devices, website rating features, and extensive data from social media accounts (comments, posts, images). Uniquely, it also incorporates live pictures from aerial photography to observe tourist behaviours and tourist movements tracked via SIM cards. This vast quantity of diverse Big Data (videos, text, images, sounds) is processed and then presented graphically on a dashboard. The most salient visualisation is the use of pointers of tourist satisfaction, where a green pointer signifies good satisfaction and a red pointer indicates bad satisfaction, offering an immediate and intuitive visual summary for decision-makers to gauge and improve tourism services [14].

Finally, in the realm of advanced geo-dashboard development for tourism research and management in Mallorca, a wide array of data types are integrated to address specific tourism challenges.

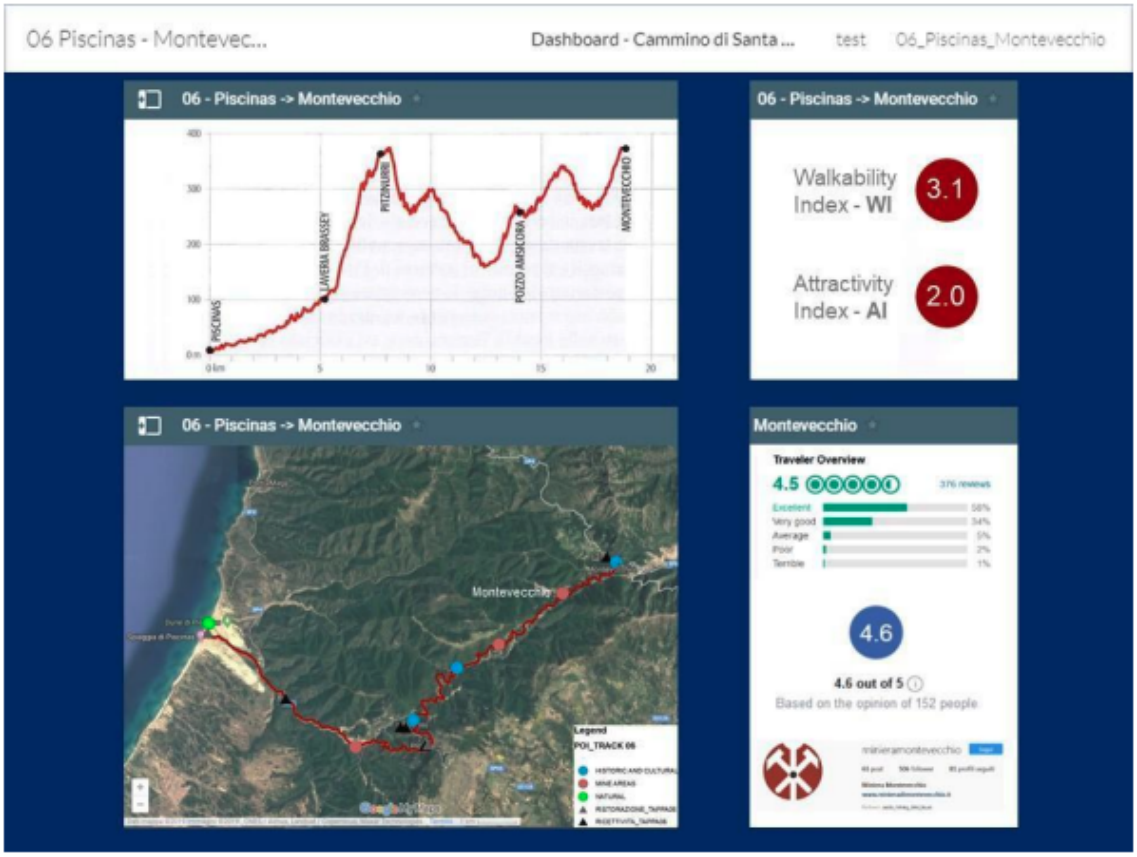

Fig. 3: Dashboard created of Santa Barbara Walk [13].

This includes location data for tourist accommodation, flood recurrence periods, various land-cover data, municipal division data, geographic metadata from social media photos (specifically Flickr geo-positioning), and urban cadastral cartography, demographic, and income data. For this diverse spatial and alphanumeric data, geo-dashboards are presented as the optimal visualisation. These are defined as interactive graphical interfaces that integrate geospatial and alphanumeric information. Key visualisations within these geo-dashboards include interactive maps with legends, complemented by various graphs and charts such as bar charts (e.g., for flood exposure of accommodation or land-cover changes), and summary graphs for population and income data. Choropleth maps are specifically highlighted for visualising the distribution of human pressure (total photos per census tract) or the area of hotels and restaurants by census tract, providing clear spatial patterns for detailed analysis and strategic planning [15].

In conclusion, the diverse applications of data visualisation in the tourism sector, as evidenced across these studies, consistently highlight the utility of tailored approaches for different data types. From interactive dashboards providing concise financial and operational metrics for festival man-

agement [10], to macro and micro data visualisations with head-to-head comparisons for national tourism performance [11], and real-time interactive maps with colour-coded indicators for environmental management [12], each solution is designed to improve data comprehension. The integration of user-generated content with intrinsic and extrinsic characteristics on circular smart dashboards offers an overall view of destination appeal [13], while simple satisfaction pointers convert complex feedback into useful insights [14]. Furthermore, geo-dashboards prove indispensable for integrating a multitude of spatial and alphanumeric data to address complex urban and environmental challenges in destinations like Mallorca [15]. Effective data visualisation transforms information into an intuitive format, enabling stakeholders to make strategic decisions that support the tourism industry [16].

2.3 Touristic Platforms Visualisation

In contrast to the previous subsection, this one adopts a *platform-centric* perspective: it surveys concrete visualisation systems and platforms deployed in tourism contexts. The focus is on how these platforms integrate multiple data sources, which technologies they use, and what functionalities they offer to end users. While the previous Section addressed "what to visualise and how" for given data types, this section addresses "which integrated platforms exist" and how they implement such visualisation in practice.

Such systems play a role in advancing tourism's digital transformation [16]. Nikolaychuk et al. present a tourism industry monitoring service based on analysis of informational web resources [3]. The dashboard allows users to filter and explore data on accommodations, including facility numbers, average costs, service categories, and locations. Tourist routes and excursions are visualised with departure points, maps, and diagrams, simplifying complex data and revealing trends to support tourism management.

Krataithong, Anutariya, and Buranarach in [17] discusses taxi GPS data's utility for tracking tourist movements but notes its lack of semantic context hinders deeper analysis. Researchers proposed a data platform visualising enriched data through dashboards. It processes tourist trajectories, segments routes, and integrates knowledge about tourist sites. Dashboards then reveal insights like origin-destination points, places visited, distances, travel times, and activity-based route categories such as religious, gastronomic, and leisure. These dashboards offer a full view of

tourist behaviour, revealing patterns in movement and activity, including high-concentration areas, peak times, and popular destinations.

Another example of a touristic visualisation platform is ViTFlow, a platform addressing the "invisible" environmental impacts of tourism, especially on fragile island ecosystems. It visualises tourist flows and environmental impacts using multi-source data, such as sensors, energy stats, weather, and transport information. ViTFlow employs interactive maps, animations, and overlays to explore tourism density, energy use, CO2 emissions, and more, with contextual panels detailing flights, cruises, demographics, and temperature [18].

Figure 4 showcases a ViTFlow visualisation of Madeira and Porto Santo [18]. It highlights flight paths to mainland Portugal, energy consumption trends (e.g., peaks during Festa da Cebola in May 2017), and overlays like temperature and pedestrian paths in Funchal.

ViTFlow promotes awareness of tourism's impact through engaging visuals, promoting dialogue and sustainable decisions [18]. Prandi et al. conducted two exploratory studies in Madeira focused on measuring this impact; their findings discuss the potential of such systems to foster awareness about sustainability issues [18].

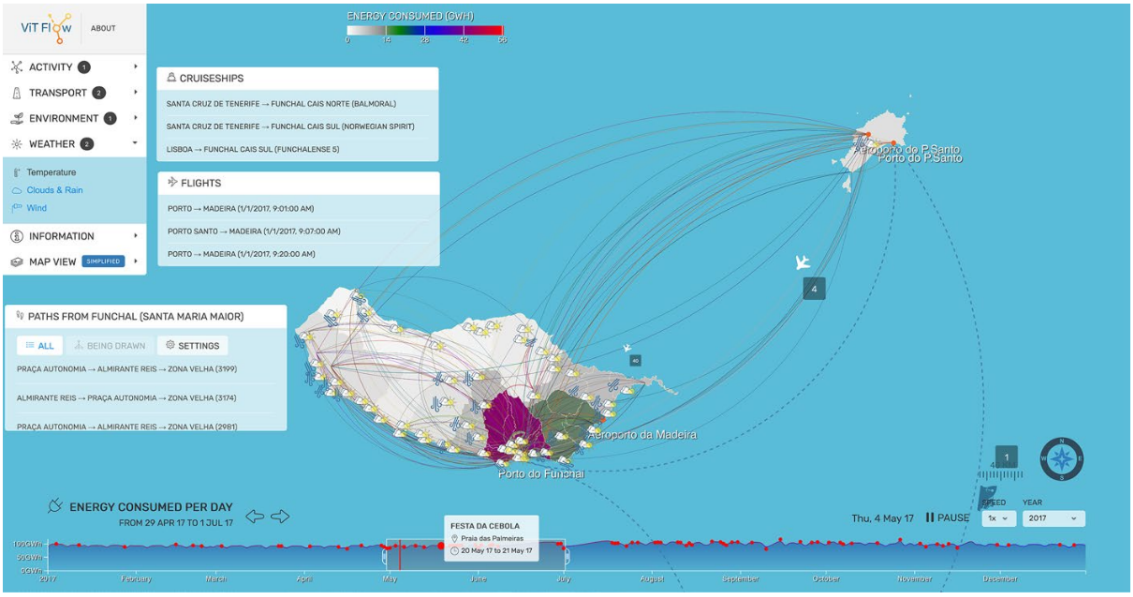

Fig. 4: Visualisation example of ViTFlow [18].

BITOUR is a platform with interactive visualisation that addresses the challenge of managing vast tourism data to improve decision-making and competitiveness [19]. It integrates data from

OpenStreetMap (OSM), Twitter, TripAdvisor, and Airbnb. OSM provides spatial data, Twitter captures social sentiment, and TripAdvisor and Airbnb offer accommodation insights. The data is processed and then interactive maps and graphs reveal patterns in tourist behaviour, such as stay durations, tweet trends, and links between attractions and accommodations. Figure 5 illustrates these capabilities.

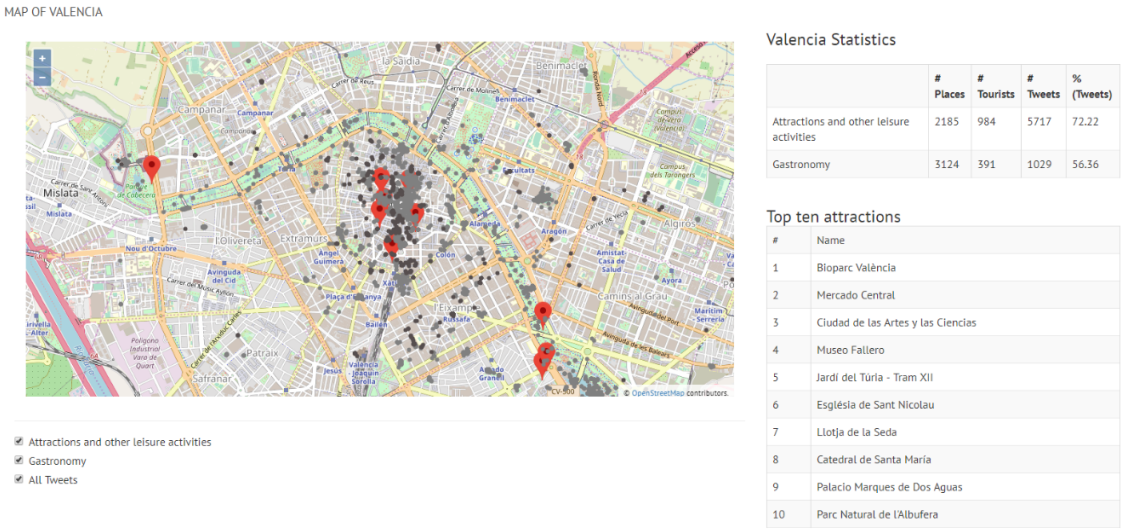

Fig. 5: Visualisation example of BITOUR [19].

The visualisation maps Valencia, highlighting attractions (red markers) like Bioparc València and Ciudad de las Artes y las Ciencias [19, 20]. Black dots indicate Twitter activity, while a panel details statistics: 72.22% of tweets relate to attractions and leisure, and 56.36% to gastronomy. The "Top Ten Attractions" ranks popular sites like Museo Fallero.

Visualisation platforms must incorporate sophisticated data mapping and transformation engines that are capable of interpreting data in different source formats into normalized structures for visualisation and analysis [7]. Such transformations must execute efficiently to maintain real-time performance while being capable of providing data accuracy and completeness throughout the integration pipeline [7].

One example of web-based tourism visualisation is BITOUR, which uses technologies like OpenLayers, HTML5, and AngularJS to deliver useful insights via interactive dashboards [19].

Bustamante, Sebastia, and Onaindia in [20] discuss an interactive visualisation website, highlighting the challenge of presenting tourism data analysis results clearly. To address this, the researchers developed an interactive website with OpenLayers, featuring maps of Valencia and Berlin. Users can explore tourist sites, view tweet statistics within a 500-meter radius, and access insights on attractions and gastronomy. The platform enables zooming and categorises tweets by type, offering statistics on places, tourists, and tweets, helping stakeholders understand activity patterns and densities and translating complex data into useful insights for tourism planning and marketing [20].

Berko et al. emphasises the need for interactive and modular online tourism information systems that provide concise, user-focused content [21]. It highlights the importance of allowing users to choose the information they want and build their own routes, rather than being restricted to predefined tours. For that the platform is developed with technologies as React.js and JavaScript. This adaptability and user-focused design, along with the use of interactive elements like quest games, helps to create an engaging and personalised experience, making it more likely that users will return and recommend the platform.

Cannata et al. describe the use of open-source web platforms for 3D documentation and storytelling of cultural heritage sites as seen in [22] and how accessibility is key to help users operate the platform. The platform allows for the display of a site's evolution over time by incorporating historical data and 3D reconstructions of past states. The focus on accessibility for both experts and non-technical users, with features like guided tours and interactive annotations, makes it suitable for a broad audience. The open documentation of the platform also ensures that others can replicate and customise it for their own use cases.

Faucher et al. introduces [23] a modular and generic platform for extracting, transforming, and visualising mobility data, which is important for understanding tourist behaviour in cities. The platform is designed to be user-friendly, even for non-computer scientists, by allowing them to create processing pipelines from a variety of modules. The platform can process spatio-temporal mobility tracks and associated contextual data, like points of interest and weather information. The use of visualisation modules like the space-time cube and the ability to display various data enrichments on the map makes it easier for decision makers to analyse the data. This enables a

deeper understanding of tourist movements, which is essential for effective urban planning and tourism management.

Lopes et al. describe in [24] the implementation of a single-page application with an interactive multimedia map that simplifies the presentation of cultural heritage content. The map is loaded on a single page, which improves the user experience by minimising loading times. The data is stored in a GeoJSON file instead of a database, which simplifies the publication of geo-referenced information. The platform includes various tools for browsing and filtering Points of Interest (PoIs), including "Table," "Folder," and "Tile" functions. The use of multi-shape markers with unambiguous colours and the ability to filter PoIs by category, type, accessibility, and timeline improves usability. The addition of features, such as a "Time slider" and the ability to view all previews one by one in sequence, further improves the user experience.

These systems demonstrate that multi-source data integration is technically feasible in tourism contexts, but they vary in architectural completeness. Table 1 compares the surveyed platforms across the architectural dimensions most relevant to this work.

Table 1: Comparison of Touristic Visualisation Platforms

Platform	**Multi-Source**	**Automated Pipeline**	**Open Source**	**Governance**	**Embeddable**
BITOUR [19]	Yes	No	No	No	No
ViTFlow [18]	Yes	No	No	No	No
Taxi GPS Platform [17]	Partial	No	No	No	No
IDATD [14]	Yes	No	No	No	No

2.4 Visualisation Tooling for Touristic Platform

To support the development of such platforms, tools that help the creation of visualisation and the processing of data are essential [25]. Business Intelligence (BI) tools serve as fundamental components in creating these platforms, offering diverse capabilities ranging from basic data visualisation to advanced analytical processing [1]. The selection of an appropriate BI tool is an important decision, demanding careful consideration of multiple factors, including customisation requirements, the technical expertise available within an organisation, scalability needs, and budgetary constraints [26, 27]. This requires an understanding of the diverse capabilities offered by

various BI tools, which range from highly adaptable solutions requiring significant technical proficiency to more user-friendly platforms with limited customisation options.

BI tools can be categorised by customisation capability, ranging from extensively customisable solutions that demand significant technical expertise to user-friendly options with more limited flexibility [1, 26].

Microsoft Power BI is a widely used BI tool that simplifies data processing and visualisation, assisting companies in overcoming challenges related to interpreting complex tabular reports and uncovering trends. It offers capabilities for transforming raw data into meaningful insights through interactive dashboards, customisable reports, and real-time data updates. Dashboards created with Power BI can integrate data from multiple sources into a data warehouse and use various visualisations, such as maps, bar graphs, line graphs, word clouds, and tables, to clearly present key performance indicators. Its implementation has been shown to improve data analysis by visualising sales by country, product category, shipping methods, and trends over time, with interactive filters enabling deeper exploration. Power BI also incorporates AI-powered features and advanced analytics tools, making predictive analytics more accessible, although advanced customisation may require some technical knowledge. It is classified as a moderately customisable BI tool [1].

Tableau is another popular tool renowned for its ability to present data interactively and intuitively, integrating with various data sources. It significantly aids tourism data visualisation for policy formulation, resource management, and infrastructure development by clearly presenting complex visitation trends and patterns. The tool improves interactivity with features like a year filter, allowing users to explore specific time-frames and easily compare total visits and visitor categories. Utama et al. exemplified this by using Tableau to visualise tourist data from Karo Regency, Indonesia (2010–2016), featuring dashboards with horizontal and vertical bar charts to display total visits, average visits, and child vs. adult visits for attractions, this dashboard is illustrated in Figure 6 [28].

