Ismael Prieto Sánchez

Web Application for management and analysis of Cadastre Data: Data Extraction and Management

FINAL DEGREE PROJECT

Directed by Dr Antoni Martínez Ballesté

Bachelor's Degree in Computer Engineering

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2023

Resum.

Aquest projecte pretenia dissenyar una aplicació web per a l'exploració de dades cadastrals. L'objectiu era crear una interfície fàcil d'utilitzar que permeti als usuaris accedir i analitzar la informació de la propietat de manera eficient, permetent-los combinar múltiples fonts de dades.

El projecte global va implicar diversos aspectes del desenvolupament web, inclòs el disseny de bases de dades, el desenvolupament de la interfície d'usuari i les tècniques de gestió de dades.

El projecte es divideix en dos components principals. Una gira al voltant de l'extracció i anàlisi de dades, i l'altra al voltant de la visualització de dades cadastrals mitjançant un sistema d'informació geogràfica. Aquest treball se centra en el component d'extracció i anàlisi de dades que va implicar el desenvolupament de CRUD, inici de sessió i funcionalitats d'importació de dades i la seva interfície d'usuari.

Al llarg del projecte, hem revisat i potenciat diversos conceptes que s'imparteixen al nostre programa de grau. Aquests inclouen aspectes de disseny com l'anàlisi de requisits, el disseny d'aplicacions i les habilitats generals de resolució de problemes.

Resumen.

Este proyecto tuvo como objetivo diseñar una aplicación web para la exploración de datos catastrales. El objetivo era crear una interfaz fácil de usar que permitiera a los usuarios acceder y analizar la información relacionada con propiedades de forma eficiente, permitiéndoles combinar múltiples fuentes de datos.

El proyecto involucró aspectos varios aspectos del desarrollo web, incluyendo diseño de bases de datos, el desarrollo de la interfaz de usuario y técnicas de gestión de datos.

El proyecto se divide en dos componentes principales. Una gira en torno a la extracción y análisis de datos y la otra a través de la visualización de los datos del catastro a través de un sistema información geográfica. Este trabajo se enfoca en el componente de extracción y análisis de datos que involucró el desarrollo de las funcionalidades CRUD, inicio de sesión e importación de datos, sumado a su interfaz de usuario.

A lo largo del proyecto, revisamos y mejoramos varios conceptos obtenidos a lo largo de nuestro programa de grado. Estos incluyeron aspectos de diseño como análisis de requisitos, diseño de aplicaciones y habilidades generales de resolución de problemas.

Abstract.

This project aimed to design a web application for cadastre data exploration. The objective was to create a user-friendly interface that allows users to access

and analyze property information efficiently, allowing them to combine multiple data sources.

The overall project involved various aspects of web development, including database design, user interface development, and data management techniques.

The project is divided into two main components. One revolves around data extraction and analysis, and the other around the visualization of cadastre data via a geographical information system. This thesis focuses on the data extraction and analysis component that involved the development of CRUD, login, and data import functionalities and their user interface.

Throughout the project, we reviewed and enhanced various concepts taught in our degree program. These included design aspects such as requirements analysis, application design, and general problem-solving skills.

Table of Contents

List of Tables

List of Figures

Code List

1 Introduction

In this section, we will provide an overview of the project. We will discuss its components, requirements, and primary objectives.

1.1 Project Description

This project involves implementing a web application offering a comprehensive solution for importing, managing, and analyzing cadastre data. The application was developed as a bachelor's thesis in computer engineering and can be broken down into two main components. The first element focuses on data extraction and management, emphasizing importing and performing operations on data. The second component is visualization and analysis, which displays the data on a map, facilitating the identification of patterns. This thesis will delve into the data extraction and management aspect.

The final application requirements were completed after numerous meetings with Escola Tècnica Superior d'Arquitectura professors. The consultations included an initial discussion to draft the first design and requirements and a presentation to gather feedback. Finally, the team agreed on the final design.

1.1.1 Application Objectives

The primary goal of the application is to offer a reliable and flexible platform for handling and examining cadastre data. It should be effortless to incorporate new data and features while maintaining security. Four essential functions are necessary to achieve this:

1. Importing new data through table expansion or creation.
2. Managing data with a user-friendly interface that allows non-expert users to manipulate database records.
3. Visualizing data with a map-based system that can represent the stored records.
4. Implementing a user authentication system that restricts access to data-modification functions.

1.1.2 Personal Objectives

The primary learning objective of the project was to participate in a real-world environment that allowed the improvement of:

1. Communication skills through contact with clients and teammates during the requirements gathering, design, and development phases.
2. Task organization using roadmaps and deadlines, allowing for early detection of problems in the project development.
3. User-oriented design prioritizing the user experience in the development.
4. Web development skills, learning technologies like HTML, PHP, and CSS.
5. Database engineering aptitudes, focusing on efficiency and usability for frequently used queries during the design.

In conclusion, the project provided a valuable learning experience for improving various skills essential in the tech industry.

2 Background

In this section, we establish the foundation for the rest of the thesis by providing explicit definitions for key concepts that will be used throughout the document. We will begin by defining architecture-related concepts, including sources of information of cadastre data, and then move on to delve in the meaning of CRUD as a database management system used for application development.

2.1 Cadastre

The cadastre serves as a land information system that one or more government agencies manage. It provides a detailed description of most properties in a country. Civilizations initially created these kinds of systems for land taxation purposes. However, with the rise of more complex urban areas and advanced analytical tools, the cadastre has become a crucial resource for various applications, such as development planning and forecasting [1].

2.1.1 Cat Files

In Spain, most cadastre information is accessible to the public, excluding protected data of properties like owner's data. Each province's cadastre information is downloaded separately. CAT files are used to store this information. They are fixed-length files that store different record types. The record type will determine which characters contain information and which are just whitespaces [2].

