PokéBinder: An Interactive Web Experience for Pokémon Card Collectors

Álvaro Enrique Martín Chango

June 30, 2025

Resumen– Este documento presenta el desarrollo de PokéBinder, una aplicación web diseñada para permitir a los usuarios coleccionar, organizar y explorar cartas de Pokémon. El principal objetivo del proyecto ha sido ofrecer una plataforma visualmente atractiva y funcional para los aficionados a Pokémon, permitiéndoles gestionar tanto colecciones predeterminadas como personales. La aplicación se ha desarrollado utilizando tecnologías web, como Vue.js para el frontend, Node.js para el backend y Microsoft SQL Server como base de datos. Además, se han empleado librerías como Pinia (para la gestión del estado) y bcrypt (para el manejo seguro de contraseñas), lo que garantiza una experiencia de usuario robusta y segura. A lo largo del documento se detalla la metodología utilizada (Kanban), el proceso de diseño de la interfaz de usuario y las decisiones técnicas tomadas durante la implementación. El informe concluye con una reflexión sobre los resultados obtenidos y posibles mejoras futuras para la plataforma.
Palabras clave– Pokémon, Aplicación Web, Vue.js, Node.js, Microsoft SQL Server, Kanban, Interfaz de usuario, Coleccionismo de Cartas, Microsoft Project

Abstract– This document presents the development of PokéBinder, a web application designed to allow users to collect, organize, and explore Pokémon cards. The main goal of the project has been to offer a visually appealing and functional platform for Pokémon enthusiasts, enabling them to manage both predefined and personal collections.
The application has been developed using web technologies such as Vue.js for the frontend, Node.js for the backend, and Microsoft SQL Server as the database. Additionally, libraries like Pinia (for state management) and bcrypt (for secure password handling) have been used, ensuring a robust and secure user experience.
The document is going to describe the methodology followed (Kanban), the user interface design process, and the technical decisions made during implementation. The report concludes with a reflection on the results obtained and possible future improvements to the platform.
Keywords– Pokémon, Web Application, Vue.js, Node.js, Microsoft SQL Server, Kanban, User Interface, Card Collection, Microsoft Project

1 Introduction

The collection of Pokémon cards is a hobby that has spread worldwide and continues to attract more and more users. Around this hobby, many collector communities have been created, each focusing on a specific aspect of the cards: some collect cards of a specific Pokémon, others use the cards for gameplay, and so on.

Despite the great popularity of Pokémon card collecting, there are not many specialized websites in this sector that allow for flexible and fully personalized collection management. The vast majority focus on the default sets released by the card game and do not allow users to create their own collections where they can clearly visualize what their collection would look like in real life.

The aim of this Final Degree Project (TFG) is to develop a web platform that allows collectors to view all Pokémon cards released to date and manage their collections effectively, with a high degree of customization. In addition, this project seeks to demonstrate that I have reached the necessary level of maturity as an engineer and have mastered the technologies required to design, implement, and deploy a full-stack application.

Following this introduction, the remainder of the article is organized into a logical progression of topics. First, Section 2 reviews the current state of the art in Pokémon card management platforms. Section 3 outlines the specific ob-

• Contact E-mail: 1635979@uab.cat
• Mention made: Information Technologies
• Tutored work by: Miguel Hernández Cabronero "Àrea de Ciències de la Computació i Intel·ligència Artificial"
• Course 2024/25

jectives we set out to achieve with PokéBinder. Section 4 describes the work methodology, including our choice of the Kanban framework. In Section 5 we present the detailed project planning and timeline. Section 6 discusses the technologies selected for both frontend and backend development. Section 7 walks through the web development process itself, from database schema design to UI prototyping. Section 8 summarizes the results obtained, demonstrating how the integration of all components delivers a cohesive application. Section 9 then explores potential avenues for future enhancements. Finally, Section 10 offers our concluding reflections on the project's accomplishments and lessons learned.

2 State of the Art

Pokémon card collecting has grown significantly in recent years, and today there is a large global community of fans. As a result, several websites and apps have emerged that offer features related to this hobby, such as card browsing, buying and selling, or collection management.

Among the most well-known platforms are TCGPlayer and Cardmarket, which focus on card trading, and Pokellector, which allows users to mark cards as owned or wanted. However, most of these platforms share a common issue: they lack flexibility. They are mainly based on official card sets and don't allow for personalized collection management or clear and attractive visualizations.

There are also very few tools that let users organize their cards according to their own criteria—like a favorite Pokémon, a themed collection, or the way they store cards in real life. Because of this, many collectors end up using homemade solutions like spreadsheets, which are neither visual nor user-friendly.

That's why this Final Degree Project proposes the development of a web platform that allows collectors to view all existing Pokémon cards and manage their collections in a personalized, visual, and realistic way.

3 Objectives of project

The primary goal of this Final Degree Project is to develop a website that enables users to manage their Pokémon card collections in a flexible, visual style. Beyond delivering feature-rich functionality, the project also aims to demonstrate full-stack engineering expertise showing proficiency with frontend frameworks (Vue.js), backend architectures (Node.js with RESTful APIs), and database design (Azure SQL) as well as adherence to best practices in code organization, security, and responsive design. The platform will include the following features:

- Create and organize custom card collections.
- Browse existing Pokémon card sets.
- View detailed information about each card.
- Access links to online stores where the cards can be purchased.