Utama et al. report popular sites like Lag Sibayak Lau Debuk-Debuk (1,099,738 visitors) and Gundaling, while showing lower visitation for Lag Sinabung and Danau Lau Kawar (52,282 visitors) [28]. Tableau dashboards simplify data analysis, enabling the identification of patterns, such as the dominance of popular sites like Lag Sibayak Lau Debuk-Debuk, which accounted for nearly 40% of visits, and highlighting the need for strategies to boost less visited sites, thus supporting better

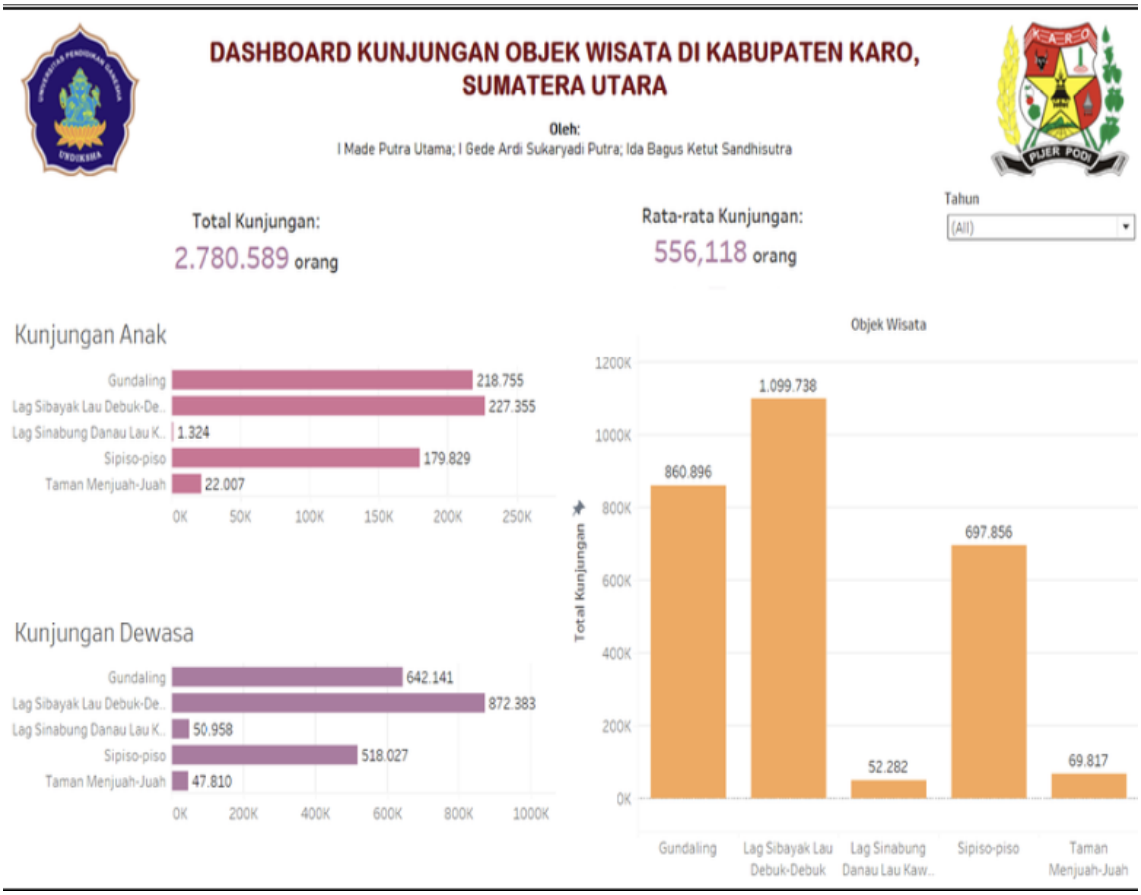

Fig. 6: Dashboard created with Tableau [28].

tourism management and development planning [28]. While offering strong visualisation capabilities and interactive features, Tableau operates within a more structured framework compared to fully customisable solutions [1]. It is considered a moderately customisable BI tool.

Google Data Studio provides an intuitive interface for creating interactive dashboards and visualisations, capable of connecting to various data sources. This tool simplifies data interpretation and decision-making, particularly addressing the difficulty of identifying patterns in large tabular datasets, which is important for effective decision-making in areas like tourism. A study used Google Data Studio to visualise tourist data from East Nusa Tenggara (2019–2022), revealing trends in arrivals and offering insights for stakeholders to improve tourism strategies. However, it may encounter difficulties with very large datasets or necessitate technical skills for advanced features [29]. The platform is designed primarily for ease of use but offers limited customisation options compared to more advanced tools. Google Data Studio is categorised as a low customisable BI tool.

Apache Superset is highlighted as an open-source business intelligence (BI) tool that serves as a cost-effective alternative to commercial BI solutions such as Microsoft Power BI [30]. Through its interface, support for various visualisations, and customisation capabilities via CSS and plug-ins, Superset enables organisations to design interactive dashboards, monitor performance indicators, and perform data analysis without incurring licensing costs. Research indicates that Superset not only matches the essential functionalities of leading BI tools but also provides significant advantages in terms of adaptability and affordability, making it a compelling choice for organisations seeking to implement a good BI solution. Apache Superset is flexible, falling into both the highly customisable BI tool category due to its customisation potential, and also offering mid-level functionality that classifies it as a moderately customisable tool for standard dashboard creation without requiring extensive technical expertise.

D3.js and Custom Web Development represent approaches that offer unlimited customisation potential for visualisation solutions, often by utilising third-party libraries like D3.js and Google Charts. These custom solutions, however, demand significant technical expertise and development resources. The TweetViz tool, for instance, is an example of such a custom solution, specifically designed for analysing and visualising data from Twitter [31]. TweetViz provides insights into user behaviour, interests, and general Twitter activity surrounding specific keywords or hashtags. It enables users to explore how topics evolve over time using a Stream graph visualisation that incorporates algorithms to represent tweets as mixtures of topics. Such custom solutions are typically categorised as highly customisable BI tools.

Grafana is a highly customisable monitoring and visualisation platform [32]. It provides data collection, alerting, and visualisation capabilities that can be extensively customised for specific use cases. Grafana requires continuous refinement to meet evolving scalability, compatibility, and compliance demands [32]. It is categorised as a highly customisable BI tool.

Custom Python-based solutions have emerged as powerful alternatives, specifically addressing the need for BI tools that require extensive customisation for non-technical users [26]. These solutions prioritise ease of use, flexibility in data formatting, preprocessing capabilities, and relevancy of data insights. Python-based platforms enable individuals without technical expertise to effectively engage with, examine, and extract valuable insights from intricate datasets. They are also categorised as highly customisable BI tools.

The Our World in Data platform, offers interactive charts and data visualisations that depict the history and current state of tourism [33]. These resources encompass a variety of topics, including international tourist trips, employment in tourism-related industries, and CO2 emissions from flights. A key characteristic is that all visualisations, data, and code on the Our World in Data website are open access. While accessible and user-friendly, this platform provides limited customisation options for specific organisational needs. It is generally considered a low customisable BI tool.

A web-based decision support tool presented by Fegraus et al. integrates diverse data sources, such as GIS data, household surveys, field data, and remote sensing imagery [34]. This dashboard displays various metrics of ecosystem stress through maps, graphs, and tables. It improves user interaction by providing pop-ups that display original graphs and statistics when the user interacts with the map. A significant advantage of this platform is its ability to visualise the dynamics of a landscape at different levels, ranging from a continent to specific regions and even local plots. The tool's primary focus is on making complex data and indicators easily understandable for decision-makers and other stakeholders, providing valuable insights for resource management and improving user experience.

Schoedon et al., introduced a web-based system for rendering transportation networks using reachability maps [35]. This system employs a client/server model to offload tasks to the server, which then uses glTF tiles for efficient, interactive visualisations. This approach enables real-time mobility analysis, offering benefits to tourism platforms by improving accessibility insights and reducing client-side load times. It demonstrates how advanced visualisation techniques can support real-time data processing for improved decision-making in tourism.

Common to these visualisation tools are several key features that enable the creation of insightful platforms. These include data integration from multiple sources, allowing for the combination of disparate datasets into a unified view. They provide interactive dashboards that enable users to explore data with filters, often revealing patterns and insights that might be missed in traditional reports. These tools also offer diverse visualisations such as maps, bar graphs, line graphs, word clouds, and tables to clearly present key performance indicators. Many feature user-friendly interfaces that simplify data interpretation and decision-making for both experts and non-technical

users. Additionally, some tools support real-time analysis, which improves accessibility insights and enables immediate decision-making [25].

When selecting Business Intelligence (BI) tools, organisations must carefully consider several key factors to ensure the chosen solution aligns with their specific needs and resources [25–27].

Technical expertise requirements are paramount, as the choice of BI tool largely depends on the technical skills available within the organisation. Highly customisable tools, such as Apache Superset, Python-based solutions, D3.js, and Grafana, demand significant programming skills and technical knowledge for effective implementation and offer very high customisation potential. Conversely, low customisable tools, including Google Data Studio and Our World in Data, are specifically designed for non-technical users and require minimal technical expertise, offering low to medium customisation options. Moderately customisable tools like Power BI and Tableau strike a balance, requiring a medium level of technical expertise for their medium-high customisation capabilities [26].

Budgetary considerations are also significant in the selection process. Open-source solutions like Apache Superset provide a cost-effective alternative to commercial BI solutions, making them attractive for organisations with limited budgets because they do not incur licensing costs. In contrast, commercial tools such as Power BI and Tableau typically involve licensing fees but often provide professional support and maintenance. Custom web development approaches, while offering unlimited customisation, can also incur medium to high costs due to the required development resources [27].

Scalability needs are important, requiring organisations to assess their expected data volume and user base [26]. Some BI tools may struggle with very large datasets or necessitate advanced technical skills to access their full features, while others are specifically designed to handle enterprise-scale deployments. For instance, Grafana requires continuous refinement to meet evolving scalability demands [32]. Integration requirements are vital, as the ability of a BI tool to integrate with existing data sources and systems provides greater value, especially for organisations with complex data ecosystems. Tools like Power BI can integrate data from multiple sources into a data warehouse, while Tableau and Google Data Studio are renowned for connecting to various data sources. The capacity to combine disparate datasets into a unified view is a common and essential feature across many visualisation tools [25].

Finally, the need for real-time analysis capabilities should be carefully considered, particularly for organisations that require immediate decision-making based on current data. Tools that support real-time data processing and analysis, such as Power BI with its real-time data updates, improve accessibility insights and enable prompt decision-making. Advanced visualisation techniques can enable real-time mobility analysis, supporting improved decision-making in sectors like tourism [36].

The Table 2 provides a detailed comparative analysis of various Business Intelligence (BI) tools [1,25,26]. It is structured to help quickly grasp the trade-offs involved in choosing a particular tool based on an organisation's specific needs and available resources. Each row represents a specific BI tool, highlighting its distinct characteristics.

The table details several important factors for each tool [25, 26]:

- Technical Expertise Required: This column indicates the level of technical skill necessary for effective implementation and use of the tool. It ranges from Low for user-friendly platforms like Google Data Studio and Our World in Data, designed for non-technical users, to Medium for tools like Microsoft Power BI and Tableau, and High for highly customisable solutions such as Apache Superset, D3.js, Grafana, and Custom Python-based solutions, which demand significant programming skills and technical knowledge.

- Cost: This factor reflects the budgetary implications, primarily licensing fees or development resources. Low costs are associated with open-source or free tools like Apache Superset, Grafana, Google Data Studio, and Our World in Data, which do not incur licensing fees. Medium to High costs are typically linked to commercial tools such as Microsoft Power BI and Tableau due to licensing fees, or to custom development approaches like D3.js and Custom Python-based solutions due to required development resources.

- Interactive Dashboards: This column assesses the tool's capability for creating dynamic and exploratory dashboards. Tools like Microsoft Power BI, Tableau, Apache Superset, D3.js, Grafana, and Custom Python-based solutions are rated as High as they offer capabilities for creating interactive dashboards and customisable reports, enabling deep data exploration. Google Data Studio and Our World in Data are rated as Medium, as they provide interactive features but with more limited customisation options.

– Multi-Source Integration: This indicates the tool's ability to combine data from multiple, disparate sources into a unified view. Tools such as Microsoft Power BI, Tableau, Apache Superset, D3.js, Grafana, and Custom Python-based solutions demonstrate High integration capabilities, allowing them to connect with various data sources and even accommodate heterogeneous systems. Google Data Studio offers Medium integration, capable of connecting to various sources, while Our World in Data primarily presents its own curated data, thus showing - for user-defined multi-source integration.

– Real-time Analysis: This column specifies whether the tool supports real-time data updates and processing for immediate decision-making. High capability in real-time analysis is noted for Microsoft Power BI and Grafana, which offer real-time data updates and monitoring features. Other tools generally show, indicating this is not a primary, out-of-the-box feature or requires significant custom development.

Table 2: Comparative Analysis of Business Intelligence Tools

Tool	**Technical Expertise Required**	**Cost**	**Interactive Dashboards**	**Multi-Source Integration**	**Real-time Analysis**
Microsoft Power BI	Medium	Medium to High	High	High	High
Tableau	Medium	Medium to High	High	High	High
Google Data Studio	Low	Low	Medium	Medium	-
Apache Superset	High	Low	High	High	-
D3.js	High	Medium to High	High	High	-
Grafana	High	Low	High	High	High
Custom Python-based solutions	High	Medium to High	High	High	-
Our World in Data	Low	Low	Medium	-	-

This overview enables an understanding of the strengths and limitations of each BI tool for informed decision-making [25, 26].

Tools span a spectrum from extensively customisable options like Apache Superset, D3.js, Grafana, and Custom Python-based solutions, which demand significant technical expertise and development resources but offer unparalleled flexibility, to mid-tier tools such as Microsoft Power BI and Tableau that balance functionality with ease of use [1,26,30]. Entry-level options like Google Data Studio and Our World in Data are ideal for organisations with limited technical resources or straightforward visualisation needs, providing quick implementation and immediate value [29, 33]. Regardless of customisation level, effective BI tools enable data integration from multiple sources, enable interactive dashboards with filters, and offer diverse visualisations for clearer presentation of key performance indicators, thereby enabling stakeholders to make informed decisions and improving tourism management [25]. The choice of a BI tool is important in transforming complex data into a strategic asset for the tourism sector [16, 28].

2.5 Conclusions

The surveyed touristic platforms confirm that multi-source integration is an established requirement, but none of the reviewed systems implements a fully automated transformation pipeline with data quality validation, governance controls, and embeddability as integrated features.

From the BI tooling survey, Apache Superset provides the necessary capabilities—multi-source connectivity, dashboards, RBAC, Row-Level Security, a public API, and embedded analytics—at zero licensing cost [30].

The following observations from this review were directly applied to the architecture proposed in this work:

- Heterogeneous sources require a dedicated transformation layer to normalise data before it reaches the visualisation layer [6, 7].
- Separating raw and serving data warehouses isolates analytical workloads from source systems, preventing performance degradation.
- All reviewed touristic platforms lack automated, scheduled pipeline execution, making data freshness a manual operational concern.
- Governance controls (RBAC, RLS) are absent from all reviewed touristic platforms, limiting deployment in multi-stakeholder environments.

– Open-source tooling can meet the functional requirements of commercial BI tools without licensing costs [1, 30].

The primary technical contribution of this work is the architecture that automates data capture and transformation from heterogeneous tourism sources into a unified, governed analytical layer.

3 Problem Description

3.1 Project Context

The SmartDest project originates from the European Union's territorial cooperation framework. It was approved under the INTERREG VA Spain–Portugal MAC (Madeira–Azores–Canaries) Programme 2014–2020, with 85% of its funding provided by the European Regional Development Fund (FEDER). The programme aims to evaluate and propose strategies for converting the islands of the Macaronesia cooperation space (Azores, Madeira, Canary Islands, and Cape Verde) into Smart Tourism Destinations [37]. Ongoing discussions on the future of Madeira indicate that an economy based on sustainable tourism and the smart city concept will play a key role, given the lack of viable alternatives [38]. The SmartDest project thus combines these priorities: using EU cooperation to integrate information technologies into tourism, enable cooperation among stakeholders, and support the transition towards smart, sustainable tourist destinations.

The SmartDest platform is a framework designed to collect, analyse, and visualise tourism data for the Macaronesia region [37]. The project arose from the recognition that the tourism industry requires context-specific data to make informed decisions and adapt to evolving trends. Existing tourism data is often limited in scope, aggregated over long periods, and dispersed across various sources. This fragmented data makes it challenging for stakeholders to gain insights and make data-driven decisions. The SmartDest project aimed to address these challenges by creating a platform that consolidates data from diverse sources, offers varying levels of data detail, and supports advanced visualisation and correlation capabilities.

The SmartDest platform uses open-source tools and provides multiple access options to stored tourism data. The system is represented in the Figure 7 and is divided into three main components that interact with each other: data collection and storage, external data access, and a web platform for visualisation and extraction.

The core of the system manages the plugins responsible for data collection, which are executed at defined intervals based on their metadata. Each plugin defines the data fields and types, allowing for isolated management of data and tables. The platform uses a MySQL database, separating staging and warehouse databases to manage data collection and pre-processing efficiently. The API enables data access in JSON format, while the web platform provides user-friendly data visualisation.

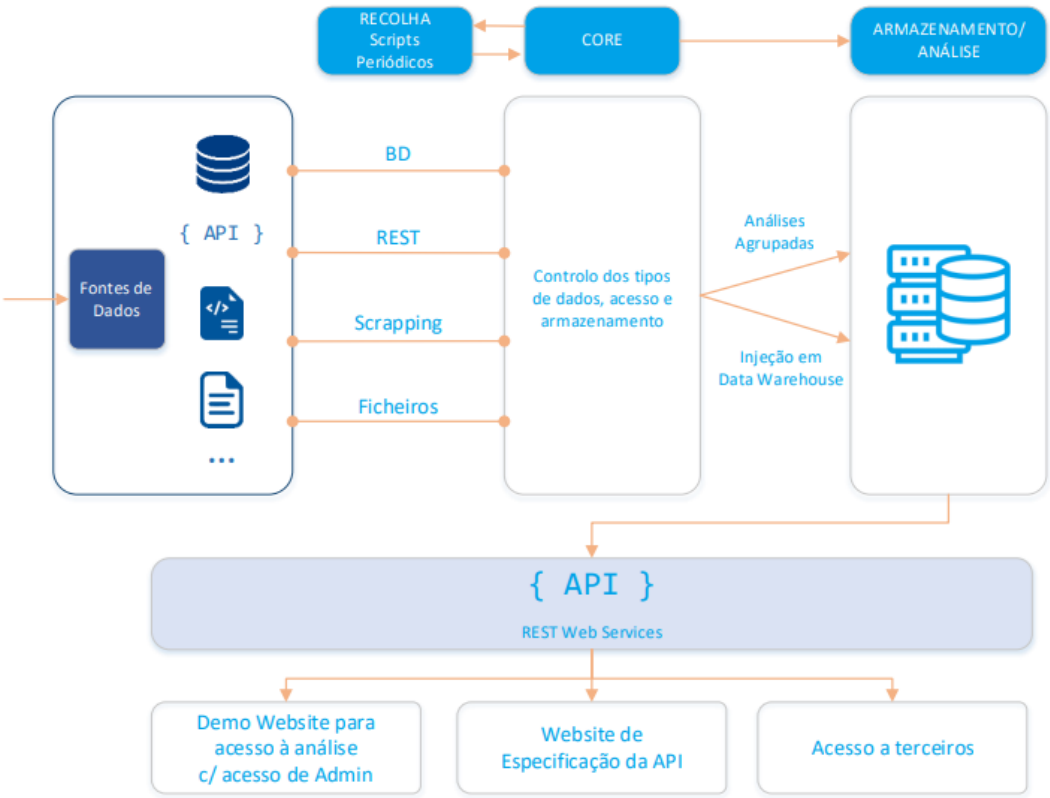

Fig. 7: SmartDest Architecture [37].

The platform gathers data from various sources, including airport and port movements, weather information, hotel indicators, social media activity, and tourist movement patterns, using methods such as APIs, web scraping, and partnerships with data providers. Built on an open-source philosophy, it ensures that tools and data are accessible, encouraging collaboration and enabling researchers to use the platform for diverse tourism studies.

The SmartDest project addresses major challenges in tourism data, including the need for context-specific methodologies and the difficulty of normalising data from varied sources. Traditional tourism statistics, often aggregated monthly or quarterly, lack the level of detail required for analysis. To tackle these limitations, the project provides a framework for collecting, transforming, and analysing data, while offering open access to support research and enable intuitive, multi-layered visualisation tools accessible through web pages and APIs.

Even with the benefits there are several problems that are highlighted by the project. One issue was the difficulty in normalising data from various regional, national, and international entities, which is a well-known problem in the tourism sector, as cited in BITOUR [19]. The project also

faced the challenge of ensuring that the data collected is not just extensive but also 'smart,' meaning that it should be applied and adapted to address real challenges and problems effectively. Furthermore, there was a noted lack of tourism statistics, which the project aimed to address through its data collection and analysis platform. The collection process itself presented difficulties, highlighting the need for continuous improvement in data quality and reliability. The platform also lacked a structured data transformation and serving layer. Without automated transformation, a dedicated serving database, and governance controls, the downstream visualisation capabilities were inherently limited. The following issues illustrate the consequences of this architectural gap.

One specific problem in the sources concerns the existing graphs. While the platform offers data visualisation, tourism professionals require more intuitive and user-friendly tools to manipulate and create visualisations effectively. For example, the existing graphs and visualisations lack some features and interactivity. The currently available visualisations may not be sufficient to meet the diverse needs of tourism stakeholders.

An example of a typical visualisation produced by the SmartDest platform is shown in Figure 8. The bar chart, titled "Sum of nr_flights vs company by year," displays the total number of flights for a wide array of airline companies. On the vertical axis, the sum of flights is quantified, while the horizontal axis lists numerous airlines. Airlines such as "TAP PORTUGAL," "EASYJET EUROPE," and "RYANAIR" are shown to have the highest flight volumes, with Swiss International significantly surpassing 4,000 flights. In contrast, most other airlines are represented by smaller bars, indicating a skewed distribution of flight operations. This type of chart, while informative, exemplifies some of the platform's limitations. It presents a static view of a large dataset, which can contribute to the performance issues noted during data loading. Furthermore, it lacks interactive features, such as filtering by year or drill-down capabilities, that tourism professionals require for more intuitive and dynamic analysis, reinforcing the critique that the visualisations are not sufficiently user-friendly.