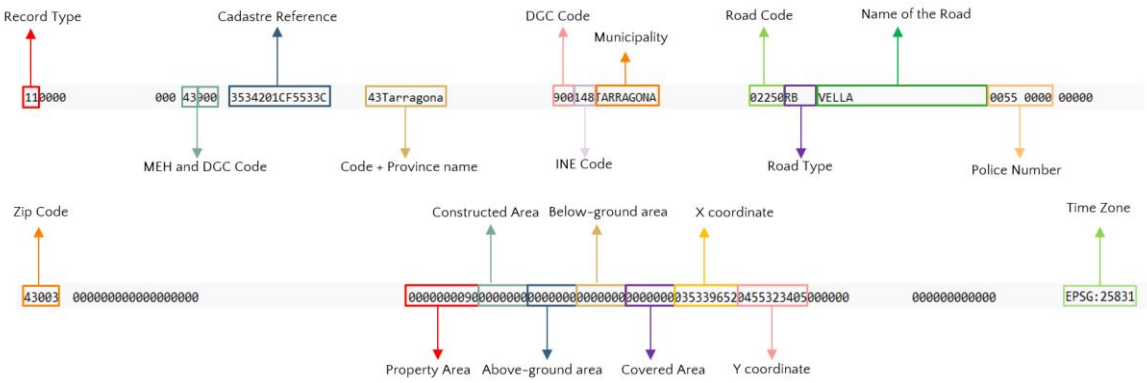


Figure 1. CAT File record

Figure 1 provides a visual representation of a property record from a CAT File. This kind of record is used for the application's data import process, as it contains information on properties like a unique cadastre reference, area, and exact location.

2.2 Other Sources of Information

While useful, more than the information provided by the cadastre is needed for multiple other applications of great use today. Data about businesses like capacity and customer traffic, among others, is invaluable for analysis like the viability of establishments or the decay of the most visited urban zones.

That is why other information sources are needed. We decided to use Google Places API as a data source because it enables "Nearby Search" queries. A JSON result file is generated with a list of all the businesses in that area by specifying a central point and a

radius. The data provided includes type, name, and rating, among other details, and can be further expanded with subsequent queries to meet specific requirements [3].

2.3 Database Management Systems

When managing a database, there are four essential operations: Create, Read, Update, and Delete. These four operations, also known as CRUD, refer to tables or records. However, CRUD has become a widely used term for a user interface that allows users to view and modify data. In this thesis, whenever we mention CRUD or a CRUD system, we refer to a web function that enables viewing and manipulating the database, including the four essential operations.

3 Development Roadmap

This section outlines the roadmap for the project's development, detailing the tasks and challenges of each phase. Finally, we will end discussing the team's coordination.

For the breakdown of project activities, we utilized a waterfall-like approach. Due to time constraints, we started feedback gathering from the client and program testing before finalizing the development of all functionalities.

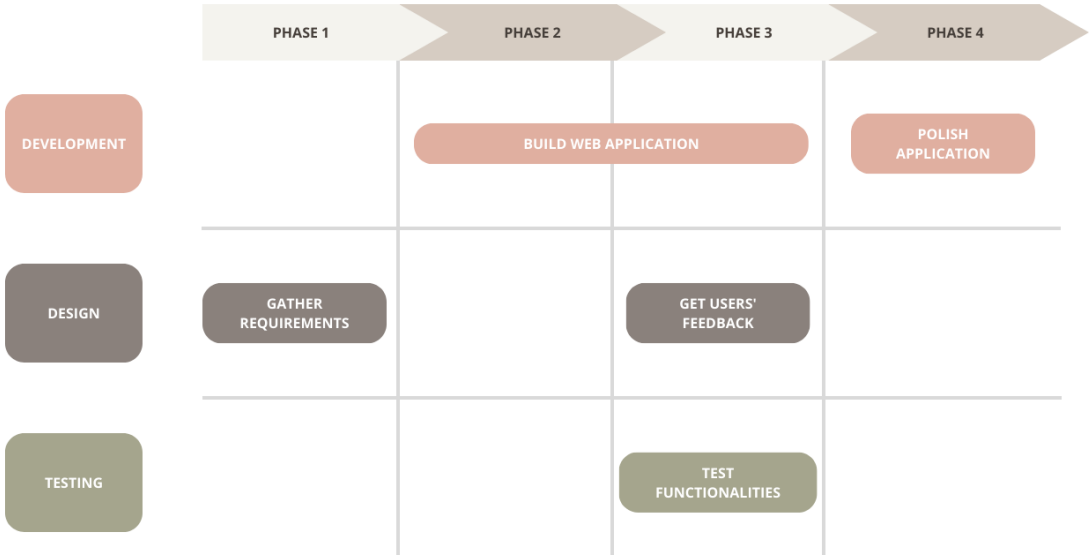


Figure 2. Roadmap of the project

3.1 Phases

3.1.1 Phase 1: Gathering Requirements

During this phase, we held several meetings with the client to identify the users' needs and develop an initial solution. After presenting the possible solution, we gathered feedback from the client and agreed on the final requirements for the product. However, the main challenge we faced was figuring out how the cadastre information was being stored to allow the client to choose which data they wanted to save in the database. We had to carefully study and analyze the file structure before finalizing the design of the product's capabilities.

3.1.2 Phase 2: Prototype Development

During the second phase, we created a web application prototype with essential features and limited data import and visualization capabilities. The main challenge we faced was the restrictions imposed by the hosting service. We had to upload and process CAT files on the server to import data into the database. Nevertheless, due to the limited file size upload and the ample space occupied by CAT files, we needed to develop one new functionality. This latest feature allowed users to download and run an executable file that would remove unnecessary information from CAT files and significantly reduce their size.

3.1.3 Phase 3: Finishing Functionalities

During this third phase, we completed the development of all remaining functionalities. We also conducted testing and gathered user feedback on the finished features, which helped us refine them in the last phase. The only challenges we encountered were related to the significant amount of work and testing required for the bulk of functionalities.

3.1.4 Phase 4: Polishing the Product

The last phase consisted of polishing the features given the users' feedback and completely redesigning the user's interface to give the whole web application a cohesive and intuitive look. There were no significant challenges.

3.2 Team’s Coordination

To ensure coordination, our team held weekly meetings except on holidays. During these gatherings, we discussed our progress and challenges in the development process, enabling us to set new goals and gain fresh perspectives. We planned both in-person and virtual meetings.

4 General Description of the Project

In this section, we will give a general description of the project going from topics of the environment used for the development, which will include how the server is set up and the technologies we used, to the needs and usage forecast of the application.

4.1 Development Environment

In order to set up the server for the web application, it is essential to have the right tools in place to ensure accessibility, availability, and efficient management. We will now explore the components that make up the server setup for this project and the principal technologies used for the application's development.