In addition, the site offers a user-friendly, intuitive, and visually appealing interface to enhance the overall experience.

From a technical perspective, the website aims to:

- Be secure against common cyberattacks such as XSS, SQL injection, etc.
- Be responsive, adapting its layout and functionality to different devices and screen sizes.
- Have an adequate response time to queries.

4 Work Methodology

After researching various development methodologies [1], I focused mainly on two approaches: **Waterfall** and **Kanban** [2]. Although Waterfall is a classic and structured model based on a sequential flow of phases, I decided to discard it for several reasons. This model requires defining all the requirements and objectives at the start of the project, which can lead to excessive rigidity and complicate the inclusion of changes or improvements during development. In individual projects where requirements may evolve, this becomes a clear drawback.

Instead, I opted for **Kanban**, an agile and visual methodology that fits much better with my needs. Kanban is based on a continuous workflow, which allowed me to organize and prioritize tasks flexibly, without the strict cycles found in other agile methodologies such as Scrum (which requires meetings, predefined roles, etc.).

Key advantages of using Kanban during the project development:

- **Visual monitoring of progress:** Kanban boards are divided into columns such as *To Do*, *In Progress*, and *Completed*, providing a clear overview of the project's status.
- **Flexible prioritization:** Tasks can be reorganized at any time without negatively affecting the overall workflow.
- **Flow control:** It limits the number of tasks in progress, helping to avoid multitasking and improving focus.
- **Ease of implementation:** There are many free and intuitive platforms available, such as *Trello*,*ClickUp* ... which allow effective task management through Kanban.

In addition, Kanban enabled me to adapt quickly to unexpected situations that arose during development. Its modular and visual nature made it easy to reorganize tasks, incorporate new features, or resolve technical issues without losing sight of overall project progress.

In conclusion, I believe that **Kanban was the most suitable methodology** for the development of this website. It allowed for a clear, flexible, and efficient work structure that contributed significantly to maintaining productivity and ensuring the quality of the final result.

5 Project Planning

This Final Degree Project required approximately 300 hours of work. To meet this time constraint, it was essential to establish a solid planning structure from the beginning. As mentioned in the methodology section, I followed a Kanban-based approach, which allowed for flexible organization of the project's different stages and better tracking of its actual progress.

Before starting the development phase, I defined the system requirements for the website. This initial step helped me understand the workload and adjust the project to my needs and the available time.

5.1 System Requirements

The system requirements were divided into two main categories: functional and non-functional.

5.1.1 Functional Requirements

These requirements described the main features the platform needed to offer:

- User registration and account management.
- Creation and customization of user collections.
- Detailed view of selected cards.
- Links to online stores where the cards could be purchased.
- Ability to add cards to the user's collections.
- View of official predefined collections.
- View of collections created by the user.
- Option to mark owned cards within each collection.
- Card search system by name.
- Offer a card recommendation to the users.

5.1.2 Non-Functional Requirements

These requirements described technical constraints and quality criteria for the website :

- Ensure appropriate response times: during development, the entire stack (frontend, API and Azure SQL Server) runs locally, so the only measurable latency is the web/API-to-database round-trip. Therefore, it is essential to guarantee consistently low response times to deliver a great user experience.
- Provide security against common cyberattacks (e.g., XSS, SQL injection).
- Implement a responsive design adaptable to different screen sizes and devices.

5.2 Project Phases

Based on the initial requirements and a Gantt diagram, the following work packages were defined [3]:

- **Requirements Analysis :** This first stage focused on researching all aspects related to web development to establish the project foundation. It also included the writing of the initial report.
- **Website Design:** During this phase, I designed the user interface and general architecture of the platform. I created prototypes to visualize the final layout of the site, which served as a reference for the technical development.
- **Web Development:** Once the foundation was established, I started the actual development of the site based on the previously defined specifications. This was the most time-consuming phase and was divided into:
  - Back-end development.
  - Front-end development.
  - Database implementation.
- **Testing and Bug Fixing:** I created a test suite to verify that all functional and non-functional requirements were met and that the platform worked correctly.
- **Documentation and Report Writing:** I wrote the full TFG report, including explanations of the development process and justification of all technical decisions. I also prepared the project poster and the final presentation for the defense.

It is worth noting that the weekly workload varied depending on my availability, so it was difficult to follow a precise weekly schedule. Instead, I adapted the pace according to the needs of each phase.

5.3 Planning Document

To ensure proper organization and optimize the use of time, I created a detailed plan using Microsoft Project [5]. This document outlined the task distribution for each work package, the estimated duration of each task, and their dependencies.

In addition, the Gantt chart generated from this plan provided a clear visualization of the sequence of activities and the time allocation throughout the project.