Furthermore, the platform lacks dashboards. Although users can create visualisations, there are no pre-built dashboards to provide at-a-glance insights into key tourism indicators. This lack of dashboards may require users to invest time in creating custom visualisations for their needs.

Finally, performance issues have been noted when loading multiple data sources for visualisation. This slowdown can hinder the user experience and limit the platform's effectiveness in

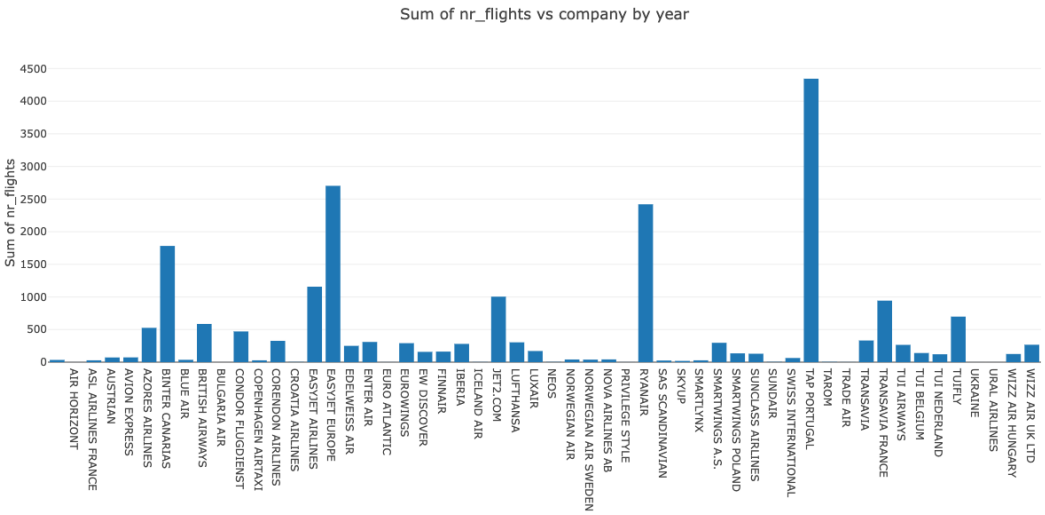

Fig. 8: Example of a chart on the SmartDest platform.

supporting real-time analysis and decision-making. To evaluate the system's performance, an alternative testing method was employed, acknowledging that it is not the best way to test performance. This was made when retrieving weather data from December 2019 to December 2024. During this process, it was observed that the loading time varied, with a maximum of 6,27 seconds and a minimum of 4,19 seconds and median of 4,87 seconds. This observation shows the performance when loading a 33,8 MB of data while retrieving a five-year span of weather data. This delay in loading data and rendering graphs emphasises the platform's scalability challenges. The increasing load times degrade the user experience, especially when switching between datasets or visualisations. Additionally, users face a lack of feedback during loading, leaving them uncertain about the platform's status. If these issues are not addressed, the platform may struggle to handle the growing volume and complexity of tourism data in the future.

The SmartDest project's primary objective was the conversion of islands into Smart Tourist Destinations. This was achieved through a two-phase approach, beginning with an analysis and strategic plan followed by the implementation of specific initiatives. The initial phase involved a Smart Tourism Diagnostic Study which identified opportunities for improvement and areas of resistance. Based on this study, a Strategic Plan for Smart Tourism was developed, including a conceptual model and a roadmap. This plan led to the selection of 25 initiatives, prioritised by criteria such as effectiveness and impact. The second phase focused on putting these initiatives into practice, with a focus on areas such as competency development, data management, and intelligent information systems, as well as creating tools such as the SMART platform, the tourist

data visualisation platform, and the App Caminho Real. Overall, the project aimed to improve tourism on Madeira Island through the strategic implementation of smart technologies and data-driven approaches.

The root cause underlying these issues is architectural: the platform lacks a structured transformation and serving layer. Without automated data transformation, a dedicated serving database, and governance controls, visualisation capabilities and performance are inherently constrained. Resolving the architecture is the prerequisite for improving the platform's analytical capabilities.

4 Methodology

This chapter details the methodology used to redesign the SmartDest tourism data platform. The process is divided into three phases: a heuristic evaluation of the existing platform to identify core limitations; the conceptual design and implementation of a new architecture using a modern data stack; and the empirical validation of the proposed solution through four targeted case studies.

The initial and most important phase of this methodology was to define a set of problems and requirements. This was achieved through a series of structured meetings with the key stakeholders interested in the platform's evolution, where their needs and frustrations with the existing system were gathered. The stakeholders consisted of the thesis supervisor (*orientador*), the co-supervisor (*co-orientador*), and a colleague from the same research area. This feedback was then organized to form a list of foundational requirements, as detailed in the following subsection, which then guided the design of the new conceptual architecture.

Following the platform evaluation, a new architectural solution was designed, incorporating a modern data stack to address the identified problems at each layer of the data lifecycle.

To validate the new architecture's ability to solve the identified problems, four case studies will be executed on a controlled development environment. For case studies 1, 2 and 3 a two-month sample of tourism data will be used. For case study 4, data from trail competitions (MIUT 2025) will be used directly from the database.

This multi-stage methodology ensures that the final proposed solution is both theoretically sound and empirically validated. By systematically quantifying SmartDest's deficiencies, designing an integrated data platform to address them, and then rigorously validating this new architecture through problem-focused case studies, the project aims to thoroughly confirm the new solution's ability to overcome the platform's initial limitations.

4.1 Platform Evaluation and Problem Identification

The formal evaluation process was centred on a series of structured meetings with these stakeholders, who were responsible for driving the platform's redesign. The objective of these sessions was to transition from a general understanding of the platform's issues to a documented set of specific, concrete problems.

A primary deficiency identified by the stakeholders was the platform's limited and inflexible visualisation capabilities. The static nature of the existing charts and the limited variety of chart types did not allow for meaningful comparative analysis. A requirement that appeared was the need for a wider selection of interactive and flexible visualisations that would allow users to dynamically filter, group, and drill down into the data, thereby improving usability and enabling deeper exploration.

Compounding this issue was the complete absence of a dashboard feature. The stakeholders emphasised that this was a major deficiency, as it made it impossible to create the cohesive, at-a-glance views needed to monitor tourism trends. The inability to effectively combine different datasets into a unified and performant dashboard was a major limitation that needed to be addressed, leading to the requirement for a platform capable of creating unified analytics from multiple integrated data sources.

During these discussions, it was reported that significant performance bottlenecks when data was from 4 years or more, which resulted in slow loading times and an unresponsive interface, made the platform frustrating to use and often discouraged deeper data exploration. A fundamental requirement for the new architecture, therefore, was to drastically improve data retrieval and rendering speeds to ensure a fast and reliable user experience.

A governance issue was also identified: the platform's inability to segregate data access between different users. The stakeholders required a system to easily manage and separate what data specific user roles could view, ideally controlled through a future API. This need for a data governance framework, including Role-Based Access Control (RBAC) and Row-Level Security (RLS), was coupled with a significant operational security concern regarding the lack of a unified authentication system. The stakeholders highlighted the inefficiency and potential security risks of managing separate credentials for each component of the data stack and expressed a requirement for a single, centralized login (SSO) to simplify the process of managing user access.

The need to integrate analytics into external workflows created the requirement for embeddability options. To extend the platform's value, the stakeholders required the ability to integrate interactive analytics and dashboards directly into other applications, a feature that was entirely absent from the legacy system.

These user-driven findings were combined into a set of requirements. The requirements are separated into two categories: Functional Requirements, which define what the system must do, and Non-Functional Requirements, which define the qualities and constraints under which the system must operate. These requirements, detailed in Table 3 and Table 4, directly guided the design of the new conceptual architecture.

Table 3: Functional Requirements

ID	Category	Description
Data Processing & Modeling		
FR1	Data Integration	The system shall be capable of joining data from multiple, distinct raw data sources into a unified model.
FR2	Data Transformation	The system shall provide capabilities for standardising, and aggregating the integrated data.
FR3	Data Cleaning	The system shall provide capabilities for cleaning the integrated data.
FR4	Automated Data Validation	The data processing workflow shall include automated tests for data quality and integrity (e.g., uniqueness, non-null constraints).
FR5	Workflow Failure	The data processing workflow must halt if any data quality test does not pass to prevent loading corrupt data.
Visualisation & Dashboards		
FR6	Chart Library	The system shall provide a diverse library of chart types, including bar, line, pie, table, and map visualisations.
FR7	Chart Customisation	Users shall be able to customise the aesthetic properties of charts, including titles, labels, and colour schemes.
FR8	Dashboard Creation	Users shall be able to create, save, and name new dashboards.
FR9	Dashboard Layout	Dashboards shall provide a customisable layout where users can arrange and resize charts and text elements.
FR10	Categorical Filtering	Dashboards shall support filtering by one or more categorical values from a given field.
FR11	Temporal Filtering	Dashboards shall support filtering by pre-defined or custom time ranges.
FR12	Cross-Filtering	Applying a single filter on a dashboard must update all relevant visualisations on that dashboard simultaneously.

Continued on next page

Table 3 (continued)

ID	Category	Description
FR13	User-Defined Columns	Users shall be able to define and add new calculated columns to a dataset using SQL expressions within the UI.
FR14	User-Defined Metrics	Users shall be able to define and add new aggregate metrics to a dataset using SQL expressions within the UI.
Security & Governance		
FR15	User Role Management	The system shall allow administrators to create and manage distinct user roles.
FR16	Asset Permissions	The system shall allow permissions for viewing, creating, and editing assets (charts, dashboards) to be assigned to user roles.
FR17	Row-Level Filtering	The system shall support filtering the rows of data a user can see based on their role and defined rules.
FR18	Centralized Authentication	The platform must provide a single, centralized authentication mechanism for all its components.
FR19	User Action Auditing	The system shall maintain a detailed and accessible audit log of user actions for security monitoring.
FR20	API Access	The system shall provide a public API for programmatic access to query data and manage assets.
Integration & Embeddability		
FR21	Secure Embedding	The system shall provide a secure, token-based mechanism to embed dashboards into external web applications.
FR22	Interactive Embedding	Embedded dashboards must remain fully interactive, allowing filter manipulation to update visualisations in real-time.

Table 4: Non-Functional Requirements

ID	Category	Description
Performance & Scalability		
NFR1	Dashboard Load Time	Average dashboard load times for common analytical queries shall be under 3 seconds.

Continued on next page

Table 4 (continued)

ID	Category	Description
NFR2	API Latency	To address legacy system latency, the API must exhibit an average latency of under 100ms for equivalent datasets.
NFR3	Scalability	The architecture must be designed to accommodate significant growth in data volume and concurrent users without significant performance degradation.
Usability & Maintainability		
NFR4	User Interface	The UI for creating charts and dashboards must be intuitive for non-technical users with a domain in tourism.
NFR5	Documentation	The system's configuration, deployment process, and API endpoints shall be clearly documented.
NFR6	Reproducibility	The entire platform deployment, including all services and configurations, shall be reproducible from configuration files.
NFR7	Version Control	All data transformation logic and infrastructure definitions shall be stored in files.
Reliability & Security		
NFR8	Automation	The data processing workflow shall be fully automated and scheduled to run on a daily basis.
NFR9	Failure Handling	The workflow orchestration system shall include mechanisms for retrying failed tasks.
NFR10	Alerting	The workflow orchestration system shall be capable of sending alerts upon pipeline failure.
NFR11	Transport Security	All web traffic and user sessions must be secured using HTTPS.
NFR12	API Security	Programmatic access to the platform's API must be secured using a token-based authentication system.
NFR13	Open Source	The core components of the technology stack should be open-source.

By combining these user-driven findings, a clear, evidence-based picture of the platform's limitations emerged. These identified problem, spanning visualisation, performance, data governance, and integration, were translated into a set of requirements. This approach ensured that the new conceptual architecture, detailed in the following section, was designed specifically to solve these defined problems.

4.2 Conceptual Architecture

The following introduces a new conceptual architecture for the entire data pipeline. While the initial stages of this pipeline, encompassing Data Sources, Data Extraction, and Data Processing, are considered part of a separate body of work, the scope of this thesis focuses specifically on the subsequent components. As such, this work details the Data Transformation stage, the complete Visualisation layer (which includes the Data API, Data Storage, and Data Visualisation), and the foundational Authentication layer. The proposed architecture is built on these distinct layers, designed to process raw data into a polished format and present it through a secure, performant, and interpretable user-facing interface.

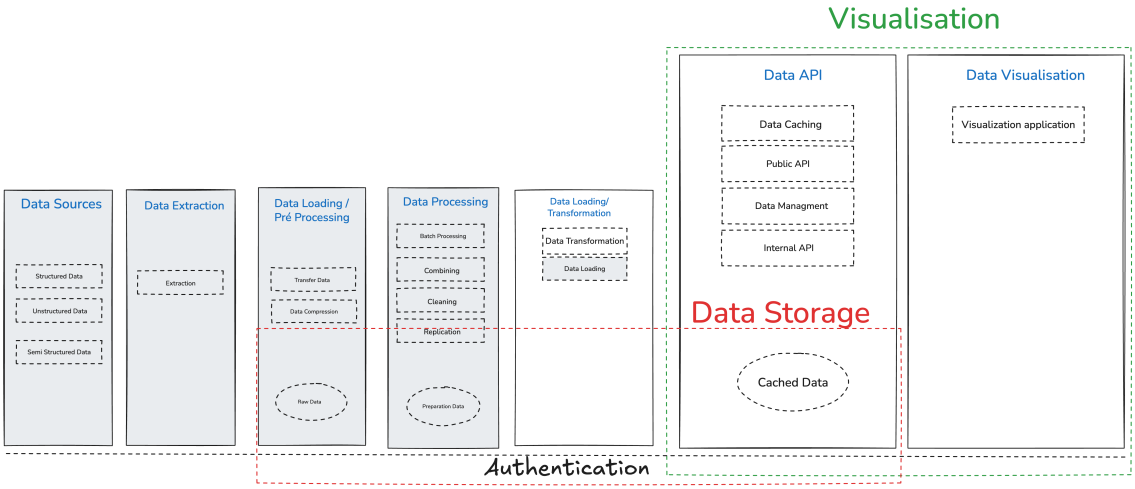

Fig. 9: Conceptual Architecture

The Data Transformation component represents an important preparatory stage within the defined scope of this pipeline. As shown in Figure 9, this phase is positioned after the initial data loading. Its primary responsibility is to process and refine the raw data into a structured, clean, and consistent format for analysis and presentation. This involves a series of operations such as cleaning to correct inconsistencies, normalising values, aggregating information, and enriching the dataset with additional calculated fields or joined data. The output of this module is a polished dataset, optimised for performance and ready to be served to the subsequent layers of the architecture, ensuring that the visualisations are based on accurate and reliable information.

The Visualisation section, demarcated by the green dashed line in Figure 9, represents the user-facing segment of the pipeline. Its fundamental role is to present the processed and transformed

data in a consumable and interactive format. This section is logically partitioned into two primary components that collaborate to achieve this objective: the Data API, which handles data access and retrieval, and Data Visualisation, which is responsible for the graphical representation of that data.

The Data API component acts as the central hub for all data requests, serving as an intermediary between the user-facing applications and the underlying data storage. It is composed of several distinct sub-modules designed to manage data flow efficiently and securely. These include Data Caching, a Public API, Data Management, and an Internal API. Collectively, these modules ensure that data is delivered in a structured, and secure manner.

A core module within the Data API is Data Caching. This sub-component is responsible for the temporary storage of frequently accessed data. By maintaining a cache, the system can improve its performance and responsiveness. The core objectives of this functionality are to minimise latency in data retrieval and to reduce the number of direct requests made to the primary database, thereby contributing to a more efficient and scalable system.

The Public API is designed as an external-facing interface. Its purpose is to provide a controlled and documented method for external users or third-party systems to interact with the platform's data. Through this API, external entities can retrieve datasets, access visualisation configurations, or integrate the system's data outputs into their own applications. This enables broader interoperability and extends the utility of the processed data beyond the immediate system.

Conversely, the Internal API serves as the system's private interface, used exclusively for its own operational needs. While it shares many functionalities with the Public API for data retrieval, the Internal API incorporates additional capabilities, particularly concerning security and administrative privileges. This allows it to handle sensitive internal operations and manage secure data access between the different components within the system's architecture.

The Data Management module, operating within the context of the Data API, is tasked with the administration of visual assets and access control. Its responsibilities include the processes for creating and configuring charts and dashboards, as well as handling the permissions associated with the data. This ensures that users can only view or interact with the data and visualisations for which they have been granted explicit authorisation.

The Data Storage layer underpins the Visualisation section. As depicted in Figure 9, its most relevant element to this architecture is the Cached Data. This represents the actual dataset that has been temporarily stored by the Data Caching module. Storing this data in a readily accessible cache allows the Visualisation application to manipulate and render it into charts and dashboards with minimal delay, avoiding the need to re-fetch the same information from the source database for every user interaction.

Following the Data API, the Data Visualisation module plays a role in the entire data exploration process. Its primary objective is transformative: to convert complex, structured, or semi-structured information into clear, interactive visual representations. This conversion is not merely for aesthetic purposes; it is an important step that enables users to explore vast datasets and identify patterns, correlations, and trends that would be hard to discern from raw data alone.

At the heart of this module is the Visualization application. This is the software component responsible for generating the final visual outputs, such as interactive charts, graphs, and consolidated dashboards.

Finally, the Authentication layer is presented as a foundational, cross-cutting concern that applies across the entire architecture, as indicated by the black dashed line in Figure 9. Its function is to secure the pipeline by verifying the identity of any user or service attempting to access the system. By enforcing strict access controls, this layer ensures that all interactions are authorised, thereby protecting the confidentiality and integrity of the data throughout the process from storage to visualisation.

4.3 Implementation Architecture

Having previously outlined the conceptual architecture, this section describes the practical implementation of the data analytics and visualisation platform. The implementation architecture, depicted in Figure 10, illustrates a modern data stack where dbt Core enables data transformation, Apache Airflow provides orchestration, and Apache Superset manages both the data API and visualisation layers. This architecture provides a concrete, tool-specific example of the end-to-end workflow, detailing how data flows from its raw state in a PostgreSQL warehouse through an automated daily pipeline to an interactive, user-friendly format ready for analysis.

The data journey originates at the Warehouse Raw Data repository. As illustrated in Figure 10, this component consists of a primary database, represented by a PostgreSQL instance, which acts

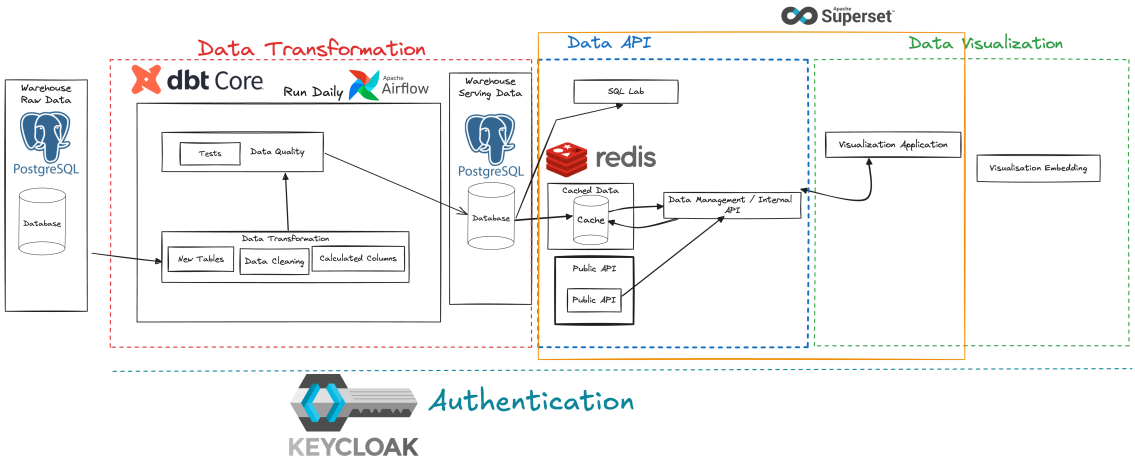

Fig. 10: Implementation Architecture.

as the foundational data store. This warehouse serves as the single source of truth for the data, containing the foundational, untransformed data upon which all subsequent analytical processes and models are built. The integrity and availability of this raw data layer are paramount for the entire pipeline.