4.1.1 Server Setup

We rely on a few key components to set up the server for our web application. Firstly, Dinahosting provides a reliable hosting platform to ensure our application always remains accessible to users. We also use FileZilla, an FTP client that simplifies remote file management, making transferring files between our local machines and the server convenient. Lastly, we utilize phpMyAdmin, a web-based tool that simplifies database management, essential for storing and organizing our application's data. With phpMyAdmin, we get a user-friendly interface that makes it easy to interact with the database.

4.1.2 Application Development

We utilized various tools and technologies to ensure the development of an efficient and organized web application. HTML was used to construct the user interface and structure the content, while CSS was chosen for styling to provide a more personalized and lightweight approach. PHP was used for the back-end development, allowing for dynamic content delivery and a smooth user experience by implementing application logic, processing, and functionality on the server side. We incorporated several functionalities, such as data processing, form handling, user authentication, and database integration. We used MariaDB as our database management system for efficient data storage and retrieval.

4.2 Needs

The major need for the application is to provide an adaptable system that can adjust to the data it is processing. All functionalities should work with any data tables, even if there are changes in data types, to ensure they are relevant and can be used again with different information sources. This ability to reuse functionalities will enhance flexibility when incorporating new features that are tailored to analyzing different data sets.

The application needs to have a user-friendly interface that is straightforward to navigate so that even non-technical users can easily access its features. It should also have a robust security system in place to prevent unauthorized access to potentially dangerous functions. Since the application handles a large amount of data, it must be efficient in its performance by only representing a limited portion of the dataset, such as a single city at a time (unless specified otherwise).

4.3 Usage Forecast

This application is designed for professors at Escola Tècnica Superior d'Arquitectura, so we anticipate a consistent number of users throughout its lifespan. However, if we allow

access to other institutions, the web tools we offer could prove beneficial and increase our user base. Considering the usage of functionalities, we know that despite all features being necessary, some, like data imports, are only required when new data is needed, making this functionality less frequently used than others that manipulate or visualize data.

5 Requirements

This section outlines the necessary components for the web application. We will begin by presenting the functional requirements, including the initial scripts, use case diagram, and textual specifications of each use case. Finally, we will cover the non-functional requirements.

5.1 Functional Requirements

To provide an overview of the functional requirements for the web application, we will present the information redacted during the requirements-gathering phase of the project. This information includes scripts that show how users will interact with the system, a diagram showing how different system actors will work together, and detailed descriptions of each step involved in each use case.

5.1.1 Scripts

5.1.1.1 Normal User's Script

The user can navigate to the GIS[1] interface, showing a map showcasing estate data visualizations. These visualizations will encompass different aspects of the estate data, such as property locations and types. The user can customize the visualizations by selecting the specific data attributes they wish to display and adjusting the map view according to their preferences, enabling zooming and panning functionality.

Moreover, the GIS will offer various filtering options, allowing users to refine the displayed data based on various criteria, empowering the user to interact with the visualizations and explore specific data points or properties. For instance, they can click on a property to access detailed information, facilitating a deeper understanding of the data.

In addition to the GIS, the user will have access to a graphical interface dedicated to data visualizations. This interface will showcase essential data and analysis, including trends and patterns. The graphics system will present diverse visualizations, such as charts, plots, or other appropriate formats, to effectively communicate the insights derived from the data.

Finally, a login system must be in place so the user can access restricted functionalities.

5.1.1.2 Logged in User's Script

Once the user logs in, they can interact with the database through an intuitive CRUD system. This system will allow users to select and switch between database tables. The system must also provide the following operations: create, update, delete, and read, allowing users to add new records, modify existing data, delete unwanted entries, and retrieve relevant information as needed.

Considering the potential presence of large datasets, the system will incorporate pagination functionality. This feature will enable users to manage and navigate multiple data pages, to facilitate dealing with extensive datasets.

[1] GIS: Geographical Information System

The system will allow users to apply filters. This functionality will allow refining the displayed data based on specific criteria.

Additionally, users can delete entire tables when necessary.

Data integration will be facilitated by allowing users to incorporate new data into existing tables using files. This feature will streamline expanding and updating the database by enabling users to import and merge external data sources. Furthermore, the system will support the creation of new tables from files, with the added functionality of linking these tables to the main table for effective data integration.

The system will provide a straightforward option for the user to close the session and return to being a regular user.