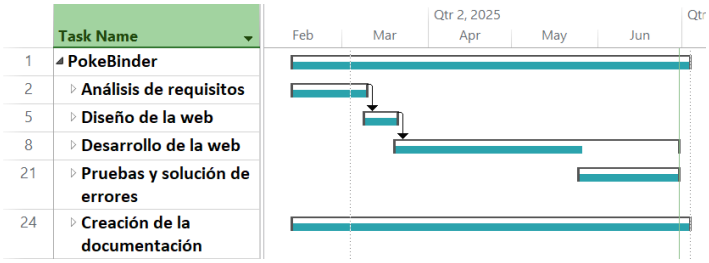

Fig. 1: Gantt digram of the main phases of the project

6 PROJECT TECHNOLOGIES

One of the main tasks carried out after submitting the initial report was to research and choose the right technologies for developing the website. The first and most important decision was to define the architecture and the infrastructure the project would require.

In the initial planning phase, I decided to host both the website and the database on a local network machine where the development would take place [4]. This initial setup allowed for quick testing and full control over the environment, which was useful during the early stages of prototyping. However, this local configuration presented significant limitations in terms of scalability, remote accessibility, data security, and system reliability.

After researching and evaluating alternative solutions, I concluded that using **Microsoft Azure** [6] was the most appropriate choice to host and manage the platform. Azure offers a robust cloud infrastructure with integrated tools for deployment, monitoring, and scaling. By migrating to Azure App Service and Azure SQL Database[7], I was able to benefit from features such as automatic backups, data encryption at rest and in transit, global availability.

Furthermore, the cloud environment greatly improved the performance and accessibility of the application, making it easier to test. This shift also ensured higher availability and resilience against local outages, aligning the infrastructure with standards commonly used in modern web development.

Ultimately, the transition to Azure provided a more secure, maintainable, and scalable environment that supported both the technical needs of the project and the academic requirements for a robust final deliverable.

6.1 Project Infrastructure (Microsoft Azure)

Although Azure simplifies deployment and offers greater reliability, the final decision to use this platform was based on several key points:

- **Security:** With Microsoft Azure, there is no need to open ports on a personal computer, which reduces the risk of cyberattacks. Additionally, Azure allows access to the site to be restricted, making it possible to develop the project as if it were hosted in a secure local environment.
- **Availability:** Hosting the infrastructure in Azure's cloud makes the site accessible from anywhere, instantly, without relying on a personal computer being turned on and waiting for someone to access the server.
- **Cost:** As a UAB student, I had access to an Azure for Students subscription, which includes a €100 credit. This allowed me to cover the costs of web services and complete the project deployment without incurring additional expenses.
- **Service integration:** Another important reason for choosing Azure was the ability to create a database server within the same platform. Having both the web and the database hosted in the same environment made development and organization much more efficient.

6.2 Technologies Used for Web Development

The development of the website was based on the **Model-View-Controller (MVC)** architectural design pattern [8], which provides a clear and structured way to organize the codebase by separating it into three interconnected components:

- **Model:** This layer manages the core data and business logic of the application. In this project, the model is responsible for interacting with the database—handling user data, card information, and collection records through SQL queries and logic encapsulated in Node.js modules.
- **View:** The view is the frontend of the application, built using Vue.js. It defines how data is presented to the user, including forms, lists, card displays, and interactive elements. Vue's component-based structure helped in modularizing the interface and enabled dynamic updates via reactive bindings.
- **Controller:** The controller serves as an intermediary between the model and the view. It receives user input (e.g., login attempts, form submissions), processes it, and invokes the appropriate model operations or renders data to the view. In this project, the backend routes and logic implemented in Node.js (using Express) perform the controller's role.

By using the MVC pattern, the project benefited from a clean separation of concerns. This made the application more scalable and easier to maintain, as changes in one layer (such as updating the user interface or modifying database logic) could be implemented with minimal impact on the others. Additionally, this structure promotes better code readability and facilitates collaborative development, even though this project was carried out individually. It also aligned well with the requirements for modularity and extensibility, ensuring the platform could evolve in future iterations without requiring a complete redesign.

Regarding specific technologies, I chose a set of tools that I considered suitable for a project of this type:

- **Node.js:**[18] This was the chosen technology for developing the backend of the application. Node.js is a widely used JavaScript runtime environment based on a non-blocking, event-driven architecture. This design makes it especially well-suited for handling multiple simultaneous requests with high performance and low resource consumption. In the context of this project, Node.js enabled the creation of lightweight and efficient RESTful APIs, simplifying the development of endpoints for authentication, data queries, and user collection management. Additionally, the use of asynchronous functions and promises streamlined interactions with the database.
- **Vue.js:** [12]Vue.js was used to develop the frontend of the application. It is a progressive JavaScript framework known for its simplicity, reactivity, and component-based architecture. Vue allows the efficient construction of interactive interfaces, and its two-way data binding system along with the virtual DOM

significantly enhance application responsiveness. In this project, Vue.js enabled a modular and maintainable frontend architecture, where components such as login forms, card views, and navigation menus could be reused and updated independently. Its seamless integration with Pinia and routing libraries also simplified session state management and navigation.