The Data Transformation layer is the core engine for data preparation and modelling. This entire stage is driven by dbt Core, a modern data transformation tool, and is orchestrated by Apache Airflow. As indicated in the architecture, this automated pipeline is scheduled to run daily, guaranteeing that the analytical data is consistently and reliably refreshed without manual intervention. Dbt enables data transformation from a warehouse by simply writing SQL select statements, bringing software engineering best practices such as version control, testing, and documentation directly into the analytics workflow.

Within this automated daily run, raw data undergoes a series of structured transformations. These processes, as shown in the diagram, include data cleaning to handle inconsistencies, null values, or incorrect formatting, ensuring the data is accurate and reliable. Furthermore, transformations are used to create new tables, which often represent aggregated, filtered, or joined datasets that are modelled for specific business use cases. Calculated columns are also generated during this process to derive new metrics and features that do not exist in the source data, thereby enriching the dataset for deeper analysis.

An important aspect of this automated, dbt-driven transformation process is the emphasis on data quality. This is achieved through a combination of automated tests that are defined within

the dbt project and executed as an integral part of the daily run orchestrated by Airflow. These tests can range from simple non-null and uniqueness checks to complex, custom business logic validations. By embedding testing directly into the daily automated pipeline, data integrity is monitored, ensuring that the data feeding into the visualisation layer is trustworthy and accurate.

The final output of this daily transformation stage is materialised in the same Warehouse but a different data repository which is called the Serving Data. This repository, also a PostgreSQL database, is purpose-built and optimised for fast analytical query performance. Segregating the serving data from the raw data warehouse is an important design choice that decouples the analytical workloads from the transactional or source systems, preventing performance degradation and providing a stable, well-defined data source for all downstream applications. The daily refresh cycle managed by Airflow ensures that this serving database always contains up-to-date information for analysis.

The Data API serves as the intermediary layer between the prepared data in the serving warehouse and the end-user visualisation tools. This entire layer operates within the Apache Superset platform. It is responsible for efficiently querying the Warehouse Serving Data and delivering the results to the front-end components. This abstraction layer is vital for managing data access, improving performance, and ensuring a consistent interface for data consumption.

To ensure low-latency responses for dashboards and charts, a caching mechanism is implemented within the Data API layer. As shown in Figure 10, a Redis Cache is employed to store Cached Data resulting from frequently executed queries. When a user requests a dashboard, Superset first checks if the required data is available in the cache. If so, it is served directly from Redis, bypassing the need for a costly database query. This improves the user experience by providing near-instantaneous load times for popular visualisations.

Access to both the serving database and the cache is governed by an integrated Data Management / Internal API within Superset. This internal API orchestrates all data retrieval logic, determines whether to fetch data from the source or the cache, and manages cache invalidation strategies to ensure data freshness. The architecture also provisions a Public API, which provides a stable, documented, and secure REST interface. This allows external applications or advanced data science workflows to programmatically access the curated datasets and visualisations managed within Superset.

Complementing the automated API access is Apache Superset's SQL Lab. This feature provides a powerful and interactive SQL IDE directly within the platform. It allows data analysts and other proficient users to directly query the Warehouse Serving Data through the Data API layer, explore data schemas, prototype queries for new charts, and perform ad-hoc data exploration without leaving the Superset environment.

The Data Visualization layer is the front-end of the system where insights are ultimately consumed by business users. This layer is also an integral part of the Apache Superset platform. The core component is the Visualization Application, which is the primary user interface. It uses the internal API to request data and then renders it into charts, tables, and interactive dashboards, enabling users to interpret data and discover insights. The architecture also supports Visualisation Embedding, a powerful feature for extending the reach of the created analytics. This capability allows for the integration of individual charts or entire dashboards created in Superset directly into other external web applications, portals, or software products. This provides a solution for delivering embedded analytics, bringing data insights directly into the context of users' daily workflows.

Underpinning the entire platform is a centralised Authentication layer. As depicted in the architecture, this component is not tied to a single application but serves as a foundational, cross-cutting security service for the entire data stack, including the transformation, API, and visualisation layers. The purpose of this layer is to provide a unified and global authentication method, such as a Single Sign-On (SSO) solution using a provider like Keycloak. This approach externalises user identity and access management, meaning that a user authenticates once against this central service and is then granted access to the various tools—such as Airflow for orchestration or Superset for visualisation—based on a single, centrally managed set of credentials and permissions. This model improves security and simplifies user administration by eliminating the need to manage separate user accounts and passwords within each individual application.

After talking about the architecture it is important to talk about the core tool of this architecture, Apache Superset. The Figure 11 provides a detailed overview of the internal architecture of Apache Superset, a modern data exploration and visualisation platform. The architecture is organised into three main pillars: data connectivity and preparation on the left, the central application core in the middle, and data consumption and integration endpoints on the right. This modular

structure delineates how data is sourced, processed, managed, and ultimately presented or exposed for external use.

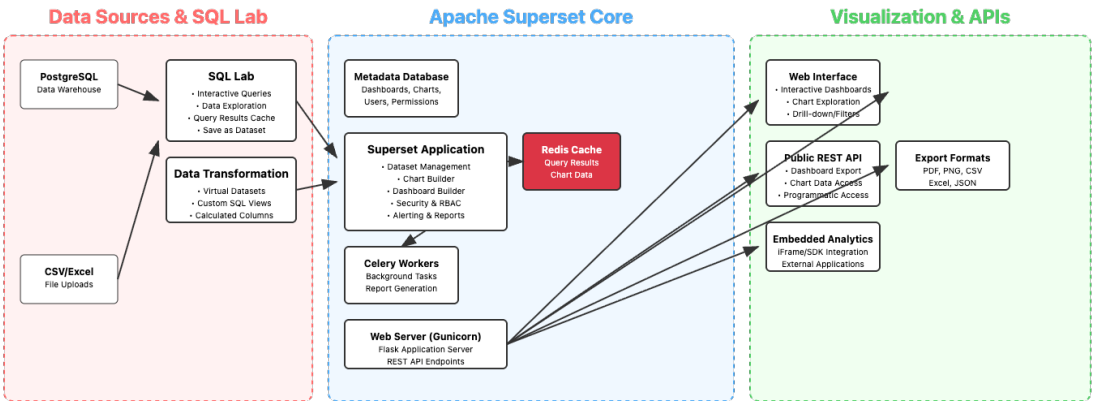

Fig. 11: Superset Internal Architecture taken from Apache Superset's official documentation and architectural patterns.

The Data Sources & SQL Lab pillar constitutes the data input and exploration layer. Superset can connect to a wide variety of data sources, with PostgreSQL shown here as a representative data warehouse. It also supports direct file uploads, such as CSV or Excel files, for rapid analysis of stand-alone datasets. The SQL Lab component is an integrated SQL IDE that allows users to perform queries, explore schemas, and browse query history. Importantly, this is also where data transformation occurs; users can define 'Virtual Datasets' from SQL queries, create 'Custom SQL Views', and add 'Calculated Columns' to enrich datasets logically without modifying the source, forming the semantic layer for analysis.

The central pillar is the Apache Superset Core, which orchestrates the platform's entire functionality. It is composed of several key components working in concert. The Metadata Database (e.g., PostgreSQL, MySQL) is foundational, storing all of Superset's state, including dashboard configurations, chart definitions, user accounts, and access permissions. The primary Superset Application, built on Python, manages dataset configurations, the chart and dashboard builders, and handles security aspects like role-based access control (RBAC), alerting, and reporting. Asynchronous or long-running tasks, such as report generation or complex query execution, are delegated to Celery Workers to prevent blocking the main application. All web traffic and API requests are handled by a Web Server (e.g., Gunicorn), which runs the core Flask application and exposes its REST API endpoints. A Redis Cache is employed to store query results and chart data, reducing the load on the underlying databases.

The Visualization & APIs pillar represents the output and integration layer, where data is consumed. The primary method of consumption is the Web Interface, which provides interactive dashboards, a chart gallery, and drill-down/drill-through capabilities for data exploration. For programmatic access and integration, Superset exposes a Public REST API, allowing developers to fetch chart data, manage dashboards, and query datasets externally. Furthermore, Superset supports various Export Formats, enabling users to download data or visualisations as PDF, PNG, CSV, or JSON files. Finally, its Embedded Analytics capability allows for the integration of Superset charts and dashboards into external applications, providing a solution for embedding data insights directly within other software frameworks.

The implementation architecture detailed above demonstrates a cohesive and modern data stack that translates the conceptual design into a functional system. By combining dbt Core for version-controlled transformations, Apache Airflow for automation, and Apache Superset as a unified visualisation and API layer, the system ensures a reliable and performant daily data refresh. Key design choices, such as the separation of raw and serving data warehouses and the strategic implementation of a Redis cache, directly address performance and scalability. This tool-specific blueprint, ending in a deep dive into Superset's modular internal structure, establishes a complete, automated, and secure pipeline from raw data to end-user insight.

4.4 Architectural Rationale and Discussion

The following explores the architectural rationale and key decisions that guided the development process, prioritising maintainability, scalability, and automation. It specifically addresses the choice of adopting a modern data stack, using dbt Core for the transformation layer and Apache Airflow for orchestration, alongside the strategic adoption of an Infrastructure-as-Code (IaC) approach for platform deployment. The development process was guided by several key architectural decisions that prioritised the creation of a modern, automated data platform. The rationale behind these choices and the benefits they provide over more traditional approaches are examined below.

A key architectural decision was to decouple the data transformation logic from the data warehouse engine by adopting dbt Core. This approach externalises all transformation logic into version-controlled SQL and configuration files, bringing software engineering best practices—such as modularity, testing, and documentation—directly into the analytics workflow. This stands in contrast to embedding logic within database-specific views or functions. By using dbt, the system gains a

highly maintainable and scalable transformation layer. It allows for complex data models to be built, tested, and iterated upon independently of the database, and ensures that the Warehouse Serving Data is always a reliable, well-documented, and validated source for Apache Superset.

To ensure the reliability and timeliness of the data, Apache Airflow was integrated to orchestrate the transformation pipeline. As depicted in the implementation architecture, Airflow is responsible for scheduling and executing the dbt models on a daily basis. This automates the entire process of data extraction, transformation, and quality testing, ensuring that the Warehouse Serving Data is consistently refreshed with validated, up-to-date information. Airflow provides logging, monitoring, and alerting capabilities, making the pipeline transparent and easy to manage.

A second key architectural principle was the adoption of an Infrastructure-as-Code (IaC) approach, managing the entire platform deployment, including PostgreSQL, dbt Core, Airflow, and Superset—through version-controllable files (e.g., docker-compose.yaml and associated configuration scripts). This makes the platform highly reproducible and transparent, standing in stark contrast to manual setup processes. It allows for automated deployments, ensures consistent environments between development and production, and provides clear, auditable tracking of any configuration changes over time.

The architectural choices reflect a pragmatic philosophy, balancing current project needs with long-term maintainability and strategic scalability. Adopting a decoupled transformation layer with dbt Core provides modularity and testing, while Apache Airflow ensures reliable automation. The overarching adoption of an Infrastructure-as-Code approach ensures reproducibility and consistent environments, collectively establishing a modern, adaptable data platform.

The project established a validated framework for data visualisation, directly addressing the identified shortcomings of the SmartDest platform. This was achieved through a multi-phase methodology, which began with a formal heuristic evaluation of the existing system to quantify its principal limitations, leading to the strategic selection of Apache Superset as a highly suitable, zero-cost replacement due to its rich features including customisable visualisations, a dashboard builder, and a powerful semantic layer.

The project laid out a clear conceptual architecture for the data management pipeline, emphasising the transformation of processed data into accessible and interpretable visualisations. This conceptual framework was then translated into a concrete implementation architecture, demon-

strating a scalable and automated data flow from the central warehouse, through a transformation pipeline orchestrated by Apache Airflow using dbt Core, an efficient API layer with Redis caching, and ending in user-focused visualisations in Apache Superset. The underlying architectural rationale prioritised maintainability and scalability, using a modern data stack and adopting an Infrastructure-as-Code approach to ensure reproducibility and operational excellence.

Taken together, the endeavour from initial problem identification through implementation and validation has resulted in a system that ensures efficient data flow and improved user accessibility. By using Apache Superset on a foundation of automated and validated data transformations, the project provides an intuitive means for users to explore vast amounts of information, thereby consistently promoting strategic insights and informed, evidence-based decision-making across all levels of the system.

4.5 Data Transformation and Orchestration

The following details the core engine of the analytics platform: the automated data transformation and orchestration pipeline. This pipeline is driven by the combination of dbt Core, which defines the logic for data modelling and quality, and Apache Airflow, which manages the scheduled, reliable execution of these processes. Together, these tools establish a modern, test-driven ETL workflow that ensures all data is systematically cleaned, transformed, and validated before it becomes available for analysis, forming a resilient foundation for the entire visualisation layer.

dbt (data build tool) enables a modular, testable SQL-based transformation methodology, providing lineage tracking and environment-aware compilation. Its declarative testing capabilities are used in the early detection of schema violations and data anomalies, thereby improving data quality throughout the pipeline. Apache Airflow, serving as a complementary tool, offers a scheduler, sophisticated dependency management, and essential observability primitives. The combined application of dbt and Airflow establishes a modern Extract, Transform, Load (ETL) approach. In this architecture, data is extracted from the raw data repository, transformed and then loaded into the final serving data warehouse. This approach improves the reproducibility and auditability of the entire data workflow.

Within this project's methodology, dbt assumes a central role in the data modelling and transformation layer for all tourism data sources, which encompass various types and levels of detail. It is systematically employed to normalise raw inputs from different origins into multiple single, column-

aligned models, ensuring consistent query semantics and enabling simpler cross-series analysis. For instance, a sparse-column approach is adopted across these models, where attributes irrelevant to a particular upstream source are explicitly set to NULL, and a source_label field indicates the data's origin, thereby preserving data fidelity. Importantly, declarative testing via schema.yml files (enforcing constraints such as not_null, unique, and accepted_values) and singular SQL tests for domain-specific rules are fundamental to the dbt process, giving immediate feedback on data quality and ensuring conformance to predefined constraints for each model.

Airflow serves as the primary orchestration engine for the entire data transformation pipeline, encompassing all 17 identified tourism data sources. It ensures the scheduled and automated execution of dbt processes by managing Directed Acyclic Graphs (DAGs). Each DAG encapsulates a precise sequence of ordered tasks for specific dbt models, typically including `dbt deps` to manage macro packages, `dbt ls` for connectivity verification, `dbt run -select <model_name>` to materialise the models, and `dbt test -select <model_name>` to validate data integrity post-transformation. This orchestration strategy adheres to principles such as minimal sequences per model to isolate defects, incorporates retries for improved reliability, and applies bounded concurrency (e.g., single active run, no catch-up) to prevent redundant recomputation and ensure efficiency. Furthermore, Airflow's observability primitives are vital, providing detailed inspection of task logs that offer insights into dbt output, row counts, and test outcomes, which are essential for monitoring and recovery operations.

The strategic integration of dbt for transformation and Airflow for orchestration provides a solution for managing the data pipeline. By using dbt's declarative testing and modular SQL-based modelling, the project enforces data quality and consistency at the source. Airflow complements this by wrapping the dbt processes in automated, and observable workflows, ensuring that all 17 tourism data sources are refreshed daily in a reliable and efficient manner. The outcome is a fully automated and auditable pipeline that guarantees the data feeding the visualisation layer is timely, trustworthy, and consistently analytics-ready.

4.6 Application Choice

The following outlines the rationale behind the selection of the core technologies for the data platform. The decision-making process extended beyond a simple evaluation of technical features to incorporate important long-term factors such as total cost of ownership, operational efficiency,

and strategic flexibility. It details why Apache Superset was chosen for the visualisation layer, dbt Core for data transformation, and Apache Airflow for orchestration, explaining how these specific tools were selected to form a cohesive, modern, and sustainable data stack tailored to the project's needs.

Beyond its technical features, the selection of Apache Superset as the core tool was guided by a set of key strategic criteria to ensure its long-term viability for the project. The primary factors scrutinised in this methodological choice included the total cost of ownership, assessing not only the absence of licensing fees but also anticipated implementation and maintenance costs. Additionally, the platform's operational efficiency in providing self-service analytics and its strategic flexibility as an open-source solution were important considerations.

The primary factors considered in this methodological choice were:

- Total Cost of Ownership: Evaluating not just the absence of licensing fees, but also the anticipated costs related to implementation, infrastructure, and the required technical expertise for maintenance.
- Operational Efficiency: The platform's ability to provide self-service analytics was an important factor, aiming to enable end-users and reduce the dependency on manual data preparation.
- Strategic Flexibility: Prioritising an open-source solution to avoid vendor lock-in, which provides complete control over the platform's future development and ensures it can be extended to meet the specific, evolving needs of the SmartDest project.

Based on this multi-faceted evaluation, Apache Superset was definitively confirmed as the most suitable technology for the visualisation layer. Its strategic selection, driven by its zero-cost licensing model, visualisation library, dashboard builder, and sophisticated semantic layer capabilities, directly addresses all identified needs of the SmartDest platform for data presentation and interactivity. Superset provides an intuitive means for users to explore vast amounts of information, thereby consistently promoting strategic insights and informed, evidence-based decision-making across all levels of the system.

dbt Core's selection was for the data transformation and modelling layer, complementing Superset by ensuring the data it consumes is clean, structured, and analytics-ready. dbt promotes a modular, testable SQL-based transformation approach, which is essential for managing the complexity of diverse tourism data sources. It provides lineage tracking and version control for all SQL

transformations, which is important for ensuring reproducibility and auditability. Furthermore, dbt's declarative testing capabilities allow for the proactive identification of data quality issues at the transformation stage, ensuring that only high-quality data is presented in Superset dashboards.

Finally, the selection of Apache Airflow as the workflow orchestrator was central to operationalising the entire data pipeline. While dbt defines what transformations need to occur, Airflow determines when and how they are executed. Its primary role in this architecture is to provide schedule-based automation for the daily execution of the dbt jobs. This guarantees that the Warehouse Serving Data is consistently refreshed, ensuring data timeliness for end-users in Superset without any manual intervention. Beyond simple scheduling, Airflow offers a full suite of features for production environments, including a detailed user interface for monitoring job status, extensive logging for troubleshooting, and sophisticated mechanisms for handling retries and alerting on failures. The choice of Airflow thus completes the modern data stack, providing the essential automation and operational oversight required to manage the data lifecycle reliably and at scale.

The selection of dbt Core, Apache Airflow, and Apache Superset was driven by the need for a scalable, automated, and zero-licensing-cost architecture. The next chapter details the technical implementation and configuration of this new data visualisation environment.

5 Development

This chapter describes the implementation and configuration of the new data visualisation architecture. It details the deployment of the containerised application stack, the development of the automated data transformation pipeline, the setup of the semantic and security layers, and the workflow for creating dashboards.

5.1 Infrastructure Deployment and Configuration

The following outlines the deployment and configuration of the application stack. It details the containerised architecture that orchestrates the distinct services for each application, including the central web applications, metadata databases, and asynchronous task workers. The configuration process, managed via version-controlled files following an IaC approach, established the environment for subsequent development.

The initial step was the deployment of the full data platform. The entire stack, including Apache Superset and Apache Airflow, was deployed using a containerised architecture managed by Docker Compose. This approach orchestrates the various services required for each application. For Superset, this includes its central web application, a PostgreSQL metadata database, Celery workers for asynchronous tasks, a scheduler, and a Redis instance that serves as both a message broker and a caching back-end. Similarly, the Airflow deployment includes its web server, scheduler, metadata database, and workers. The dbt Core CLI is installed within the Airflow environment, allowing Airflow DAGs to execute dbt commands.