5.1.2 Use Cases Diagram

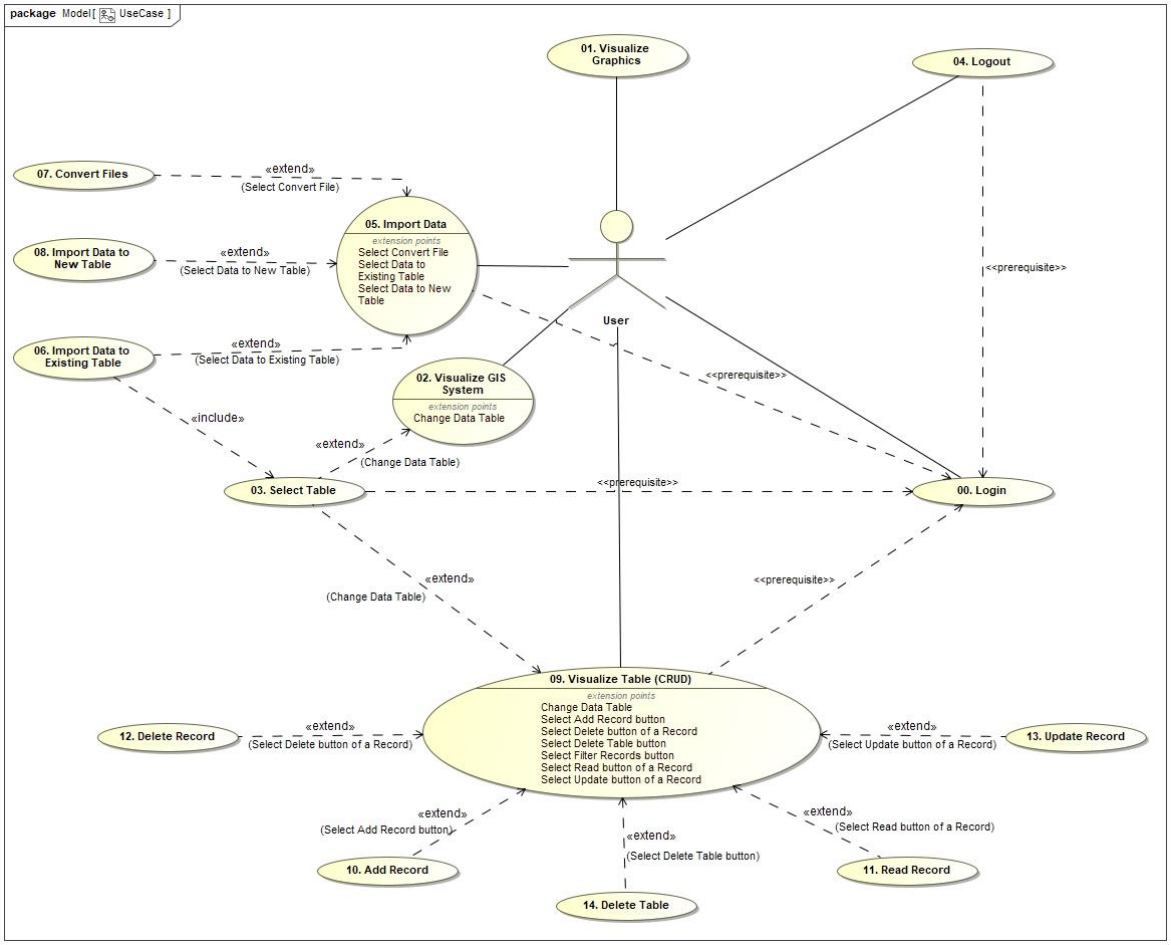


Figure 3. Use cases diagram

5.1.3 Textual Specifications of Use Cases

The following specifications document the desired behavior and interactions within the system. To correctly reflect the work done, we will not include textual specifications and subsequent analysis of use cases of the visualization and analysis component of the project. That is why "Use Case 02. Visualize GIS System" will not be developed further in this thesis.

UC 00. Login

Functionality summary: The user authenticates by providing valid credentials, gaining access to the restricted functionalities of the web application.

Input parameters: Page to return to.

Output parameters: None.

Actors: User.

Precondition: The system has at least a user credential stored in the system.

Postcondition: The user is correctly logged in and can access previously restricted functionalities.

Main flow:

1. The system asks for the credentials of the User.
2. The User introduces the credentials.
3. The system starts a session for the User, indicating that the User is logged in and can access restricted functionalities. If the system has a page to return stored, it returns to that page; else, it will go to the log-out page.

Alternatives of process and exceptions:

2a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

3a. The credentials of the User do not match any of the database credentials.

3a1. The system asks for the User's credentials while displaying an error message.

3a2. The system returns to step 2.

UC 01. Visualize Graphics

Functionality summary: The user visualizes essential graphics created from the data stored in the database.

Input parameters: None.

Output parameters: None.

Actors: User.

Precondition: None.

Postcondition: The essential graphics are displayed to the user, who can interact with them.

Main flow:

1. The system retrieves the data for the visualization from the database and presents it in graphical format.
2. The User interacts with the visualization, viewing the content, zooming, or panning.
3. The User can navigate to other pages via the navigation menu.

Alternatives of process and exceptions: None.

UC 03. Select Table

Functionality summary: The user selects one of the tables presented by the system to work with. Once selected, the user returns to the page they came from with the information of the set table.

Input parameters: Page to return to.

Output parameters: Selected Table.

Actors: User.

Precondition: The user is logged into the system.

Postcondition: The user is returned to the previous page from which they accessed the Select Table functionality with the information of the set table.

Main flow:

1. The system retrieves the data of all table names available and shows them to the User, asking them to select one.
2. The User selects one of the tables shown.
3. The system takes the User to the page to return to with the information of the set table.

Alternatives of process and exceptions:

2a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

UC 04. Logout

Functionality summary: The user logs out, losing access to the restricted functionalities of the web application.

Input parameters: None.

Output parameters: The user is logged out.

Actors: User.

Precondition: The user is logged into the system.

Postcondition: The user no longer can access restricted functionalities.

Main flow:

1. The system asks the User if they want to log out.
2. The User selects the button to log out.
3. The system finishes the User session, so they cannot access restricted functionalities.

Alternatives of process and exceptions:

2a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

UC 05. Import Data

Functionality summary: The user imports data to the database via file upload or transforms a file to upload later.

Input parameters: None.

Output parameters: None.

Actors: User.

Precondition: The user is logged into the system.

Postcondition: Either the user imported new data to the database, or the user has converted a file for later uploading.

Main flow:

1. The system asks the User which possible import or convert functionalities they want to perform.
2. The User selects one of the functionalities executing one of the following use cases: UC 06 Import Data to Existing Table, UC 07 Convert Files, or UC 08 Import Data to New Table.

Alternatives of process and exceptions:

2a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

UC 06. Import Data to Existing Table

Functionality summary: The user imports data to an existing database table via a file upload.

Input parameters: None.

Output parameters: None.

Actors: User.

Precondition: The user is logged into the system.

Postcondition: The user imported new data to an existing database table.

Main flow:

1. The system asks the User which file type they want to upload.
2. The User selects the file type to upload.
3. The system executes UC 03. Select Table.
4. The system asks the User to upload the files.
5. The User uploads the file.
6. The system processes the file and imports the data to the existing table.

Alternatives of process and exceptions:

2a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

4a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

5a. The file is not correct for the task.

5a1. The system informs of the error and asks the User to try again with a return button.

5a2. The User presses the button, returning to step 4.

UC 07. Convert Files

Functionality summary: The user uploads a JSON file and downloads the conversion to CSV.

Input parameters: None.

Output parameters: None.

Actors: User.

Precondition: The user is logged into the system.

Postcondition: The user received the converted CSV file.

Main flow:

1. The system asks the User to upload the JSON file.
2. The User uploads the file.
3. The system converts the file and automatically downloads it.

Alternatives of process and exceptions:

2a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

3a. The file is not correct for the task.

3a1. The system informs of the error and asks the User to try again with a return button.

3a2. The User presses the button, returning to step 1.