- **HTML and CSS:** These fundamental web technologies were used to define the structure (HTML) and visual styling (CSS) of the user interface. HTML5 semantic tags were employed to improve accessibility and code maintainability, while modern CSS techniques such as Flexbox and Grid Layout were applied to achieve a responsive design adaptable to various screen sizes. Special attention was given to user-centered design principles (UX), ensuring visual coherence, good readability, and intuitive navigation. Additionally, the use of CSS variables and scoped styles improved the modularity and consistency of styling across Vue components.

- **Pinia:**[13] As the state management library for the frontend developed with Vue.js, Pinia played a key role by enabling centralized and reactive management of the application's global state. In this project, Pinia was used to track the user session state and associated information (such as ID and email), allowing for secure and synchronized authentication flow management across all relevant components, thereby improving user experience consistency and security.

- **bcrypt:**[15] To ensure secure password management, the bcrypt library was integrated into the backend authentication system. bcrypt is a hashing algorithm specifically designed to protect passwords, applying salting and multiple computationally expensive hashing rounds. This approach makes brute-force or rainbow table attacks significantly more difficult. By using bcrypt to store encrypted password hashes and securely compare them during login, the system ensures that no plain-text passwords are stored and that modern web security standards are met.

- **RESTful API:**[14] A RESTful API architecture was used to structure communication between the frontend and backend. This approach organizes data operations through standard HTTP requests such as GET, POST, PUT, and DELETE. In the Pokebinder project, using a RESTful API facilitated a clear separation of concerns, allowing the Vue.js-based client to efficiently interact with the server and database. Additionally, by using JSON as the data exchange format, communication remains lightweight, human-readable, and easy to maintain. This design choice also allows for future scalability, enabling other clients (like mobile apps) to consume the same API without changes to backend logic.

- **Additional tools and considerations:** Throughout the development process, additional technologies were selectively integrated to maintain a balance between functionality, maintainability, and performance. Complementary libraries—such as middleware for session handling, form validation, among others—were used modularly, only when they provided clear advantages. This approach kept the project lightweight and efficient, leveraging specific tools when they were truly needed.

7 Web Development

Planning has been the main guide throughout the development of the project, always taking into account the initially defined requirements as well as any modifications that arose in the various progress reports. This planning was structured using the Kanban methodology, allowing a visual and flexible control of the status of each task.

7.1 Start of Development: Planning and Design

The first step before starting to write code was the creation of a Kanban board on the ClickUp platform[9]. On this board, the main tasks necessary for the development of the project were defined, grouped into blocks such as: frontend, backend, testing, documentation, and deployment. This board served as a roadmap throughout the entire process, and was progressively updated as tasks were completed, reorganized, or as new needs arose. Thanks to this tool, it was possible to maintain a clear view of the project's status and ensure a continuous workflow.

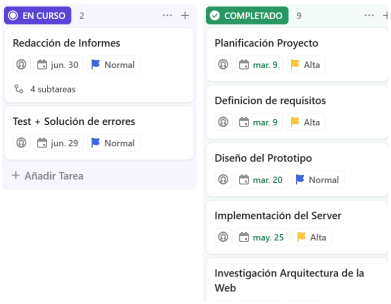

Fig. 2: Kanban Board of the project

In parallel with this planning phase, work was done on the visual design of the user interface. For this, the Canva tool[11] was used to create an initial prototype of the website[10]. This prototype served as a guide to establish the basic interface structure, the overall visual style (colors, fonts, layout of elements), and the desired navigation experience. This approach made it possible to anticipate potential usability issues and laid the foundation for maintaining visual consistency throughout the frontend development process.

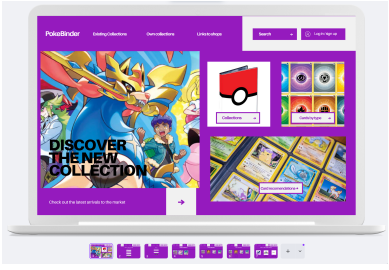

Fig. 3: Prototype of the web

7.2 Database Design

Once the project was completed, it became clear that one of the most decisive elements for its proper functioning was the design and implementation of the database. From the early development stages, it was evident that a solid and well-structured system was required to manage all data related to users, cards, collections, and their interrelations.

The final result is a **relational database hosted on Azure SQL**[7], which serves as the core of the project. This structure ensures **data persistence** and supports both the *functional* and *non-functional requirements* defined during the planning phase. Thanks to this database, the application is capable of offering essential features such as secure user authentication, personalized collection management, and the display of predefined thematic card sets.

Choosing Azure SQL DB was driven by practical advantages: the student account provided convenient free credits for experimentation, provisioning and ongoing management via the Azure portal are extremely straightforward, and the service's built-in performance tuning and low-latency connectivity guarantee consistently fast response times.

7.2.1 Data Models

Throughout the development, the database was structured around the following key entities:

- **Users**: Stores all registered user information, such as user ID, email, encrypted password, city, and favorite Pokémon.
- **Cards**: Contains relevant data for each card, including name, image, type, HP, rarity, and evolution relationships.
- **Collections**: Represents the user's custom-made collections.
- **Collection_Cards**: A join table linking custom collections to the cards they contain.
- **Predefined_Collections**: Includes thematic or recommended collections available by default.
- **Predefined_Collection_Cards**: Join table that associates cards with their corresponding predefined collections.
- **News**: Contains the data related to the news articles displayed on the website.