The configuration of each component is governed by central, version-controlled files. For Superset, the superset_config.py file dictates all settings, from security features to database connections. For dbt, the dbt_project.yml file defines the project structure and model configurations. For Airflow, the DAG Python files define the orchestration logic. The entire setup is defined in a docker-compose.yaml file, which uses depends_on conditions to ensure a proper startup sequence. This Infrastructure-as-Code approach created a reproducible and scalable environment for analytical assets. The detailed configuration files and setup instructions for the Superset instance are available in Appendix A, while the full configuration for the Airflow and dbt environment is provided in Appendix B.

The development established a scalable infrastructure for the Superset application. The platform was configured to use a PostgreSQL database for its metadata repository, as defined by the SQLALCHEMY_DATABASE_URI, which stores all essential application state such as dashboard configurations, user permissions, and chart definitions. For performance and scalability, Redis was integrated as a multi-purpose service layer. It was configured as the primary query results backend (RESULTS_BACKEND), accelerating dashboard load times by caching the output of expensive queries. Furthermore, Redis was implemented for server-side session management (SESSION_TYPE = "redis"), an important requirement for maintaining user sessions consistently across multiple web server instances in a load-balanced production environment. Finally, the Celery workers, responsible for handling asynchronous tasks, were configured to use Redis as both the message broker and result backend (CELERY_CONFIG), ensuring that long-running jobs like report generation or data caching do not impact the responsiveness of the main web application.

For programmatic and embedded access, a JSON Web Token (JWT) strategy was configured in Superset. This included defining secrets for embedded analytics and guest tokens, essential for securely integrating dashboards into external applications. The security model was further hardened by enabling role-based access control at the dashboard level (DASHBOARD_RBAC), ensuring users can only view assets for which they have explicit permission.

The Superset platform was further tailored by strategically enabling specific features and configuring web server settings for secure integration. A set of FEATURE_FLAGS was activated to unlock essential production capabilities, including EMBEDDED_SUPERSET, ALERT_REPORTS, TAGGING_SYSTEM, and GLOBAL_ASYNC_QUERIES. The latter is particularly important for stability, as it offloads long-running queries to the asynchronous Celery workers, preventing web server timeouts and improving the user experience. To enable the integration with external front-end applications, Cross-Origin Resource Sharing (CORS) was enabled and configured with a specific whitelist of origins (CORS_OPTIONS), allowing secure communication between the visualisation application and the Superset back-end API. While the security middleware Talisman was not fully enabled, its configuration (TALISMAN_CONFIG) was prepared with a Content-Security-Policy to explicitly permit the embedding of dashboards within the designated frontend domains.

An important aspect of the security architecture is authentication. During development, a unified, single sign-on (SSO) solution across all applications (Superset, Airflow, etc.) was not implemented due to complexity and scope constraints. Instead, each application was configured with its own local authentication system. Users requiring access to Superset have credentials managed directly within Superset, while users needing to access the Airflow UI have separate credentials managed by Airflow. This approach ensures that each application remains secure on an individual basis, although it requires users to maintain separate logins for each component of the data stack.

The infrastructure deployment resulted in an operational, containerised platform. By using Docker Compose and version-controlled configuration files, a reproducible and secure environment was established, comprising Apache Superset, Airflow, and dbt Core, supported by PostgreSQL and Redis. The specific configurations for performance caching, asynchronous task handling, security features like JWT and RBAC, and integration settings like CORS, collectively created a foundation ready for the subsequent stages of data pipeline integration and analytical asset development.

5.2 Data Integration and Transformation Pipeline

With the infrastructure deployed, the development focus shifted to populating the platform with analytics-ready data. The automated data integration and transformation pipeline for all 17 tourism data sources was implemented as follows. dbt Core was used to define the transformation and data quality logic, and Apache Airflow was configured to orchestrate this process, creating a reliable and automated daily ETL workflow that feeds the visualisation layer.

The core of this architecture is an automated, daily pipeline that transforms raw data into analytics-ready datasets. Instead of using PostgreSQL functions or views, this pipeline is managed entirely by dbt Core, with orchestration provided by Apache Airflow. Raw data from the Warehouse Raw Data is processed by a series of dbt models. Each model, a SQL SELECT statement, defines the logic for cleaning, joining, and aggregating the data. These models are configured to materialise as physical tables in the Warehouse Serving Data database.

To ensure the quality and reliability of these transformed datasets, dbt's declarative testing capabilities were used. Generic tests (e.g., not_null, unique, accepted_values) are defined in .yml files alongside the models to enforce basic data integrity constraints. Additionally, singular SQL-based tests were developed to validate more complex business logic.

The orchestration of this entire pipeline is managed by Apache Airflow. A dedicated Airflow Directed Acyclic Graph (DAG) is configured to run the dbt jobs daily. This DAG executes a sequence of commands: first dbt run to execute the transformation models and build the tables in the serving warehouse, followed by dbt test to run all the defined data quality tests against the newly created tables. This automated orchestration ensures that the datasets are refreshed consistently every day and that any data quality issues are detected promptly, providing an auditable ETL workflow that feeds Superset with high-quality, usable data. The complete set of configuration files for this pipeline, including the Airflow Docker setup and the dbt project files, are detailed in Appendix B.

The process for creating each new curated dataset follows a systematic workflow. It begins with defining the business requirements. Then, the dbt model is created using readable SQL, explicit typing, and deduplication logic. Alongside the model, schema definitions and tests are created in .yml files. The model is iteratively built and tested locally until all checks pass. Finally, a dedicated Airflow DAG is created to automate the daily execution of the dbt run and dbt test commands for that specific model, ensuring it becomes part of the daily automated refresh cycle. This workflow ensures a maintainable, testable, and observable pipeline for all data transformations.

This phase established an automated data transformation pipeline. The combination of dbt for modular, test-driven transformations and Airflow for daily orchestration ensures that the serving data warehouse is consistently populated with clean, validated, and analytics-ready data. The systematic workflow defined for creating new datasets provides a maintainable and scalable process, resulting in a data foundation for the semantic and visualisation layers.

5.3 Security Layer Implementation

With transformed data tables being created daily in the serving warehouse, the development proceeded to implementing a security and semantic layer within Superset. This abstract layer serves as the intermediary between the physical, materialised dbt tables and the end-user, covering both the creation of a flexible data modelling environment and the configuration of a fine-grained access control model to ensure secure and appropriate data governance. Virtual datasets can also be created using Superset's SQL Lab, allowing for the definition of custom views through SQL queries that could join multiple dbt tables or apply final, presentation-layer transformations. Furthermore, new calculated columns and custom metrics were defined directly within the dataset settings in

Superset, enabling analysts to create derived variables on-the-fly without altering the underlying dbt models.

For access control within Superset, RLS exists to ensure secure data governance. Following the roles defined in the methodology, distinct user profiles such as 'Admin', 'Alpha' (analyst), and 'Gamma' (viewer) were configured. Each role was assigned a specific set of permissions dictating their ability to view, edit, or create dashboards, charts, and datasets. This fine-grained control ensures that users can only access the data and functionalities appropriate for their role, establishing a secure, multi-tenant environment ready for deployment.

The implementation of this security layer established a secure, multi-tenant analytics environment. By constructing a semantic layer with virtual datasets and calculated metrics, the platform provides a data modelling experience for analysts. This was complemented by the configuration of a fine-grained Role-Based Access Control model with distinct user profiles, ensuring that data access is governed by clear and enforceable permissions, making the platform ready for secure consumption by different user groups.

5.4 Data Visualisation and Dashboard Creation

The following describes the workflow for translating curated datasets into analytical assets for end-users. The development of analytical assets from curated datasets follows a structured, multi-stage process: creation of individual charts, assembly into dashboards, incorporation of interactivity, and performance improvement and publication. Each step is designed to be systematic and repeatable, ensuring consistency and quality across all data visualisations.

The initial procedure is the creation of individual charts within Superset's 'Explore' interface. This process commences by selecting a curated dataset as the data source. A suitable visualisation type is then chosen from the available library, and the chart is configured by mapping dataset columns to the required metrics and dimensions. Further customisation, such as adjusting titles, labels, and colour schemes, is applied to ensure the chart effectively communicates its intended insight.

Following the creation of individual charts, they are assembled into a cohesive dashboard. Superset's dashboard builder provides a drag-and-drop grid layout, which is used to arrange charts, text boxes, and other components into a logical narrative. Charts, referred to as slices, are resized

and repositioned to create a visually organised and intuitive layout that guides the user through the data story.

To enable data exploration, interactivity is integrated into the dashboards through the implementation of native filter components. Filters, such as time range selectors or drop-down lists, are added to the dashboard and linked to the relevant data columns. This configuration allows multiple charts to be updated simultaneously in response to a user's filter selection, transforming the dashboard from a static report into an interactive analytical tool.

The final stage of the workflow involves optimising and publishing the dashboard. Performance is improved by configuring caching options at both the dashboard and individual chart levels, which allows frequently accessed query results to be served from the Redis backend, thereby reducing load times. After a thorough verification of functionality, interactivity, and performance, the dashboard is published. This action makes it accessible to end-users who possess the appropriate 'Gamma' role permissions, completing the process from dataset creation to a fully operational, interactive analytical asset.

This workflow provides an end-to-end process for the creation of analytical assets. By moving sequentially from chart creation and dashboard assembly to the implementation of interactive filters and performance caching, the process ends with the publication of fully operational and performant dashboards. This approach ensures that visualisations are accessible and reliable for their intended audience.

The development phase translated the conceptual architecture into a functional platform. By using Docker Compose, dbt, Airflow, and Superset, an end-to-end data pipeline was established. The resulting environment is now prepared for the empirical evaluation detailed in Chapter 6.

6 Evaluation

The following evaluates the new architecture's effectiveness in resolving the limitations of the original SmartDest platform. The analysis is structured around four case studies, each addressing a specific problem identified in the review: the inflexibility of visualisations, the difficulty of integrating heterogeneous data sources, the need for data governance, and the challenge of embedding analytics into external applications. Each case study documents the approach taken to solve the problem, the challenges encountered, and the results observed, resulting in an assessment of the new solution's capabilities.

6.1 Case Study 1: Recreating a SmartDest Chart

A deficiency of the SmartDest platform was its static and inflexible visualisations, which limited comparative analysis. This case study aimed to solve this problem by recreating a core visualisation in a way that would provide deeper, more interactive analytical capabilities. The goal was not merely to replicate the original chart, but to improve its analytical value by making it a dynamic tool for exploration. The dataset used in this evaluation covered flight arrivals and departures at Madeira Airport, sourced from the SmartDest data portal. It included fields such as timestamp, airline, origin or destination city, flight status, flight type (arrival or departure), and number of flights. This dataset provided a relevant foundation for addressing SmartDest's tourism-related analysis limitations.

To begin solving the problem of data inflexibility, the data was first modelled to allow for the creation of derived metrics and calculated columns directly within the analytical layer. This addressed a notable limitation in SmartDest, which lacked built-in support for such on-the-fly transformations. This step enabled a broader exploration of the dataset than was previously possible.

The process of creating a new chart immediately highlighted the benefits of a more dynamic approach to solving usability issues. The new interface provided real-time feedback, allowing the analyst to see the impact of modifying filters or groupings instantly. This process contrasted with the legacy system, where the outcome of changes was not always clear.

Figure 12 shows the visualisation as rendered by the original SmartDest platform. This chart exemplifies the platform's core analytical problem: its flattened structure treats each combination

of flight type and year as a distinct category (e.g., 'arrival-2019', 'departure-2020'). This design makes it impossible to perform comparative analysis, such as directly comparing all arrivals across multiple years, and scales poorly as new data is added.

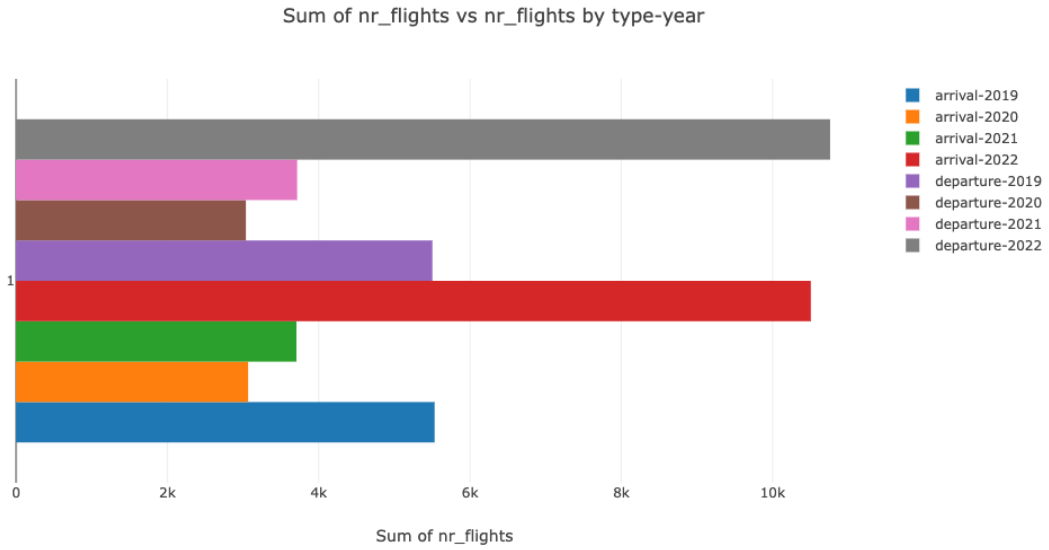

Fig. 12: Graph of Original SmartDest with title Sum of nr_flights vs nr_flights by type-year.

In contrast, the chart recreated within Apache Superset, represented by Figure 13, employs a more structured and analytical design. The choice of a grouped bar chart was deliberate, as outlined by Robert Spence in Information Visualization, this type of chart is exceptionally effective for comparing sub-categories (in this case, years) within main categories (arrivals and departures) [39]. This vertical grouped bar chart organises the data first by the primary category ('arrival' and 'departure') on the x-axis. Within each of these two groups, individual bars represent the sum of flights for each year, distinguished by colour. This grouped structure provides a much clearer structural comparison between arrivals and departures across the entire time period. A functional advantage of the Superset version is the interactive legend at the top, which allows the user to dynamically select or deselect years to filter the visualisation. This feature enables a more focused and flexible analysis that is not possible with the static SmartDest chart.

Two chart versions were ultimately created to demonstrate the solution's flexibility: one that closely replicated the legacy format for continuity, and the second, improved version that used the new grouping and filtering capabilities. This underscored the platform's ability to solve for both legacy replication and advanced analytical use cases.

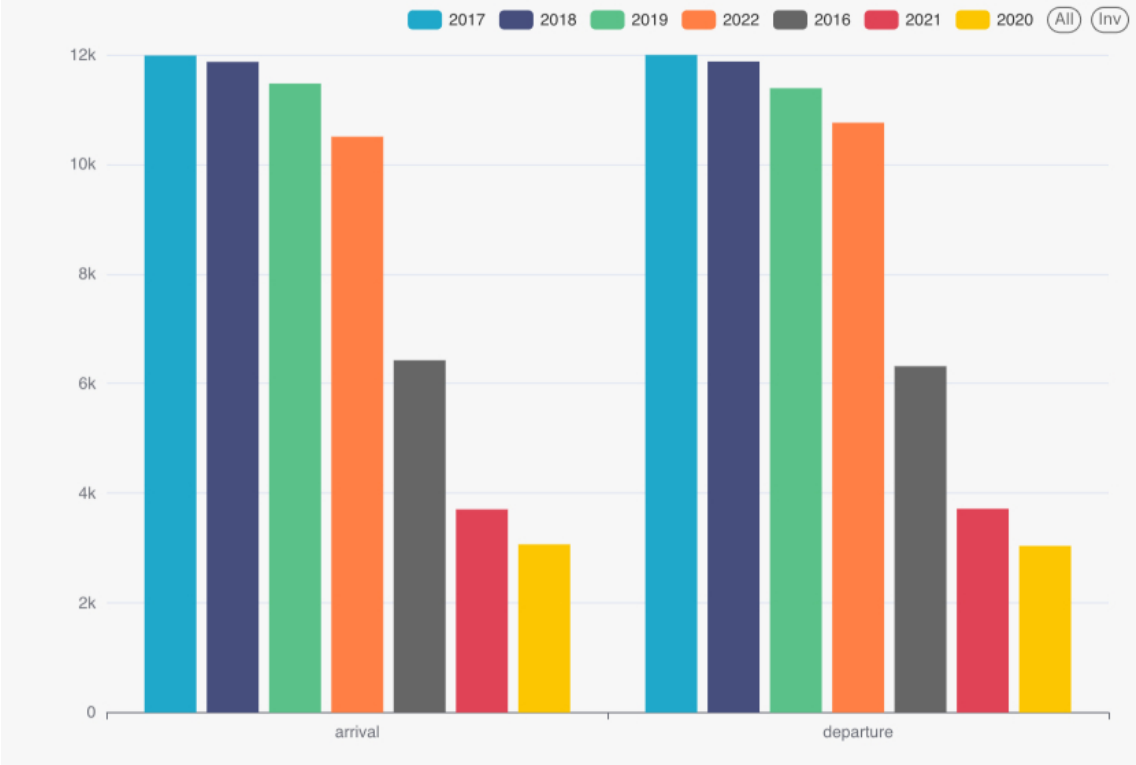

Fig. 13: Graph of Superset adapted from Original SmartDest.

Beyond the improvements in analytical capability, it was important to quantitatively measure whether the new architecture improves upon the performance limitations of the original platform. To achieve this, a direct performance comparison was conducted using the HTTP benchmark tool wrk to simulate concurrent user load against both the legacy SmartDest API and the new platform's API, with both systems querying the same underlying database [40] [41]. The test was configured to run for 30 seconds with 12 connections across 4 threads. The full configuration and raw output of this benchmark test are available in Appendix C. The results showed a dramatic improvement: the legacy API handled an average of 1.00 requests per second with a high average latency of 10.10 seconds, while the new architecture handled 3.23 requests per second with an average latency of just 43.64 milliseconds. This represents a 223% increase in throughput and a latency that is over 230 times lower, confirming the new architecture provides a more performant and scalable solution.

This case study confirms that the new architecture resolves the visualisation inflexibility and performance issues that limited the old platform. The new system not only achieves functional parity with the original SmartDest chart but also delivers significant improvements. While the legacy platform was simple for basic, predefined tasks, it fundamentally lacked the advanced ana-

lytical features required for deeper inquiry. In stark contrast, the new architecture enables tourism professionals to model data on-the-fly, derive new variables, and build truly interactive, multi-dimensional charts on a high-performance backend. These improvements provide a more capable solution for data exploration and address the need for analytical tools.

6.2 Case Study 2: Two Data Sources

A challenge in tourism analytics is the integration of disparate data sources to form an overall view, a task the original platform struggled with. This case study addressed this problem by attempting to create a unified dashboard from multiple, distinct datasets, testing the limits of data harmonisation when explicit join keys are absent. Success was measured by the ability to create a dashboard with functional cross-filtering and acceptable performance.

The evaluation was structured as a two-part test to understand the boundaries of the solution. The first part aimed to solve the integration problem for three contextually related tourism datasets (guest arrivals, revenues, and statistics by country). The second part attempted to solve the same problem for three structurally heterogeneous environmental datasets (wind and sea temperature) to identify the limits of the technology. To solve this, the architecture moved away from slow, on-the-fly virtual queries towards a more performant ETL process where data models are pre-calculated as physical tables using dbt.

The first part of the evaluation focused on solving the integration problem for the related tourism datasets: guest arrivals (source_4), revenues (source_5), and annual statistics (source_6). The solution involved creating a dedicated dbt model to join these three source tables on their common year and total fields. By configuring the model to be materialized as a table, dbt pre-computes the result and stores it as a new, physical table in the data warehouse. This pre-aggregated table, containing all necessary metrics, is then used to power the dashboard, a key part of the solution to the performance and data blending problem.

Similarly, the second test attempted to solve the integration problem for the heterogeneous environmental datasets: weather (source_1), warnings (source_2), and sea conditions (source_3). A dbt model was again created to join these datasets by date and location, aggregating various metrics into a single, materialised table. To improve query performance on this larger dataset, the model configuration explicitly includes database indexes on the date_forecast and location

columns. This demonstrates a further level of optimisation available within the dbt framework to ensure the technical part of the solution is as efficient as possible. Both of this models can be found at D.1 and D.2.