UC 08. Import Data to New Table

Functionality summary: The user imports data to a new database table via a file upload.

Input parameters: None.

Output parameters: None.

Actors: User.

Precondition: The user is logged into the system.

Postcondition: The user imported new data to a new database table.

Main flow:

1. The system asks the User which file type they want to upload.
2. The User selects the file type to upload.
3. The system asks the User to upload the file.
4. The User uploads the file.
5. The system processes the file, creates a new table on the database, and inserts the data.

Alternatives of process and exceptions:

2a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

4a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

5a. The file is not correct for the task.

5a1. The system informs of the error and asks the User to try again with a return button.

5a2. The User presses the button, returning to step 3.

UC 09. Visualize Table (CRUD)

Functionality summary: The user visualizes and interacts with data from a database table.

Input parameters: Set Table, Set Page, Set Filters.

Output parameters: None.

Actors: User.

Precondition: The user is logged into the system.

Postcondition: The user has interacted with the data.

Main flow:

1. The system checks the input parameters (if not set, uses default values), retrieves the corresponding data, and presents it to the User with buttons to interact.
2. The User can select another page, filters or any buttons leading them to the following functionalities: UC 03. Select Table, UC 10. Add Record, UC 11. Read Record, UC 12. Delete Record, UC 13. Update Record, UC 14. Delete Table.

Alternatives of process and exceptions:

1a. The Set Table is not in the database or is not a valid table.

1a1. The system shows an error message and asks the user to return with a button.

1a2. The User presses the button and returns to step 1 with the default table**.**

2a. The User selects a page button from the navigate the data buttons or sets some filters.

2a1. The system returns to step 1 with the new Set Page or Set Filters.

2b. The User selects one of the page buttons from the navigation bar, leaving the functionality.

UC 10. Add Record

Functionality summary: The user adds a new record to an existing database table.

Input parameters: Set Table.

Output parameters: None.

Actors: User.

Precondition: The user is logged into the system.

Postcondition: A new record is added to an existing database table.

Main flow:

1. The system checks the Set Table, retrieves information about its columns, and creates a form for the user to add the data for the new record.
2. The User adds the data and sends it to the system.
3. The system processes the information, checks it is correct, and adds the new record. Then asks the User to return to the last page.

Alternatives of process and exceptions:

1a. The Set Table is not in the database or is not a valid table.

1a1. The system shows an error message and asks the user to return with a button.

1a2. The User presses the button and executes UC 09. Visualize Table (CRUD) with the default table.

2a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

3a. The data introduced by the user is not valid.

3a1. The system shows an error message and asks the user to return with a button.

3a2. The User presses the button and executes UC 09. Visualize Table (CRUD) with the default table.

UC 11. Read Record

Functionality summary: The user visualizes the information of a selected record in an existing database table.

Input parameters: Set Table, Set ID.

Output parameters: None.

Actors: User.

Precondition: The user is logged into the system.

Postcondition: The user visualizes the record information.

Main flow:

1. The system checks the Set Table and Set ID, retrieves information about its values, and presents it to the user.
2. The User visualizes the data.

Alternatives of process and exceptions:

1a. The input parameters are not correct.

1a1. The system shows an error message and asks the user to return with a button.

1a2. The User presses the button and executes UC 09. Visualize Table (CRUD) with the default table.

UC 12. Delete Record

Functionality summary: The user deletes an existing record from a database table.

Input parameters: Set Table, Set ID.

Output parameters: None.

Actors: User.

Precondition: The user is logged into the system.

Postcondition: The selected record is deleted from the database table.

Main flow:

1. The system checks the Set Table and Set ID and asks the User if they are sure about deleting the record.
2. The User presses the button to delete the record.
3. The system processes delete of the record and informs the User of the success asking them to return to the last page.

Alternatives of process and exceptions:

1a. The input parameters are not correct.

1a1. The system shows an error message and asks the user to return with a button.

1a2. The User presses the button and executes UC 09. Visualize Table (CRUD) with the default table.

2a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

UC 13. Update Record

Functionality summary: The user updates the information of an existing record from a database table.

Input parameters: Set Table, Set ID.

Output parameters: None.

Actors: User.

Precondition: The user is logged into the system.

Postcondition: The selected record information is updated in the database table.

Main flow:

1. The system checks the Set Table and Set ID and presents the User with a form to introduce the new data.
2. The User fills the form with the new data and sends it to the system.
3. The system processes the new data, updates it in the database table, and informs the User of the success. Then asks the User to return to last page.

Alternatives of process and exceptions:

1a. The input parameters are not correct.

1a1. The system shows an error message and asks the user to return with a button.

1a2. The User presses the button and executes UC 09. Visualize Table (CRUD) with the default table.

2a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

UC 14. Delete Table

Functionality summary: The user removes a database table.

Input parameters: Set Table.

Output parameters: None.

Actors: User.

Precondition: The user is logged into the system.

Postcondition: The selected table is dropped from the database.

Main flow:

1. The system checks the Set Table and asks the User if they are sure about the action.
2. The User presses the button to delete the table.
3. The system processes delete of the table and informs the User of the success. Then asks the User to return to last page.

Alternatives of process and exceptions:

1a. The input parameters are not correct.

1a1. The system shows an error message and asks the user to return with a button.

1a2. The User presses the button and executes UC 09. Visualize Table (CRUD) with the default table.

2a. The User selects one of the page buttons from the navigation bar, leaving the functionality.

5.2 Non-functional requirements

The following specifications outline the non-functional requirements that define the system's expected qualities, constraints, and characteristics.

5.2.1 Security

- User passwords shall be stored securely using industry-standard encryption algorithms.
- The system shall implement role-based access control to restrict unauthorized access to sensitive information.
- The system must implement strict access controls and user permissions to ensure that even users with higher access privileges cannot modify protected data.
- User sessions should automatically expire after a time of inactivity.

5.2.2 Performance

- The system shall ensure fast processing of data, including efficient execution of queries for importing new data and generating visualizations.

5.2.3 Usability

- The user interface shall be intuitive and easy to navigate, minimizing user errors.
- The user interface should be consistent across different pages and sections of the application.
- Help documentation and contextual tooltips should be provided to assist users.
- Error messages should be user-friendly and provide guidance for resolving issues.