7.2.2 Entity Relationships

The following table provides a summary of the main relationships between the database entities. This structure was designed to meet the project's requirements and enables smooth and efficient management of both user and card data within the application.

Table 1	Table 2	Relation
Users	Collections	1:N
Users	User_owned_cards	1:N
Collections	Collection_cards	1:N
Predefined_collections	Predefined_collection_cards	1:N
Cards	Collection_cards	1:N
Cards	Predefined_collection_cards	1:N

Table 1: MAIN ENTITY RELATIONSHIPS IN THE PROJECT DATABASE

7.3 Web Development Process

Once the database design was completed, the next step was to begin the development of the website's code. The decision was made to start with the frontend, creating all the necessary Vue components to maximize code reusability. This part of the development was structured into two main folders: **views** and **components**. The views represent the pages visible to the user and form the main interface of the web application. In contrast, the components are reusable code pieces that make up parts of the views, supporting modularity and ease of maintenance.

To verify that everything worked as expected, mock data was used during this phase. This allowed simulating application behavior and ensured that the information was correctly displayed. It is worth noting that some design changes were introduced compared to the original UI prototype, aiming to enhance usability and present a more polished interface.

Once the frontend was complete, the backend development started using Node.js. The main goal of the backend was to handle the requests coming from the frontend and to interact with the database to retrieve or store data accordingly. This part was divided into two main folders: **model** and **controller**. The model folder contains the files responsible for direct communication with the database (e.g., insertions, queries), while the controller folder handles the logic and processing of data before sending the response to the frontend.

It is important to highlight that this part of the system is particularly critical, as it deals with sensitive data. Therefore, special attention was paid to securing the database queries against threats like SQL injection. Additionally, in the context of user management, the project uses the **bcrypt** library, which allows for securely hashing user passwords and protecting them from potential breaches.

7.4 Information Management

One of the fundamental aspects of the website is the information it presents and the structure in which the data is organized. The importance of this lies in the fact that good organization and data handling not only improves system performance, but also has a direct impact on user experience. Presenting data in a clear, accurate, and visually appealing way allows users to navigate the site more intuitively and efficiently.

To display Pokémon cards along with their corresponding details, a JSON file was used containing the data for each card. This file includes information such as the card's name, image, health points (HP), types, abilities, and other relevant characteristics. These data were obtained from a specialized GitHub repository [20] that collects structured

information from multiple official Pokémon cards. Thanks to this approach, the initial database could be populated with a considerable amount of verified, high-quality data.

To automate this process, two custom scripts were developed: one for parsing and inserting the Pokémon card data from the JSON file, and another for loading curated news articles into the database. These scripts allowed bulk processing of the input data, ensuring consistency and significantly reducing the risk of manual entry errors. As a result, the system was initialized quickly and efficiently with all the necessary content.

Furthermore, using JSON as an input format has made it easier to read and process data on both the backend and frontend, allowing seamless integration with database queries and visual components of the website.

Additionally, the entire project code is hosted in a GitHub repository [21], which offers several advantages. Firstly, it facilitates version control, enabling a clear record of all changes made throughout development. Secondly, it provides a cloud-based backup accessible from any device.

7.5 Testing

Once the development of the website was completed and all its main features were implemented, a testing phase began with the aim of verifying the correct functioning of the system and identifying possible bugs or areas for improvement. For this purpose, the user testing [16] technique was used, which involved inviting a small group of testers — specifically my parents, my partner and a close friend — to explore the platform as if they were real users. Each session began with a short oral survey outlining specific tasks they needed to complete (for example, signing up, logging in, and creating a new collection), after which they were free to navigate the site on their own and verbally report any errors encountered or suggest improvements.
During these tests, users performed common tasks such as registering, logging in, browsing collections, or adding cards. This interaction allowed the identification of several aspects requiring attention. For example, unclear error messages were revised, form validation was improved, and small design changes were made to achieve a more coherent and user-friendly interface.

On a technical level, several relevant errors were also detected. One of the main issues was in the management of user sessions from the backend. Specifically, if a user attempted to log in with incorrect credentials, subsequent attempts—even with valid data—were sometimes rejected due to improper session state persistence. This bug was located and resolved, ensuring correct handling of the authentication flow.

Another noteworthy point was the *card recommendation* functionality, which is designed to display a completely random card from the system. During testing, it was detected that in certain cases no card was returned, breaking the intended experience. This error stemmed from a database query that occasionally produced an empty result, even when cards were available. After identifying the issue, the logic was adjusted to ensure that a valid random card is always returned, thus guaranteeing the proper and continuous functioning of this feature.

Thanks to this testing process, it was possible to refine multiple visual and functional aspects, significantly contributing to an improved user experience and greater platform stability.