To ensure the data in these new tables remains current, an Apache Airflow DAG is scheduled to run these dbt models daily. This automates the entire refresh process, guaranteeing that the dashboard is always powered by fresh, validated, and performant data. This automated ETL process solves both dashboard performance and long-term maintainability.

The integration of the related tourism datasets was successful, demonstrating a viable solution to the problem. The dashboard's average load time was measured at 2.6 seconds, a result directly attributable to querying the pre-aggregated physical table. As shown in Figure 14, a coherent dashboard was created where a global "Year" filter controls all charts simultaneously, providing a unified analytical view.

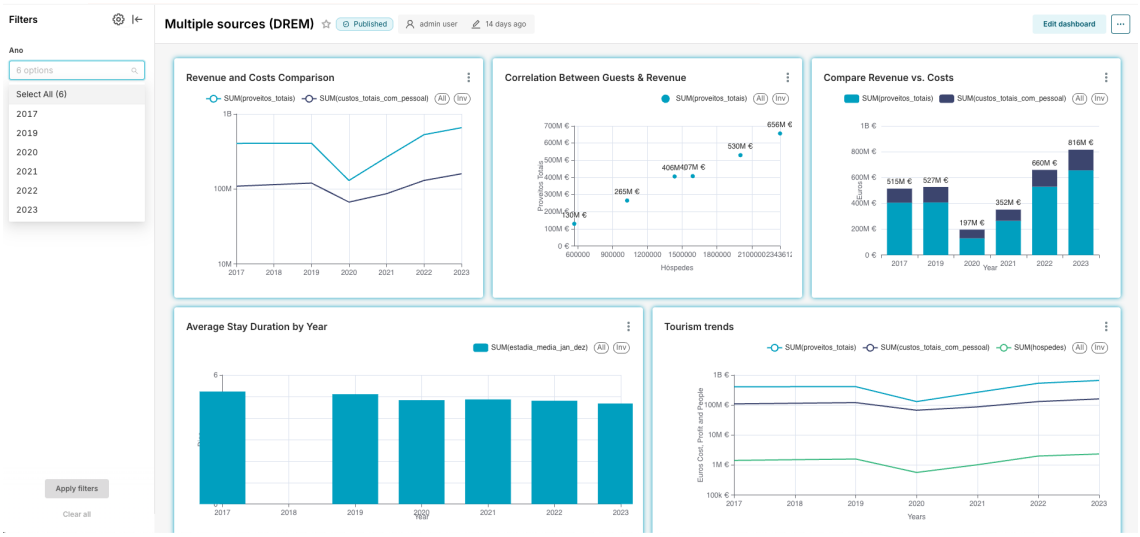

Fig. 14: The final integrated dashboard, demonstrating a solution to the cross-source filtering problem.

Figure 15 shows a chart from this dashboard that highlights the value of the solution. For this chart, a multi-line chart was chosen because it is the canonical visualisation for showing trends over a continuous variable like time [39]. It effectively combines metrics from the three different sources, allowing for clear trend analysis. The dip around 2020 clearly illustrates the cross-metric impact of the COVID-19 pandemic, an insight that was not possible when the data integration problem remained unsolved.

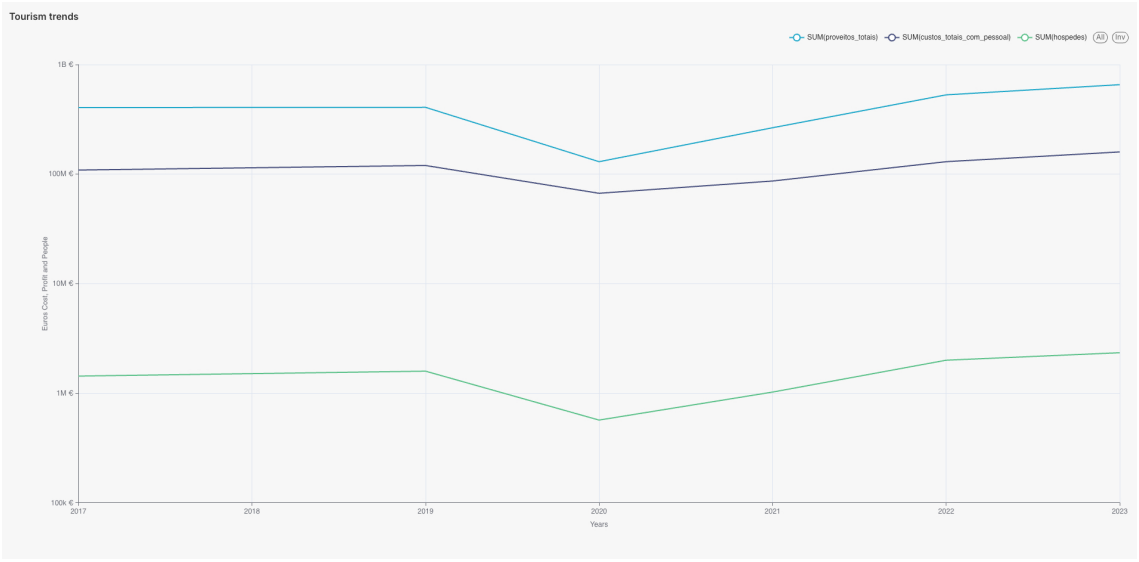

Fig. 15: A chart showcasing the successful integration of related tourism datasets (ID 4, 5, 6).

In stark contrast, the attempt to solve the integration problem for the environmental data failed from an analytical perspective. Although the dbt model technically succeeded in creating a consolidated table, this could not overcome the fundamental logical disconnect in the source data. The box plot (Figure 16) was selected with the intention of comparing the statistical distributions (median, quartiles, outliers) of the two environmental metrics. While a box plot is an excellent tool for this purpose, its application here was flawed. As Spence argues, a visualisation must map data to graphical properties in a way that preserves the data's integrity, by plotting two metrics with different units and scales on a single Y-axis, this principle was violated, rendering any direct comparison between the two plots meaningless [39]. The visualisation plots two metrics with different units and scales (wind speed in km/h and sea surface temperature in ^{o}C) against a single numerical Y-axis, rendering any direct comparison between them meaningless. This chart serves as a clear artefact of an unsolved data fusion problem.

This case study concludes that while the new architecture provides powerful tools for solving data integration problems, success is ultimately contingent on the logical compatibility of the data itself. However, the method for solving this problem highlights a key architectural advancement. While the initial creation of the dbt model requires domain knowledge and SQL expertise, the ongoing maintenance and data refresh is fully automated, reliable, and performant thanks to Airflow. This shifts the burden from a repeated, slow, and manual query process to a one-time, version-

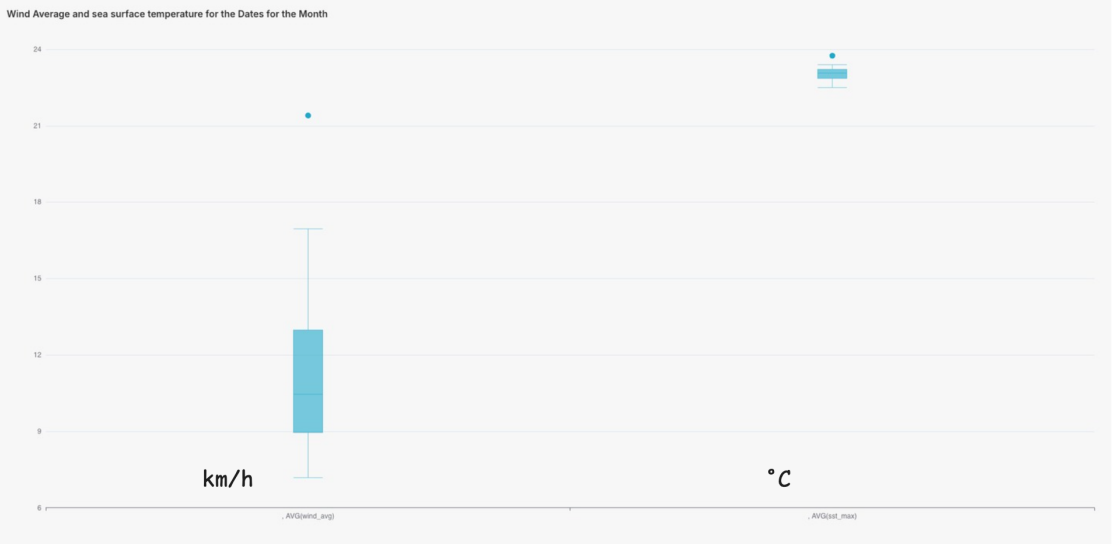

Fig. 16: A failed attempt to integrate unrelated environmental datasets (ID 1, 2, 3).

controlled development task. This dbt and Airflow approach provides a scalable and maintainable solution to the data integration challenge.

6.3 Case Study 3: Solving for Secure Data Governance

Effective data governance is important for any multi-user analytics platform, presenting the challenge of ensuring data is not only secure but also accurate, reliable, and auditable. This case study addressed this by validating the new architecture's governance framework across three core pillars: Access Control, Data Quality, and Auditability. The evaluation aimed to confirm that the platform provides a solution for managing data from its creation in the pipeline to its final consumption in a dashboard.

The first pillar of governance is ensuring that different users see only the data they are authorised to access. This case study addressed the challenge of implementing a secure and flexible permission model. The success criterion was binary: a data access rule must restrict a user's view to only the permitted data subset.

The evaluation was structured around a specific, practical scenario. First, a standard user hierarchy ('Admin', 'Alpha', 'Gamma') was created to establish broad access tiers. The core of the test was to solve a more specific security problem: restricting a 'Gamma' user to view data for only one specific competition ("MIUT 16") within a larger sporting event dataset.

The solution involved implementing a Row-Level Security (RLS) policy. This was defined with a simple SQL clause that filtered the dataset based on the competition name and was applied to the 'Gamma' role, while explicitly excluding the 'Admin' role. The test produced a clear and unambiguous outcome. For the visualisations in this test, a simple bar chart was chosen. According to Spence, bar charts excel at allowing for quick and accurate comparison of magnitudes between different discrete categories [39]. Here, the goal was to clearly show the quantity of athletes in each competition, making the bar chart the most effective choice.

The view for an 'Admin' user, shown in Figure 17, provides a complete, unfiltered overview of the event, with data for all five race categories visible.

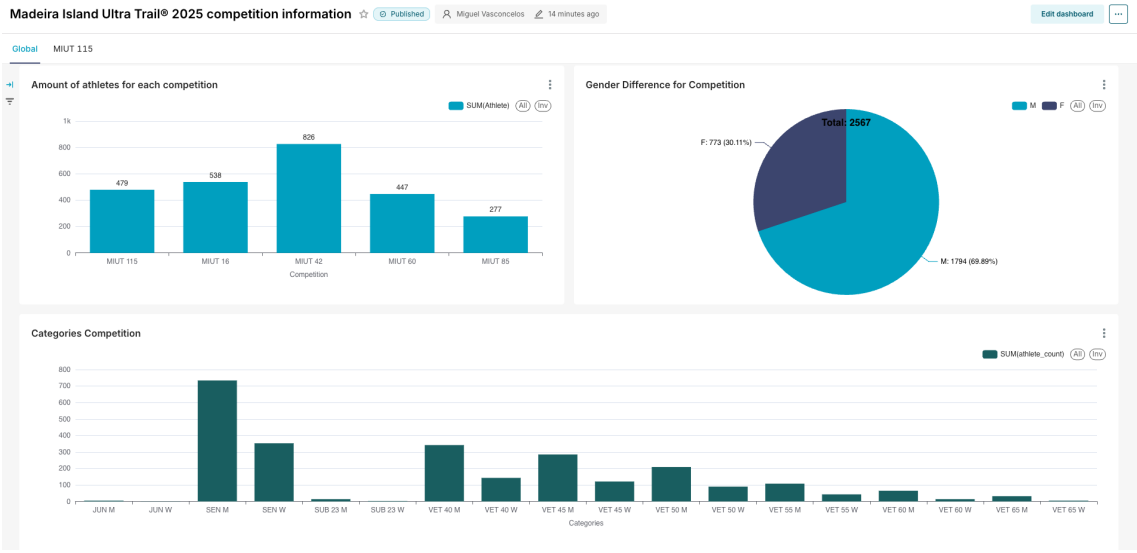

Fig. 17: Dashboard view for an Admin, showing all competition data (unfiltered).

In stark contrast, Figure 18 demonstrates that the problem was solved for the restricted user. Upon logging into the same dashboard, the RLS policy automatically filters the data, rendering only a single bar for the "MIUT 16" competition. The "Amount of athletes for each competition" chart now renders only a single bar representing the 538 athletes in the "MIUT 16" competition. The same happened for the other charts on the Figure. These results visually confirm the successful implementation of the RLS policy, as the user was effectively prevented from seeing or accessing data related to any other race.

The second pillar of governance is ensuring the data is accurate and trustworthy before it reaches a user. This was validated by testing the automated data quality checks built directly into

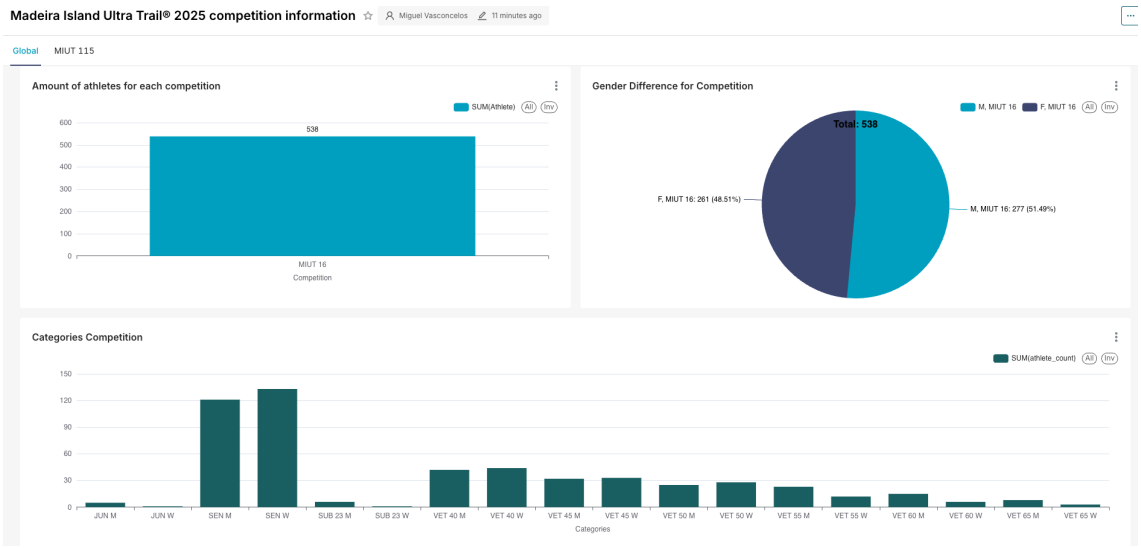

Fig. 18: Dashboard view for the restricted user, showing only "MIUT 16" data (filtered).

the data transformation pipeline. Within the dbt project, every data source has a corresponding schema.yml file where data quality tests are defined as part of the data model's contract. For example, the configuration for source_7 (Events in Madeira) includes declarative tests to enforce key integrity constraints. These tests ensure the primary key (id) is never null and is always unique, and that the data_source_id is always the expected value of 7, preventing data from other sources from being incorrectly loaded.

```
- name: source_7
    description: "Events in Madeira data from data source 7"
    columns:
      - name: id
        description: "Primary key from data lake"
        tests:
          - not_null
          - unique
      - name: data_source_id
        description: "Reference to data sources table"
        tests:
          - not_null
          - accepted_values:
              values: [7]
      - name: domain
```

```
    description: "Domain from data sources table"
    tests:
      - not_null
```

These defined tests function as an automated governance gate. Each time the Airflow DAG runs to create or refresh a data source table, it executes the dbt test command as a mandatory final step. If any of the data fails to meet these defined quality criteria (e.g., a duplicate ID is found), the test fails, causing the entire pipeline to fail. This important step prevents low-quality or corrupt data from ever reaching the analytics layer, thus ensuring the integrity of the data presented in all dashboards.

For instance, the images below illustrate a failure scenario during the daily refresh of the dbt_source_18_daily DAG.

The Figure 19 shows the Airflow Event Log, which provides a summary of the DAG run. It indicates that the task, test_dbt_source_18, entered a failed state at 2025-09-10, 00:40:55 UTC. This visual confirmation immediately signals that a data quality issue was detected during the validation step, preventing the successful completion of the run.

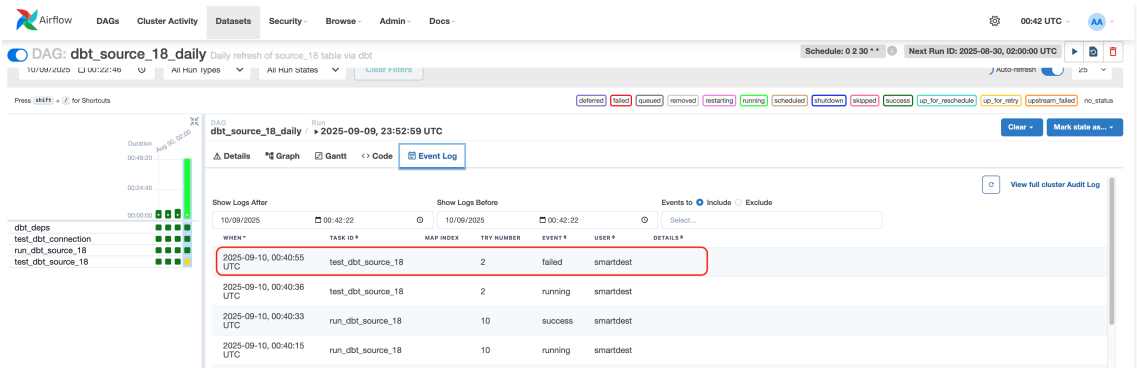

Fig. 19: Airflow Event Log view for the `dbt_source_18_daily` DAG.

Drilling down into the detailed task logs, as shown in the Figure 20, reveals the cause of the failure. The log states `Failure state in test accepted_values_source_18__data_source_id__99`. This message confirms that the data quality test, configured to ensure that the data_source_id column only contains accepted values, has failed. The log also indicates the message `Got 1 result, configured to fail if != 0`, meaning at least one record contained an invalid value.

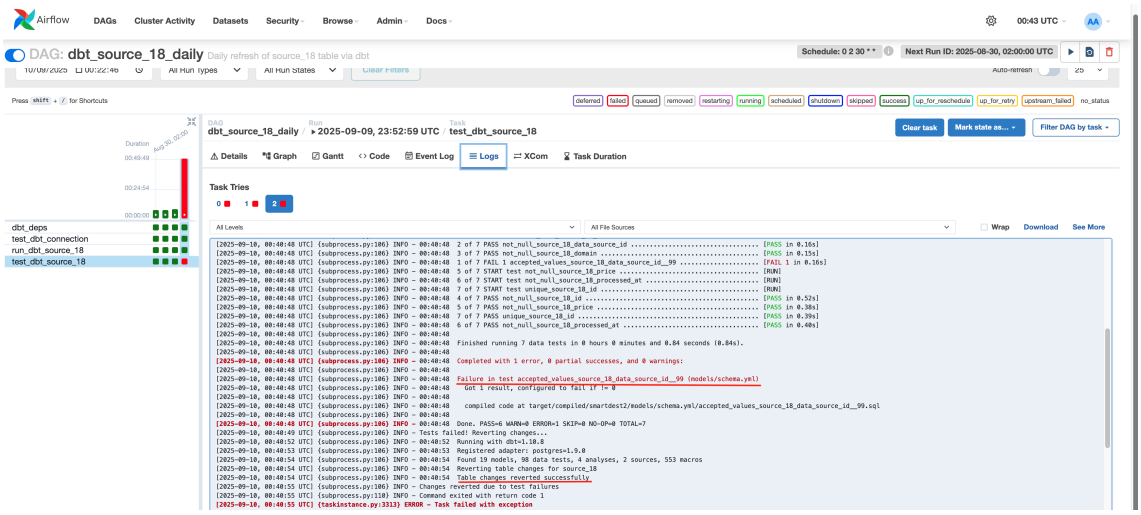

Fig. 20: Airflow task log showing the specific dbt test failure message.