5.2.4 Compatibility

- The system should be compatible with the latest versions of popular web browsers.
- The system should be accessible from mobile devices.

5.2.5 Maintainability

- The code should follow industry-standard coding conventions and be well-documented.
- The software should be modular and allow for future enhancements or modifications without significant rework.

6 Requirement Analysis

This section focuses on presenting the results of our analysis, which involved the creation of essential diagrams such as activity, class, and sequence for all the use cases we specified textually.

UC 00. Login

Activity diagram

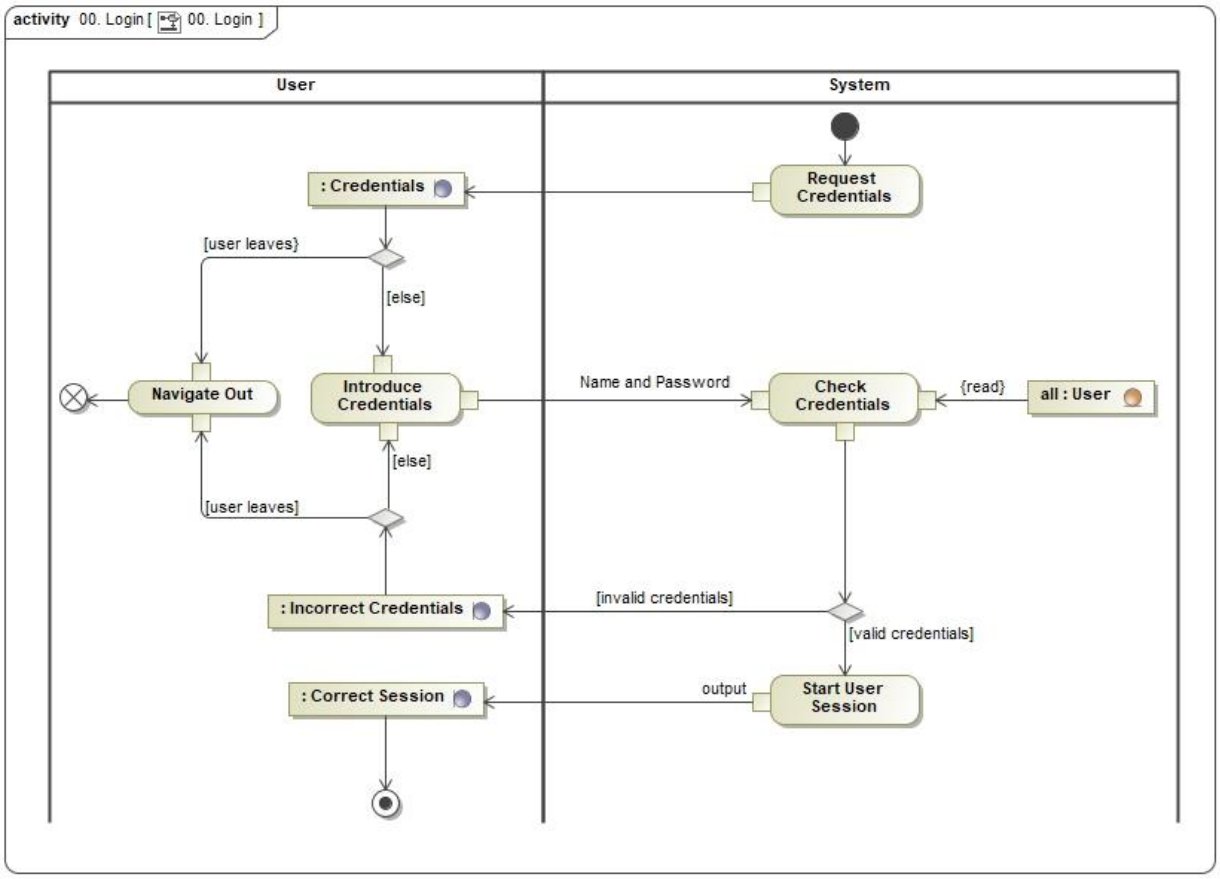


Figure 4. Activity diagram UC 00. Login

Class diagram

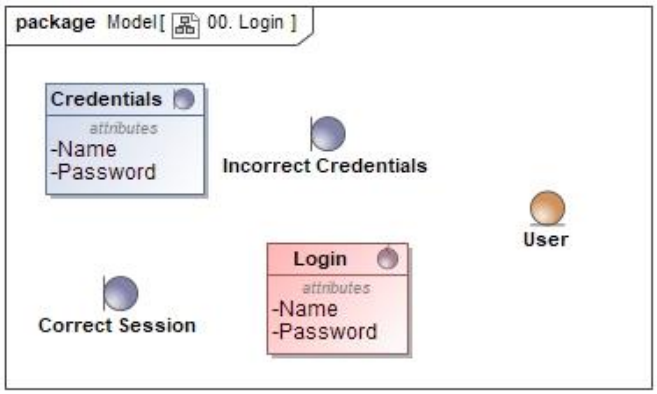


Figure 5. Class diagram UC 00. Login

Sequence diagram

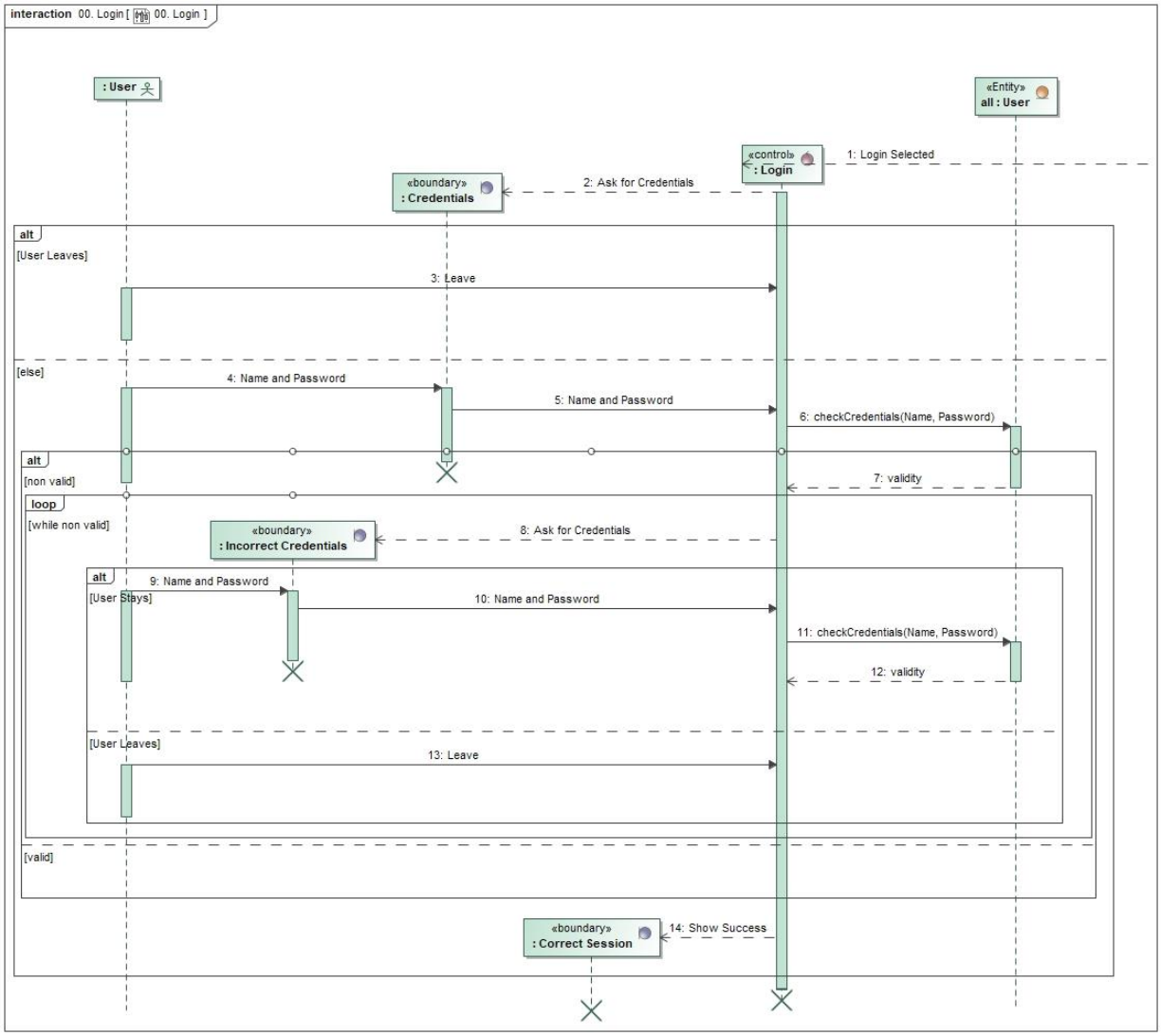


Figure 6. Sequence diagram UC 00. Login

UC 01. Visualize Graphics

Activity diagram

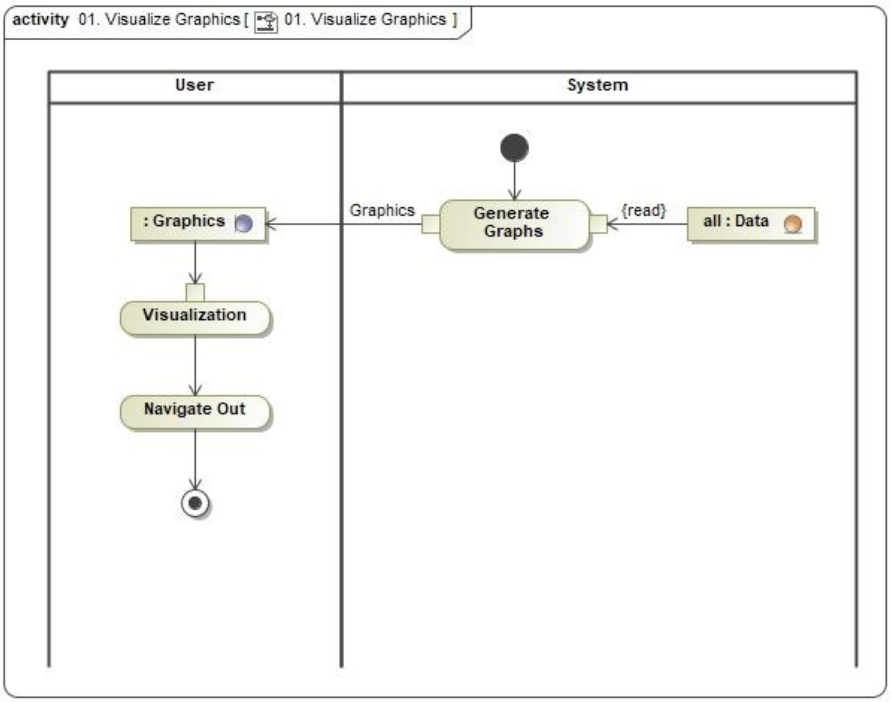