8 Project Results

The project was developed through several deliverables, including code modules and progress reports. In addition to this final report, three interim documents were produced: the **initial report**, in which the project plan was defined; the **first progress report**, which incorporated revisions to that plan and covered the implementation of the database alongside the web prototype; and the **second progress report**, which described the finalization of the website's development. As a result of this project, a fully functional and coherent web application has been successfully developed from scratch. Every component — from the user interface to the server logic and data layer — has been custom-built to meet the specific objectives defined during the initial planning phase.

8.1 Backend and Database Integration

A key technical foundation of the platform is its dedicated relational database, deployed on a cloud server using Azure SQL [7]. The schema was carefully designed to cover the core elements of the system, including:

- **User information:** securely storing usernames, encrypted passwords, and user preferences.
- **Cards:** maintaining a catalog of collectible Pokémon cards with associated metadata.
- **Collections:** supporting both predefined (e.g., official sets) and user-generated collections.
- **Relationships:** efficiently linking users with the cards they own or wish to collect.

The database also stores news content related to the Pokémon universe, ensuring that users are provided with regularly updated and thematically relevant information. The use of Azure provided the project with essential benefits such as data persistence, cloud-based scalability, and integrated security features like encrypted communication and access management.

8.2 Frontend Development

The visual interface of the site was developed using **Vue.js** [12], a progressive JavaScript framework focused on building interactive and reactive web interfaces. Vue.js was chosen because its intuitive, HTML-based template syntax and shallow learning curve allowed rapid onboarding. The official Vue CLI streamlined project scaffolding and build configuration, while single-file components and a clear separation of concerns kept the codebase clean and maintainable. Moreover, Vue's reactive core and built-in HTTP support integrate effortlessly with our Node.js backend, ensuring smooth data flow, fast UI updates and a highly productive experience. Additionally, it leveraged features such as:

- **Two-way data binding:** allowing real-time updates between the user interface and application state.
- **SPA architecture:** reducing page reloads and improving responsiveness through seamless navigation between views.
- **State management:** using Pinia for centralized and reactive control of session information and application-wide data.

The result is a smooth and visually appealing interface that aligns with the aesthetic of the Pokémon theme while delivering a consistent and intuitive user experience. Figure 4 shows the homepage of the application.

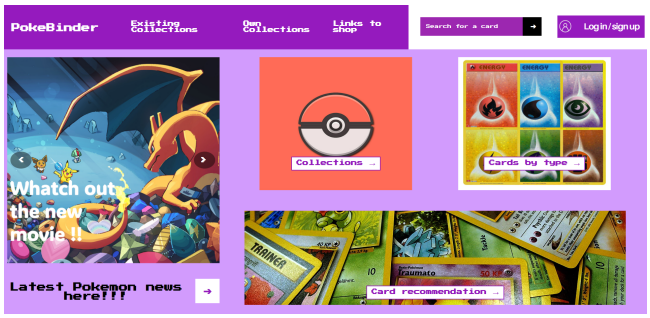

Fig. 4: Homepage of the web application

8.3 Backend Functionality

The server-side logic was implemented using **Node.js** [18], a high-performance JavaScript runtime designed for building fast and scalable network applications. The backend exposes a set of RESTful API endpoints responsible for user authentication, session management, data fetching, and operations on card collections.

Node's asynchronous programming model and use of Promises and middleware enabled efficient request handling and modular code structure. An example of this backend-powered functionality is shown in Figure 5, where a random card from the database is displayed as part of the recommendation system.

Fig. 5: Dynamic content served by the backend

8.4 Summary

In summary, the coordinated integration of frontend, backend, and database layers has led to the successful delivery of a modern, scalable, and user-friendly web application. The implementation adheres to best practices in software architecture (such as MVC), employs secure handling of user data, and is structured in a way that supports future extensibility. The final result fulfills both the functional and non-functional requirements initially defined and represents a solid foundation for potential enhancements and expansion.

9 Future Work

Although the main goals of the project have been successfully achieved, there are several areas that could be expanded or improved in the future to increase the functionality, robustness, and user experience of the platform:

- **Hosting for the web:** Currently, the application runs locally. Although deploying the project to a cloud-based service such as Azure App Service was planned, time constraints during development prevented the full migration. Completing this step in the future would enable public access and real-world usability of the platform.
- **Advanced recommendation system:** Currently, the card recommendation functionality selects a random card. A future improvement would involve implementing a personalized recommendation algorithm based on the user's preferences, collection history, or card popularity.
- **User-to-user interactions:** Introducing features that allow users to interact with each other—such as card trading, messaging, or sharing collections—could significantly enrich the social aspect of the platform.
- **Progress tracking and achievements:** Adding gamified elements such as progress bars, badges, or collection milestones would motivate users to keep expanding their collections and improve user engagement.
- **Mobile-friendly version or mobile app:** Although the current interface is responsive, designing a dedicated mobile application or optimizing the UI specifically for mobile devices would enhance the experience for users accessing the platform on the go.
- **Expand default content:** Increasing the number of predefined cards and news items included in the platform would improve the initial user experience and provide more engaging content out of the box.
- **Feature enhancement:** Some existing features could be extended with new options or better interactivity, such as improving search filters, enhancing the collection sorting mechanism, or adding more detailed card statistics.