Because this test failed, the dbt command exited with an error, which automatically failed the Airflow pipeline. This action effectively reverts the process, and the dataset containing the poor-quality data is not loaded or published. If the test had passed successfully, the pipeline would have completed, and that timestamped version of the data would have been made removed.

The third pillar of governance is auditability—the ability to track user activity on the platform for security and compliance purposes. This was validated by examining the platform's built-in action logs, which provide a detailed and immutable record of user interactions.

As shown in Figure 21, the system captures a detailed, timestamped record of all significant user events. The logs for the user "Miguel2 Vasconcelos" show a clear history of activities, including API calls to fetch dashboard lists (DashboardRestApi.get_list), get chart information (ChartRestApi.info), and run queries (ExploreFormDataRestApi.post). This detailed logging provides a complete audit trail, which is essential for security monitoring, understanding how users are interacting with the data, and troubleshooting issues.

The evaluation concluded that the new architecture's multi-faceted governance model is highly effective. It not only solves the problem of secure data access through RBAC and RLS but also enforces data quality through automated dbt testing and ensures full auditability via detailed action logs. However, implementing these solutions is not trivial. Managing a large number of access rules can become complex, and their creation requires a solid understanding of SQL for RLS rules and YAML for dbt tests. Despite this, the framework provides the necessary tools to build a secure, reliable, and transparent analytics environment.

	Miguel2 Vasconcelos	log	2025-08-20 16:08:20.056367
	Miguel2 Vasconcelos	DashboardRestApi.thumbnail	2025-08-20 16:08:19.451200
	Miguel2 Vasconcelos	DashboardRestApi.favorite_status	2025-08-20 16:08:19.420606
	Miguel2 Vasconcelos	DashboardRestApi.info	2025-08-20 16:08:19.408135
	Miguel2 Vasconcelos	ChartRestApi.info	2025-08-20 16:08:19.399594
	Miguel2 Vasconcelos	DashboardRestApi.get_list	2025-08-20 16:08:19.304858
	Miguel2 Vasconcelos	ChartRestApi.get_list	2025-08-20 16:08:19.246536
	Miguel2 Vasconcelos	DashboardRestApi.get_list	2025-08-20 16:08:19.178719
	Miguel2 Vasconcelos	ChartRestApi.get_list	2025-08-20 16:08:19.162874
	Miguel2 Vasconcelos	LogRestApi.recent_activity	2025-08-20 16:08:19.072776
	Miguel2 Vasconcelos	ExploreFormDataRestApi.post	2025-08-20 16:08:18.033940

Fig. 21: Superset's action log, showing a detailed audit trail for user Miguel2 Vasconcelos.

6.4 Case Study 4: Embedding Dashboards

To maximize the value of data insights, they must be delivered within the user's workflow, a challenge the previous platform did not address. This case study evaluated solutions for this integration problem by embedding analytics into an external web application while ensuring a high degree of user interactivity and security.

Two approaches were tested to solve the embedding challenge. The first was a simple static embed using an <iframe>. The second was a more sophisticated dynamic embed using an SDK, which required solving additional security problems by configuring guest token authentication and Cross-Origin Resource Sharing (CORS) policies.

The evaluation revealed that the simple <iframe> solution failed to solve the interactivity problem, as dashboard filters and controls were not functional. In contrast, the dashboard embedded using the SDK solved this issue, providing an experience nearly indistinguishable from the native platform where users could manipulate filters and see charts update in real-time.

From a security perspective, the guest token system provided a solution for granting temporary, scoped access. However, a significant architectural limitation was identified: the system supports only a single, globally defined role for all guest users. This means it cannot solve the embedding problem for complex multi-tenant scenarios where different external users require different levels of data access. Overcoming this would require complex workarounds, such as deploying and maintaining multiple platform instances.

This case study confirms that the new architecture provides a powerful framework for solving the embedded analytics problem, but the solution is technically complex and carries important

architectural trade-offs. The SDK method is the superior solution for user engagement, but the single guest role limitation is a major constraint for applications requiring varied permission levels and must be carefully considered.

6.5 Case Studies Findings

The four case studies demonstrate that the new architecture provides a flexible solution to the key problems of the legacy SmartDest platform. The findings confirm that the new system effectively solves issues related to visualisation flexibility, data integration performance, and provides a multi-faceted governance framework. However, this power comes with increased technical complexity and specific architectural limitations that must be carefully considered.

The first case study demonstrated that the new architecture solves the problems of static visualisations and poor performance. By enabling on-the-fly data modelling and the creation of interactive, multi-dimensional charts, the platform enables users to move beyond simple reporting. The ability to dynamically filter, group, and drill down into data provides a more powerful solution for data exploration. Furthermore, a direct performance benchmark confirmed that the new architecture is substantially more performant, handling over 300% more requests per second with an average latency 200 times lower than the legacy system. This combination of flexibility and speed directly addresses the core need for more intuitive and effective analytical tools for tourism professionals.

However, this newfound flexibility introduces a higher requirement for user skill and analytical maturity. Unlike the legacy system with its simple, predefined toggles, the new platform provides a full authoring environment. This means that to create effective visualisations, users must now possess a deeper understanding of the data and the principles of chart design. This represents a shift in responsibility from the platform to the user, requiring a higher level of training and analytical capability to unlock its full potential.

Regarding data integration, the second case study highlighted a major architectural advancement in solving performance and maintenance issues. The use of dbt to create pre-aggregated, materialised tables, which are then refreshed daily by Airflow, provides a scalable solution. This ELT approach eliminates the need for slow, on-the-fly queries, resulting in fast dashboard load times and a maintainable, version-controlled transformation pipeline.

Despite the technical success, this case study also revealed that the solution's effectiveness is entirely dependent on the logical compatibility of the source data. As demonstrated by the failed attempt to integrate heterogeneous environmental datasets, the platform cannot automatically resolve fundamental disconnects between sources. The process of creating a unified data model remains a manual, expertise-driven task that requires a deep understanding of both SQL and the business domain. The tools enable the solution, but they do not automate the analytical work required.

The third case study validated a data governance framework that extends beyond basic security. The architecture solves the challenge of data governance in a general manner, combining three pillars: fine-grained Access Control through RBAC and RLS, automated Data Quality checks via dbt tests integrated into the pipeline, and full Auditability with detailed user action logs. This multi-faceted approach ensures that data is not only secure but also trustworthy and that all user activity is traceable.

On the negative side, a significant governance gap was identified: the lack of a simple, built-in method for implementing Column-Level Security (CLS). This feature, which is essential for hiding sensitive columns like financial data or personal information from certain user roles, cannot be configured directly in the user interface. The required workaround involves creating and managing separate data models or views in the transformation layer (dbt) for each permission set. This adds significant complexity and maintenance overhead to the data engineering workflow.

For embedded analytics, the fourth case study proved that the new architecture can solve the challenge of delivering a fully interactive experience within an external application. By using the provided SDK and secure guest tokens, it is possible to create a smooth user experience where embedded dashboards behave almost identically to the native platform, with fully functional real-time filtering and interactivity.

This powerful embedding capability is constrained by a significant architectural limitation: the platform's reliance on a single, global role for all guest users. This makes it impossible to solve the embedding problem for common multi-tenant scenarios where different external customers or partners require different levels of data access. Fulfilling this requirement would necessitate complex and costly workarounds, such as deploying and maintaining multiple, separately configured platform instances.

The findings from the four case studies show a flexible platform that addresses the limitations of its predecessor, though it is not a simple drop-in replacement. The architecture excels in providing flexible visualisations, performant data integration, and a governance framework. However, these advanced capabilities come with the trade-off of increased technical complexity and specific architectural limitations. Therefore, the new architecture stands as an excellent choice for organisations with strong data engineering and development capabilities seeking a customisable and scalable BI solution, where its true value is realised not as an off-the-shelf tool, but as a platform for creating bespoke data experiences that solve specific business problems.

The four case studies provide an assessment of the new architecture's capabilities. The findings demonstrate that the platform addresses the core limitations of its predecessor in visualisation, data integration, and governance, while also introducing powerful new features such as embedded analytics. However, a recurring theme emerges: this increased power and flexibility comes with increased technical complexity and specific architectural trade-offs, such as the manual effort required for logical data modelling and limitations in the embedding feature for multi-tenant use cases. The new architecture thus stands as a powerful and flexible solution, while also requiring a realistic appraisal of its inherent complexities and constraints for any future implementation.

7 Conclusion

This project redesigned the SmartDest visualisation platform to resolve its limitations regarding chart diversity, dashboard integration, and rendering performance. By implementing a modern open-source data stack—comprising dbt Core, Apache Airflow, and Apache Superset—the platform transitioned from a static, tightly-coupled reporting tool to an automated, scalable analytics environment.

The four case studies validated that the new architecture meets the primary project objectives. The implementation of pre-aggregated, materialised tables via dbt reduced query latency (handling 223% more requests per second with 230 times lower latency). Furthermore, the integration of Apache Superset introduced dynamic cross-filtering, full dashboard authoring, and a governance framework featuring Row-Level Security (RLS).

However, these functional improvements introduce a trade-off in technical complexity. The new architecture requires a higher degree of data engineering expertise for maintenance, specifically in writing SQL for dbt models and managing Airflow DAGs. Additionally, while dashboard embedding was achieved, the current system's reliance on a single guest role limits its scalability for multi-tenant scenarios.

7.1 Future Work

An area for future development involves creating a middle system for OLAP, such as integrating Apache Druid. While the current PostgreSQL-based serving layer is performant for tested data volumes, it is anticipated to become a bottleneck as data grows, necessitating migration to a more specialised, high-performance OLAP engine for better query performance on large datasets.

An improved semantic layer for self-service analytics and Column-Level Security is also proposed. This layer could use dbt's semantic layer or integrate a dedicated metrics tool like Cube.js, allowing for the central definition of business metrics, dimensions, and access permissions. This would enable less technical users and provide a more elegant solution to the CLS gap by defining permissions on semantic objects. It would also be necessary to improve the use of dbt when creating the updated data for the daily formation.

The work that was not done concerning Keycloak, as conceptualized for the platform, centered on establishing a unified and global authentication method such as a Single Sign-On (SSO) solution

using a provider like Keycloak, which was envisioned as a foundational, cross-cutting security service for the entire data stack, encompassing the transformation, API, and visualization layers. The core idea was to externalize user identity and access management, allowing a user to authenticate only once in this central service and subsequently gain access to various tools within the platform, such as Airflow for orchestration or Superset for visualization, based on a single, centrally managed set of credentials and permissions. This approach would improve security and simplify user administration by eliminating the need to manage separate user accounts for each application. The development of this authentication layer with Keycloak remains for future work.

An iteration on the platform's usability and visualisation layer is a step for the next phase. The current implementation validates the architecture's ability to serve multi-source tourism data reliably, but the usability of the dashboard creation workflow and the design of pre-built visualisations for tourism professionals have not been evaluated. A usability study with domain stakeholders, followed by iterative UI refinement would complete the platform's development cycle.

Finally, proactive data quality monitoring with anomaly detection is proposed to evolve the current reactive data governance framework. By integrating machine learning libraries into the data pipeline to monitor time-series metrics, the system could learn normal data patterns, flag unexpected deviations, and trigger alerts via Airflow.

References

[1] C. Gonçalves, M. Angélico Gonçalves, and M. Campante, "Developing integrated performance dashboards visualisations using power bi as a platform," *Information*, vol. 14, p. 614, 11 2023.

[2] X. Wu and J. Huang, "Construction of a rural tourism information service management system for multi-source heterogeneous data processing," *PeerJ Computer Science*, vol. 9, p. e1334, Jun. 2023, publisher: PeerJ Inc. [Online]. Available: https://peerj.com/articles/cs-1334

[3] O. A. Nikolaychuk, D. E. Kosogorov, Y. V. Pestova, and A. I. Pavlov, "Tourism industry monitoring service based on analysis of informational web resources," *Preprints*, August 2023. [Online]. Available: https://doi.org/10.20944/preprints202308.0517.v1

[4] T. Kalvet, M. Olesk, M. Tiits, and J. Raun, "Innovative Tools for Tourism and Cultural Tourism Impact Assessment," *Sustainability*, vol. 12, no. 18, p. 7470, Jan. 2020, number: 18 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2071-1050/12/18/7470

[5] W. A. Yousef, A. A. Abouelkahire, O. S. Marzouk, S. K. Mohamed, and M. N. Alaggan, "DVP: data visualization platform," *CoRR*, vol. abs/1906.11738, 2019. [Online]. Available: http://arxiv.org/abs/1906.11738

[6] L. Wang, "Enhancing tourism management through big data: Design and implementation of an integrated information system," *Heliyon*, vol. 10, no. 20, Oct. 2024, publisher: Elsevier. [Online]. Available: https://doi.org/10.1016/j.heliyon.2024.e38256

[7] X. Wang and B. Yi, "The application of data cubes in business data visualization," *Computing in Science & Engineering*, vol. 14, no. 6, pp. 44–50, 2012.

[8] C. Balzer, R. Oktavian, M. Zandi, D. Fairen-Jimenez, and P. Z. Moghadam, "Wiz: A Web-Based Tool for Interactive Visualization of Big Data," *Patterns*, vol. 1, no. 8, Nov. 2020, publisher: Elsevier. [Online]. Available: https://doi.org/10.1016/j.patter.2020.100107

[9] T. Hossain, S. S. Movva, and R. Ritika, "Chronological outlooks of globe illustrated with web-based visualization," *arXiv preprint arXiv:2404.16063*, 2024.

[10] R. P. Soesanto, I. Zulkarnain, A. A. Pramudita, and L. Andrawina, "Optimization of Banyuwangi Festival Management through ERP-Tourism Using Dashboard," *Metris: Jurnal Sains dan Teknologi*, vol. 25, no. 02, pp. 61–66, Oct. 2024, number: 02. [Online]. Available: https://ejournal.atmajaya.ac.id/index.php/metris/article/view/5424

[11] R. Soeprawa and R. Supono, "Developing Tourism Performance Dashboard for National Tourism Organisation: The Case of Indonesia," *Jurnal Kepariwisataan: Destinasi, Hospitalitas dan Perjalanan*, vol. 8, no. 2, pp. 197–221, Dec. 2024, number: 2. [Online]. Available: https://journal.poltekpar-nhi.ac.id/index.php/jk/article/view/1611

[12] S. Laori, V. Sarapang, N. Nurjannah, and masri ridwan, "Optimizing sustainable tourism capacity in tanjung bira: A real-time environmental management dashboard approach," *Social, Humanities, and Educational Studies (SHES): Conference Series*, vol. 7, no. 4, pp. 40–56, 2024. [Online]. Available: https://jurnal.uns.ac.id/SHES/article/view/95670

[13] G. Balletto, A. Milesi, M. Ladu, and G. Borruso, "Smart Community & Smart Dashboard to Support Slow Tourism: The Case Study of Santa Barbara Walk (Sardinia, Italy)," *Preprints*, Nov. 2019, publisher: Preprints. [Online]. Available: https://doi.org/10.20944/preprints201911.0029.v1

[14] H. S. Albusaidi, P. K. Udupi, and V. Dattana, "Integrated data analytic tourism dashboard (idatd)," in *2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2016, pp. 497–500.

[15] D. Ordóñez-Martínez, J. M. Seguí-Pons, and M. Ruiz-Pérez, "Toward establishing a tourism data space: Innovative geo-dashboard development for tourism research and management," *Smart Cities*, vol. 7, no. 1, pp. 633–661, 2024. [Online]. Available: https://www.mdpi.com/2624-6511/7/1/26

[16] Z. Qin and Y. Pan, "Design of A Smart Tourism Management System through Multisource Data Visualization-Based Knowledge Discovery," *Electronics*, vol. 12, no. 3, p. 642, Jan. 2023, number: 3 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2079-9292/12/3/642

[17] P. Krataithong, C. Anutariya, and M. Buranarach, "A Data Management Platform for

Taxi Trajectory-based Tourist Behavior Analysis," in *Proceedings of the 13th International Conference on Management of Digital EcoSystems*, ser. MEDES '21. New York, NY, USA: Association for Computing Machinery, Nov. 2021, pp. 15–21. [Online]. Available: https://doi.org/10.1145/3444757.3485104

[18] C. Prandi, V. Nisi, M. Ribeiro, and N. Nunes, "Sensing and making sense of tourism flows and urban data to foster sustainability awareness: a real-world experience," *Journal of Big Data*, vol. 8, no. 1, p. 51, Mar. 2021. [Online]. Available: https://doi.org/10.1186/s40537-021-00442-w

[19] "BITOUR: A Business Intelligence Platform for Tourism Analysis." [Online]. Available: https://www.mdpi.com/2220-9964/9/11/671

[20] A. Bustamante, L. Sebastia, and E. Onaindia, "Can Tourist Attractions Boost Other Activities Around? A Data Analysis through Social Networks," *Sensors*, vol. 19, no. 11, p. 2612, Jan. 2019, number: 11 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/19/11/2612

[21] A. Berko, V. Vysotska, O. Naum, N. Borovets, S. Chyrun, and V. Panasyuk, "Big Data Analysis for Startup of Supporting Ukraine Internet Tourism," *2023 IEEE 5th International Conference on Advanced Information and Communication Technologies (AICT)*, pp. 164–169, Nov. 2023, conference Name: 2023 IEEE 5th International Conference on Advanced Information and Communication Technologies (AICT) ISBN: 9798350372571 Place: Lviv, Ukraine Publisher: IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/10452425/

[22] F. Gaspari, F. Barbieri, R. Fascia, F. Ioli, and L. Pinto, "An open-source web platform for 3d documentation and storytelling of hidden cultural heritage," *Heritage*, vol. 7, no. 2, pp. 517–536, 2024. [Online]. Available: https://www.mdpi.com/2571-9408/7/2/25

[23] M. Masson, C. Cayèré, M.-N. Bessagnet, C. Sallaberry, P. Roose, and C. Faucher, "An ETL-like platform for the processing of mobility data," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '22. New York, NY, USA: Association for Computing Machinery, May 2022, pp. 547–555. [Online]. Available: https://doi.org/10.1145/3477314.3507057

[24] N. Maiellaro and A. Varasano, "One-page multimedia interactive map," *ISPRS International Journal of Geo-Information*, vol. 6, no. 2, 2017. [Online]. Available: https://www.mdpi.com/2220-9964/6/2/34

[25] N. Das, H. Rahman, J. Hassan, M. Bhuiyan, and F. Mahmud, "The strategic impact of business intelligence tools: A review of decision-making and ambidexterity," *Membrane Technology*, vol. 2025, pp. 542–553, 01 2025.

[26] M. Durra and G. Al-Naymat, "Self-service business intelligence: Meeting user needs in a data-rich world," in *2023 24th International Arab Conference on Information Technology (ACIT)*, 2023, pp. 1–6.

[27] K. K. Tirupati, A. Joshi, D. S. P. Singh, A. Chhapola, S. Jain, and D. A. Gupta, "Leveraging Power BI for Enhanced Data Visualization and Business Intelligence," *Universal Research Reports*, vol. 10, no. 2, pp. 676–711, Sep. 2024. [Online]. Available: https://urr.shodhsagar.com/index.php/j/article/view/1375

[28] I. M. P. Utama, I. G. A. S. Putra, I. B. K. Sandhisutra, G. S. Mahendra, and N. M. M. R. Desmayani, "Visualization of Tourism Site Visit Levels in Karo Regency Using Tableau," *RIGGS: Journal of Artificial Intelligence and Digital Business*, vol. 2, no. 2, pp. 50–55, Jan. 2024, number: 2. [Online]. Available: https://journal.ilmudata.co.id/index.php/RIGGS/article/view/221

[29] N. Azis, A. J. Wahidin, P. A. Cakranegara, A. Muditomo, and E. Efendi, "Visualization Of Tourist Visit Time Series Data Using Google Data Studio," *Jurnal Mantik*, vol. 6, no. 2, pp. 2153–2159, Aug. 2022, number: 2. [Online]. Available: https://iocscience.org/ejournal/index.php/mantik/article/view/2731

[30] G. H. D. d. C. Soares, "Leveraging apache superset: a business intelligence solution," Master's thesis, Universidade do Minho, Sep. 2023, accepted: 2025-01-28T11:08:44Z Journal Abbreviation: Potenciar o apache superset: uma solução de business intelligence. [Online]. Available: https://repositorium.sdum.uminho.pt/handle/1822/94650

[31] B. Sijtsma, P. Qvarfordt, and F. Chen, "Tweetviz: Visualizing tweets for business intelligence," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development*

in Information Retrieval, ser. SIGIR '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1153–1156. [Online]. Available: https://doi.org/10.1145/2911451.2911470

[32] A. Mehdi, M. K. Bali, S. I. Abbas, and M. Singh, "\"unleashing the potential of grafana: A comprehensive study on real-time monitoring and visualization\"," in *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2023, pp. 1–8.

[33] B. Herre and V. Samborska, "Tourism," *Our World in Data*, 2023, https://ourworldindata.org/tourism.

[34] E. H. Fegraus, I. Zaslavsky, T. Whitenack, J. Dempewolf, J. A. Ahumada, K. Lin, and S. J. Andelman, "Interdisciplinary decision support dashboard: A new framework for a tanzanian agricultural and ecosystem service monitoring system pilot," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 6, pp. 1700–1708, 2012.

[35] A. Schoedon, M. Trapp, H. Hollburg, D. Gerber, and J. D¨ollner, "Web-based visualization of transportation networks for mobility analytics," in *Proceedings of the 12th International Symposium on Visual Information Communication and Interaction*, ser. VINCI '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3356422.3356425

[36] G. O. Osho, J. O. Omisola, and J. O. Shiyanbola, "An Integrated AI-Power BI Model for Real-Time Supply Chain Visibility and Forecasting: A Data-Intelligence Approach to Operational Excellence," *International Journal of Advanced Multidisciplinary Research and Studies*, vol. 4, no. 6, pp. 1463–1480, Apr. 2025. [Online]. Available: https://www.multiresearchjournal.com/arclist/list-2024.4.6/id-4047

[37] E. Marques and F. Gonçalves, "Smartdest project: Madeira island implementation and experience," in *International Tourism Congress ITC2022*, Łódź, Poland, November 16–19 2022.

[38] P. Majdak, A. M. M. de Almeida, and A. Nowakowska, "Smart island and sustainable tourist development with the example of madeira. part 1: Theoretical contexts and development conditions," *European Research Studies Journal*, vol. XXIV, no. 4B, pp. 421–432, 2021.

[39] R. Spence, *Information Visualization: An Introduction*, 3rd ed. Springer Publishing Company, Incorporated, 2014.

[40] F. K. Ramadhan, G. Garno, and A. Solehudin, "Comparative study of web server performance testing with and without docker based on virtual machines," *Journal of Applied Informatics and Computing*, vol. 8, no. 1, p. 155–166, Jul. 2024. [Online]. Available: https://jurnal.polibatam.ac.id/index.php/JAIC/article/view/3884

[41] F. H. L. Buzato and A. Goldman, "Optimizing microservices performance and resource utilization through containerized grouping: An experimental study," in *2023 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW)*, 2023, pp. 115–122.

Appendices

A Apache Superset Configuration Details

This appendix contains the configuration files and instructions for deploying the Apache Superset 5.0.0 instance. This version requires an additional websocket service for asynchronous chart loading and uses helper scripts for a more reliable startup process.

A.1 docker-compose.yaml

This updated ‘docker-compose.yaml‘ file includes the new ‘superset-websocket‘ service and uses modern startup commands.

```
version: "3.7"

x-superset-common:
  &superset-common
  image: apache/superset:${SUPERSET_VERSION:-5.0.0}
  volumes:
    - ./docker/superset_config.py:/app/superset_config.py
    - superset-home:/app/superset_home
    - ./docker/docker-bootstrap.sh:/app/docker/docker-bootstrap.sh
    - ./docker/docker-init.sh:/app/docker/docker-init.sh
  user: "${SUPERSET_UID:-1001}:0"
  restart: unless-stopped
  depends_on:
    &superset-depends-on
    db:
      condition: service_healthy
    redis:
      condition: service_healthy

services:
  db:
    image: postgres:15
    container_name: superset_db
```

```
    environment:
      POSTGRES_USER: ${POSTGRES_USER:-superset}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD:-superset}
      POSTGRES_DB: ${POSTGRES_DB:-superset}
    volumes:
      - postgres-db-volume:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD", "pg_isready", "-U", "${POSTGRES_USER:-superset}"]
      interval: 10s
      retries: 5
      start_period: 5s
    restart: unless-stopped

  redis:
    image: redis:7.2
    container_name: superset_redis
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 10s
      timeout: 5s
      retries: 5

  superset:
    <<: *superset-common
    container_name: superset_app
    command: ["/app/docker/docker-bootstrap.sh", "app-gunicorn"]
    ports:
      - "8088:8088"
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8088/health"]
      interval: 30s
      timeout: 10s
      retries: 5
```

```
    start_period: 30s

superset-worker:
  <<: *superset-common
  container_name: superset_worker
  command: ["/app/docker/docker-bootstrap.sh", "worker"]
  healthcheck:
    test: ["CMD-SHELL", "celery -A superset.tasks.celery_app:app inspect ping -d celery@$$HOSTNAME"
    interval: 30s
    timeout: 10s
    retries: 5

superset-beat:
  <<: *superset-common
  container_name: superset_beat
  command: ["/app/docker/docker-bootstrap.sh", "beat"]
  healthcheck:
    test: ["CMD-SHELL", "celery -A superset.tasks.celery_app:app inspect ping -d celery@$$HOSTNAME"
    interval: 30s
    timeout: 10s
    retries: 5

superset-init:
  <<: *superset-common
  container_name: superset_init
  command: ["/app/docker/docker-init.sh"]
  user: "root"
  depends_on:
    - db
    - redis
  healthcheck:
    test: ["CMD-SHELL", "superset --help"]
    interval: 10s
    timeout: 5s
    retries: 5
```

```
  superset-websocket:
    image: apache/superset-websocket:${SUPERSET_WEBSOCKET_VERSION:-1.1.0}
    container_name: superset_websocket
    ports:
      - "8080:8080"
    environment:
      - SUPERSET_WEBSOCKET_SECRET=${SUPERSET_WEBSOCKET_SECRET}
    depends_on:
      - redis
    user: "node"
    restart: unless-stopped
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:8080/health"]
      interval: 30s
      timeout: 10s
      retries: 5

volumes:
  postgres-db-volume:
  superset-home:
```

A.2 superset_config.py

This file should be placed in a ‘docker‘ subdirectory (‘docker/superset_config.py‘). It has been updated to include websocket configuration and modern feature flags.

```
import os
from datetime import timedelta
from celery.schedules import crontab
from redis import Redis

SECRET_KEY = os.environ.get("SUPERSET_SECRET_KEY", "a_very_secret_key_that_should_be_changed")
SQLALCHEMY_DATABASE_URI = (
    f"postgresql+psycopg2://{os.getenv('POSTGRES_USER')}:{os.getenv('POSTGRES_PASSWORD')}@"
    f"db:{os.getenv('POSTGRES_PORT', '5432')}/{os.getenv('POSTGRES_DB')}"
```

```
)
SQLALCHEMY_TRACK_MODIFICATIONS = True
REDIS_HOST = os.getenv("REDIS_HOST", "redis")
REDIS_PORT = os.getenv("REDIS_PORT", "6379")

CACHE_CONFIG = {
    "CACHE_TYPE": "RedisCache",
    "CACHE_DEFAULT_TIMEOUT": 300,
    "CACHE_KEY_PREFIX": "superset_cache_",
    "CACHE_REDIS_URL": f"redis://{REDIS_HOST}:{REDIS_PORT}/1",
}
DATA_CACHE_CONFIG = CACHE_CONFIG

class CeleryConfig:
    broker_url = f"redis://{REDIS_HOST}:{REDIS_PORT}/0"
    imports = ("superset.sql_lab",)
    result_backend = f"redis://{REDIS_HOST}:{REDIS_PORT}/0"
    worker_prefetch_multiplier = 1
    task_acks_late = True
CELERY_CONFIG = CeleryConfig

EMBEDDED_SUPERSET_JWT_SECRET = os.environ.get("EMBEDDED_SUPERSET_JWT_SECRET", "default_embedded_secre
GUEST_TOKEN_JWT_SECRET = os.environ.get("GUEST_TOKEN_JWT_SECRET", "default_guest_token_secret_change_
SESSION_SERVER_SIDE = True
SESSION_TYPE = "redis"
SESSION_REDIS = Redis(host=REDIS_HOST, port=int(REDIS_PORT), db=3)
ENABLE_CORS = True
CORS_OPTIONS = {
    "supports_credentials": True,
    "origins": ["http://localhost:3000"],
}
TALISMAN_ENABLED = True
TALISMAN_CONFIG = {
    "content_security_policy": {
```

```
        "frame-ancestors": ["'self'", "http://localhost:3000"]
    }
}
FEATURE_FLAGS = {
    "ALERT_REPORTS": True,
    "EMBEDDED_SUPERSET": True,
    "TAGGING_SYSTEM": True,
    "GLOBAL_ASYNC_QUERIES": True,
    "DASHBOARD_RBAC": True,
}
GLOBAL_ASYNC_QUERIES_JWT_SECRET = os.environ.get("GLOBAL_ASYNC_QUERIES_JWT_SECRET", "a_very_sec
```

A.3 .env file

Create this file in the same directory as ‘docker-compose.yaml‘ to manage environment variables.

```
SUPERSET_UID=1001
SUPERSET_VERSION=5.0.0
SUPERSET_WEBSOCKET_VERSION=1.1.0
SUPERSET_WEBSOCKET_SECRET="a_very_secret_key_that_should_be_changed"
POSTGRES_USER=superset
POSTGRES_PASSWORD=superset
POSTGRES_DB=superset
```

A.4 How to Use

1. Create a project directory.
2. Inside it, save the `docker-compose.yaml` and `.env` files.
3. Create a subdirectory named `docker`.
4. Inside the `docker` subdirectory, save the `superset_config.py` file.
5. **Important:** You must also download the `docker-bootstrap.sh` and `docker-init.sh` scripts from the official Apache Superset GitHub repository (from the `5.0.0` tag) and place them in the `docker` subdirectory. These scripts are required to properly initialize and start the services.

6. From your project directory, run the command: `docker compose up -d`.

7. On the first run, the `superset-init` service will set up the database and create the default admin user. The credentials are `admin` / `superset`.

B Apache Airflow and dbt Configuration Details

B.1 Introduction

This appendix provides a collection of all configuration files required for setting up and operating the Apache Airflow and dbt environment. The setup includes:

- Apache Airflow with a PostgreSQL backend
- dbt Core with a PostgreSQL adapter
- A custom Docker image with extended dependencies
- The complete project structure and configuration

B.2 Project Structure

```
apache_airflow/
├── docker-compose.yaml
├── Dockerfile
├── requirements.txt
├── dags/
├── logs/
├── config/
├── plugins/
├── dbt/
│   ├── profiles.yml
│   └── smartdest2/
│       ├── dbt_project.yml
│       ├── packages.yml
│       └── models/
```

```
└── tests/
```

B.3 Core Configuration Files

B.3.1 Docker Compose Configuration (docker-compose.yaml)

```yaml
x-airflow-common:
  &airflow-common
  image: ${AIRFLOW_IMAGE_NAME:-extending_airflow:latest_v9}
  environment:
    &airflow-common-env
    AIRFLOW__CORE__EXECUTOR: LocalExecutor
    AIRFLOW__DATABASE__SQL_ALCHEMY_CONN: postgresql+psycopg2://airflow:airflow@postgres/airflow
    AIRFLOW__CORE__FERNET_KEY: 'rgaQPnWgGWtCg_5Kvy8jpU7CsydUyhXbFVgDPuBW4ac='
    AIRFLOW__CORE__LOAD_EXAMPLES: 'false'
  volumes:
    - ${AIRFLOW_PROJ_DIR:-.}/dags:/opt/airflow/dags
    - ${AIRFLOW_PROJ_DIR:-.}/logs:/opt/airflow/logs
    - ${AIRFLOW_PROJ_DIR:-.}/dbt:/opt/airflow/dbt
  user: "${AIRFLOW_UID:-50000}:0"
  depends_on:
    &airflow-common-depends-on
    postgres:
      condition: service_healthy

services:
  postgres:
    image: postgres:13
    environment:
      POSTGRES_USER: airflow
      POSTGRES_PASSWORD: airflow
      POSTGRES_DB: airflow
    volumes:
      - postgres-db-volume:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD", "pg_isready", "-U", "airflow"]
```

```
  restart: always

airflow-webserver:
  <<: *airflow-common
  command: webserver
  ports:
    - "8080:8080"
  healthcheck:
    test: ["CMD", "curl", "--fail", "http://localhost:8080/health"]
  restart: always
  depends_on:
    <<: *airflow-common-depends-on
    airflow-init:
      condition: service_completed_successfully

airflow-scheduler:
  <<: *airflow-common
  command: scheduler
  healthcheck:
    test: ["CMD", "airflow", "jobs", "check", "--job-type", "SchedulerJob", "--hostname", "$${HOSTN
  restart: always
  depends_on:
    <<: *airflow-common-depends-on
    airflow-init:
      condition: service_completed_successfully

airflow-init:
  <<: *airflow-common
  entrypoint: /bin/bash
  command:
    - -c
    - |
      mkdir -p /sources/logs /sources/dags /sources/plugins
      chown -R "${AIRFLOW_UID}:0" /sources/{logs,dags,plugins}
      exec /entrypoint airflow version
```

```
    environment:
      <<: *airflow-common-env
      _AIRFLOW_DB_MIGRATE: 'true'
      _AIRFLOW_WWW_USER_CREATE: 'true'
      _AIRFLOW_WWW_USER_USERNAME: ${_AIRFLOW_WWW_USER_USERNAME:-airflow}
      _AIRFLOW_WWW_USER_PASSWORD: ${_AIRFLOW_WWW_USER_PASSWORD:-airflow}
    user: "0:0"
    volumes:
      - ${AIRFLOW_PROJ_DIR:-.}:/sources

volumes:
  postgres-db-volume:
```

B.3.2 Dockerfile

```
FROM apache/airflow:2.10.5

COPY requirements.txt /requirements.txt

RUN pip install --no-cache-dir -r /requirements.txt
```

B.3.3 Python Requirements (requirements.txt)

```
apache-airflow-providers-postgres==6.0.0

psycopg2-binary==2.9.10

dbt-core==1.10.8

dbt-postgres==1.9.0
```

B.4 dbt Configuration Files

B.4.1 Database Connection Profiles (dbt/profiles.yml)

```
smartdest2:
  target: dev
  outputs:
    dev:
      type: postgres
      host: 10.2.15.210
      user: postgres
      password: your_db_password
```

```
      port: 5432
      dbname: smartdest_production
      schema: public
      threads: 4
```

B.4.2 dbt Project Configuration (dbt/smartdest2/dbt_project.yml)

```
name: 'smartdest2'
version: '1.0.0'
config-version: 2
profile: 'smartdest2'
model-paths: ["models"]
test-paths: ["tests"]
clean-targets:
  - "target"
  - "dbt_packages"
models:
  smartdest2:
    +materialized: table
```

B.4.3 dbt Package Dependencies (dbt/smartdest2/packages.yml)

```
packages:
  - package: dbt-labs/dbt_utils
    version: 1.3.0
```

C Case Study 1 Test

```
=== API Performance Comparison using wrk ===
Date: Sat Aug 20 15:12:09 WEST 2025

System Specifications:
OS: macOS 15.6.1
CPU: Apple M3 Pro
Cores: 11 (5 performance and 6 efficiency)
Memory: 18 GB
```

```
Network: 340.194ms avg ping to 8.8.8.8

Test Configuration:
Duration: 30s
Connections: 12
Threads: 4
Timeout: 120s

Starting performance tests...

Testing Endpoint 1: GET sd-data.web.uma.pt
=== sd-data.web.uma.pt API ===
Running 30s test @ https://sd-data.web.uma.pt/api/source/1/data?start=2019-01-01T00:00:00.000Z&
  4 threads and 12 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    10.10s     1.44s   14.20s    83.33%
    Req/Sec     1.00      2.08     9.00     86.67%
  30 requests in 30.05s, 377.32MB read
Requests/sec:      1.00
Transfer/sec:     12.56MB

Key Metrics for sd-data.web.uma.pt API:
Requests/sec: 1.00
Average Latency: 10.10s
Max Latency: 14.20s
Total Requests: 30
Data Transferred: read

Test completed successfully in in30.05s,
Note: wrk waits for ALL responses to complete during the test duration

Waiting 15 seconds before local test...

Authenticating with Superset API...
Session token obtained successfully (71 characters)
```

```
Token type: Session Cookie

Testing Endpoint 2: POST to local server (10.2.15.212) - ALL DATA
=== Local Server API (10.2.15.212) ===
Running 30s test @ http://10.2.15.212:8088/api/v1/chart/data
  4 threads and 12 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    43.64ms    3.61ms  59.47ms   81.44%
    Req/Sec    56.76     22.90    90.00     58.82%
  97 requests in 30.03s, 65.65KB read
  Socket errors: connect 0, read 3291, write 0, timeout 0
  Non-2xx or 3xx responses: 97
Requests/sec:      3.23
Transfer/sec:      2.19KB

Key Metrics for Local Server API (10.2.15.212):
Requests/sec: 3.23
Average Latency: 43.64ms
Max Latency: 59.47ms
Total Requests: 97
Data Transferred: read
Errors:   Socket errors: connect 0, read 3291, write 0, timeout 0

Test completed successfully in in30.03s,
Note: wrk waits for ALL responses to complete during the test duration

=== PERFORMANCE COMPARISON SUMMARY ===

Requests per Second:
  sd-data.web.uma.pt: 1.00 RPS
  Local server: 3.23 RPS
  Local server is 223.00% faster

Average Latency:
  sd-data.web.uma.pt: 10.10
```

```
  Local server: 43.64

Notes:
- sd-data.web.uma.pt: GET request, external HTTPS endpoint (4-year date range)
- Local server (10.2.15.212): POST request to Superset with ALL DATA (no row limit), HTTP endpo
- Different request types and payload sizes significantly affect performance
- Network latency to external vs local server impacts results
- wrk waits for ALL HTTP responses to complete within the test duration
- External endpoint may timeout due to large date range (2019-2022)
- Superset endpoint uses 'full' format to get ALL data from the table


Note: The socket errors in the local test are expected when the authentication fails or the enc

Performance test completed!
```

D Case Study 2 dbt Models

D.1 dbt model for source 4,5,6

```
{{ config(
materialized='table'
) }}
SELECT
s4.year, s4.dormidas, s4.hospedes, s5.proveitos_totais,
s5.custos_totais_com_pessoal, s6.dormidas_jan_dez,
s6.hospedes_jan_dez, s6.estadia_media_jan_dez
FROM {{ ref('source_4') }} AS s4
JOIN {{ ref('source_5') }} AS s5
ON s4.year = s5.year AND s4.total = s5.total
JOIN {{ ref('source_6') }} AS s6
ON s4.year = s6.year AND s4.total = s6.total
WHERE s4.total IS TRUE
ORDER BY s4.year
```

D.2 dbt model for source 1,2,3

```
{{ config(
    materialized='table',
    indexes=[
        {'columns': ['date_forecast'], 'type': 'btree'},
        {'columns': ['location'], 'type': 'btree'}
    ]
) }}

SELECT
  T1.date_forecast,
  T1.location,
  AVG(T1.temperature_avg) AS temperature_avg,
  AVG(T1.rain_avg) AS rain_avg,
  AVG(T1.wind_avg) AS wind_avg,
  AVG(T1.cloud_avg) AS cloud_avg,
  AVG(T1.humidity_avg) AS humidity_avg,
  AVG(T3.sst_max) AS sst_max,
  AVG(T3.wave_high_max) AS wave_high_max,
  AVG(T3.wave_period_max) AS wave_period_max,
  STRING_AGG(DISTINCT T3.pred_wave_dir, ', ') AS pred_wave_dir,
  STRING_AGG(T2.awareness_type_name, ', ') AS warning_types,
  STRING_AGG(T2.awareness_level_id, ', ') AS warning_levels
FROM {{ ref('source_1') }} AS T1
JOIN {{ ref('source_3') }} AS T3
  ON T1.date_forecast = T3.forecast_date
LEFT JOIN {{ ref('source_2') }} AS T2
  ON T1.date_forecast = CAST(T2.start_time AS DATE)
GROUP BY
  T1.date_forecast,
  T1.location
```