These future developments aim to make the platform not only more complete and user-friendly but also scalable and maintainable in the long term.

10 Conclusions

The objectives established during the initial planning phase have been successfully achieved, resulting in a fully functional, modern, and user-centered web application for Pokémon card collection.

An agile methodology (Kanban) proved to be highly effective throughout development, providing flexibility, clear task visualization, and adaptive planning. Tools like ClickUp supported a smooth workflow and continuous project monitoring.

From a technical standpoint, the integration between the frontend (Vue.js), backend (Node.js), and cloud-hosted database (Azure SQL) was completed with care, security, and maintainability in mind. Features such as card browsing, personalized collections, and secure user authentication were implemented and tested successfully.

Although certain tasks—such as the full deployment to Azure App Service—were left pending due to time constraints, the system is technically ready for production and scalable expansion. The project structure allows future iterations to add advanced features like personalized recommendations or user-to-user trading.

Beyond the technical outcome, this project has strengthened my skills in full-stack development, project planning, and decision-making under uncertainty. It also reaffirmed the importance of balancing design, usability, and performance in real-world applications.

In summary, the platform meets the functional and non-functional requirements defined at the beginning and sets a strong foundation for future growth and improvement.

Acknowledgements

I would like to express my sincere gratitude to all the people who have supported me throughout the development of this Final Degree Project.

First and foremost, to my parents, for always being by my side, for offering me their unconditional help, and for providing constant support during the entire process. Their presence has been essential, as a significant part of the success of this project stems from the emotional and practical support they have always given me.

I would also like to thank my project supervisor, Miguel Hernández Cabronero, for his dedication, availability, and valuable advice. In every meeting, he offered useful insights, new perspectives, and alternatives that have been decisive in the successful development of this work.

Finally, my most personal thanks go to my partner, Ingrid Sánchez Ruiz, for her ongoing support, her understanding, and above all, her great patience throughout this journey. Her presence has been key to maintaining motivation and peace of mind, even during the most demanding moments.

Without the help and encouragement of these people, this project would not have been possible, nor as positive an experience as it has been.

References

[1] EALDE Business School. *Agile Methodologies for Project Management: Basic Guide*. Retrieved from https://www.ealde.es/principales-metodologias-agiles/

[2] Asana. (2025). *What is the Kanban methodology and how does it work?* Retrieved from https://asana.com/es/resources/what-is-kanban

[3] Arsys. *Phases in the Web Development Process*. Retrieved from https://www.arsys.es/blog/desarrollo-pagina-web

[4] DesarrolloWeb.com. *Hosting a Website on a Home Computer*. Retrieved from https://desarrolloweb.com/faq/alojar-web-en-ordenador-casa

[5] Microsoft Project Tutorial. *Gantt Chart – GanttPRO Blog*. Retrieved from https://blog.ganttpro.com/es/tutorial-de-microsoft-project/

[6] Microsoft Azure. *App Service*. Retrieved from https://azure.microsoft.com/es-es/products/app-service

[7] Microsoft Azure. *Azure SQL Database*. Retrieved from https://azure.microsoft.com/es-es/products/azure-sql/database/

[8] MDN Web Docs. *MVC - Glossary of MDN Web Docs*. Retrieved from https://developer.mozilla.org/es/docs/Glossary/MVC

[9] ClickUp. *Streamline Workflows with Board View*. Retrieved from https://clickup.com/features/kanban-board

[10] Emprende A Conciencia. *Prototype Design – Tool*. Retrieved from https://www.emprendeaconciencia.com/herramientas/diseno-prototipo/

[11] Fernández, Y. (2023, June 9). *What is Canva, how does it work, and how to use it to create a design*. Xataka. Retrieved from https://www.xataka.com/basics/que-canva-como-funciona-como-usarlo-para-crea

[12] Vue.js. *Vue.js – The Progressive JavaScript Framework*. Retrieved from https://vuejs.org/

[13] Defining a Store — Pinia. *Pinia — The intuitive store for Vue.js*. Retrieved from https://pinia.vuejs.org/core-concepts/

[14] Amazon Web Services, Inc. *What is RESTful API? – RESTful API Explained*. Retrieved from https://aws.amazon.com/what-is/restful-api/#:~:text=RESTful%20API%20is%20an%20interface,applications%20to%20perform%20various%20tasks.

[15] VPN Unlimited. *What is Bcrypt – Cybersecurity Terms and Definitions*. Retrieved from https://www.vpnunlimited.com/es/help/cybersecurity/bcrypt

[16] What is User Testing? Definition, Types & Methods – Trymata. *Trymata*. Retrieved from https://trymata.com/blog/what-is-user-testing/#:~:text=User\%20testing\%20is\%20a\%20systematic,the\%20product\%20meets\%20user\%20needs.

[17] DIGITAL55. *What are Single-Page Applications (SPA)? The development approach chosen by Gmail and LinkedIn*. Retrieved from https://digital55.com/blog/que-son-single-page-application-spa-des

[18] Node.js. *Node.js — Run JavaScript Anywhere*. Retrieved from https://nodejs.org/es

[19] GitHub. *Build and ship software on a single, collaborative platform*. Retrieved from https://github.com/

[20] Pokemon GitHub. *pokemon-tcg-data/cards/en at master · PokemonTCG/pokemon-tcg-data*. Retrieved from https://github.com/PokemonTCG/pokemon-tcg-data/tree/master/cards/en

[21] Alma1616. *web-Pokebinder*. GitHub repository. Retrieved from https://github.com/Alma1616/web-Pokebinder

Appendix A: Visual Evidence of the Project

This appendix presents several screenshots from the PokéBinder web application to visually showcase its functionality, interface design, and various sections accessible to the user.

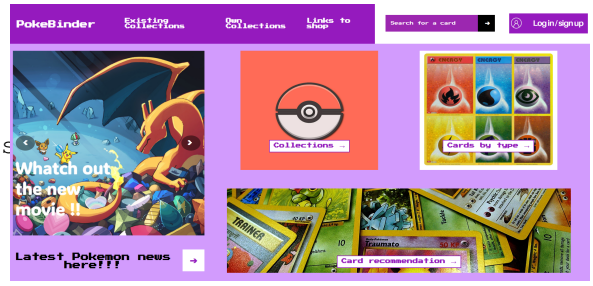

Fig. 6: Homepage of the PokéBinder application

Fig. 7: View of a predefined collection

Fig. 8: Detailed view of a Pokémon card

Fig. 9: Editor for personal collections created by the user

Fig. 10: View of the News tab

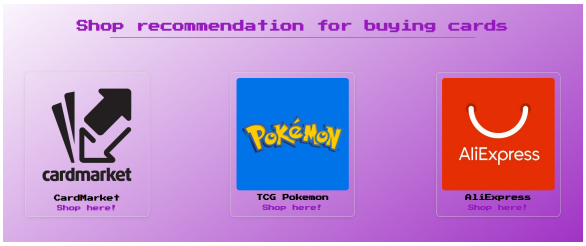

Fig. 11: Shops tab showing links to online card stores

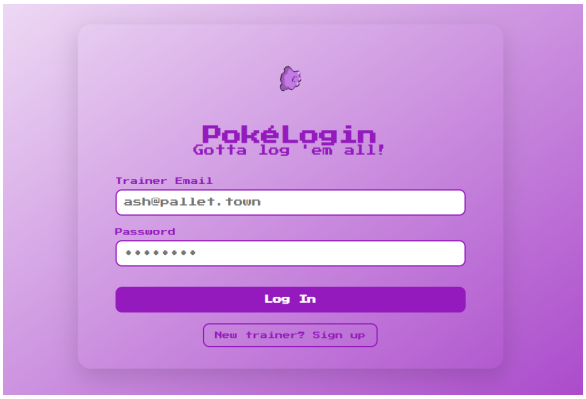

Fig. 12: Login page for registered users

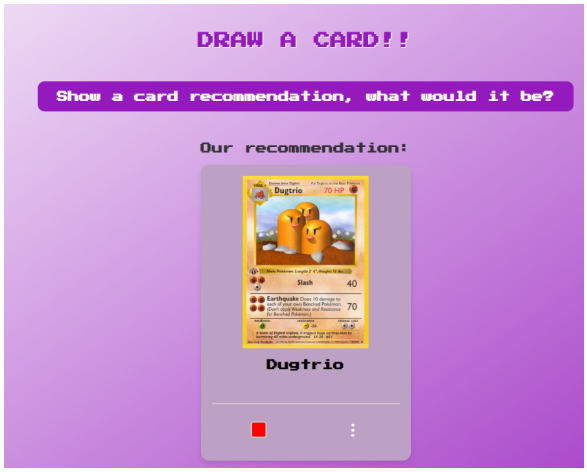

Fig. 14: Random card recommendation tab

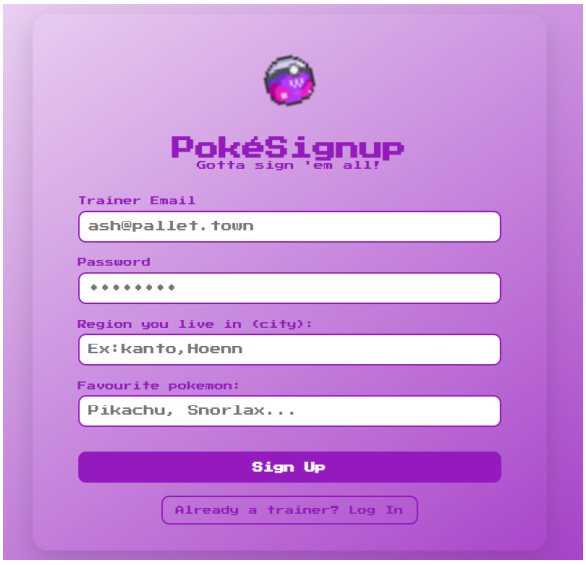

Fig. 13: Signup form for new users