Figure 7. Activity diagram UC 01. Visualize Graphics

Class diagram

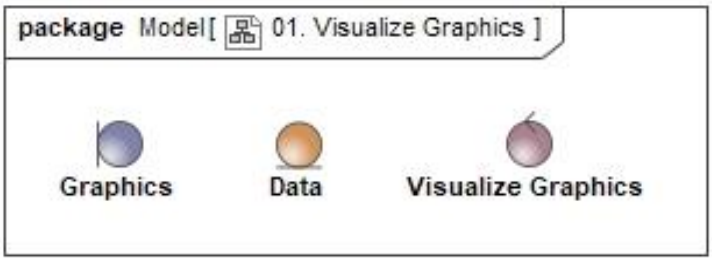


Figure 8. Class diagram UC 01. Visualize Graphics

Sequence diagram

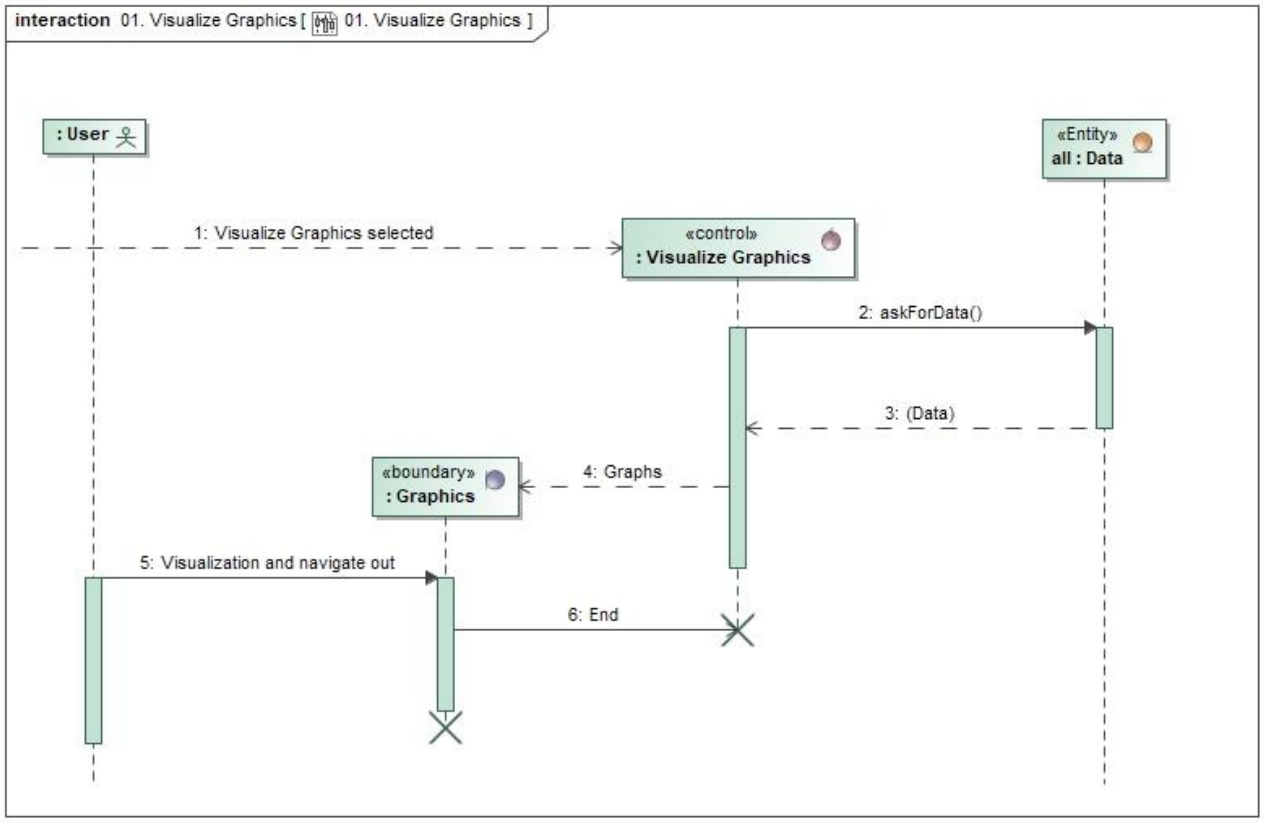


Figure 9. Sequence diagram UC 01. Visualize Graphics

UC 03. Select Table

Activity diagram

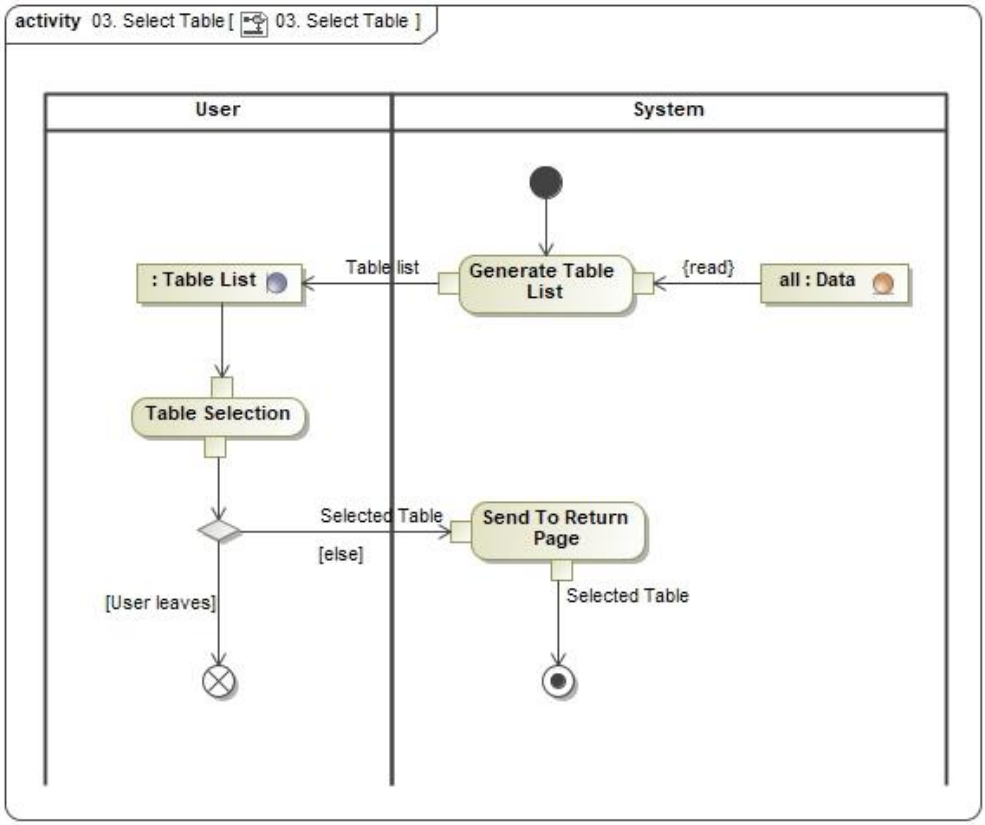


Figure 10. Activity diagram UC 03. Select Table

Class diagram

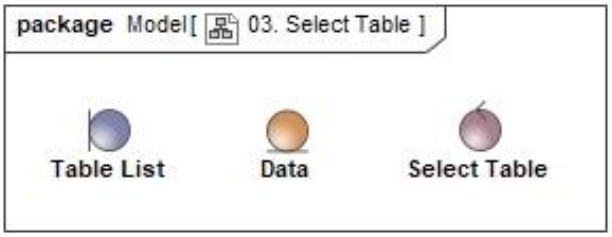


Figure 11. Class diagram UC 03. Select Table

Sequence diagram

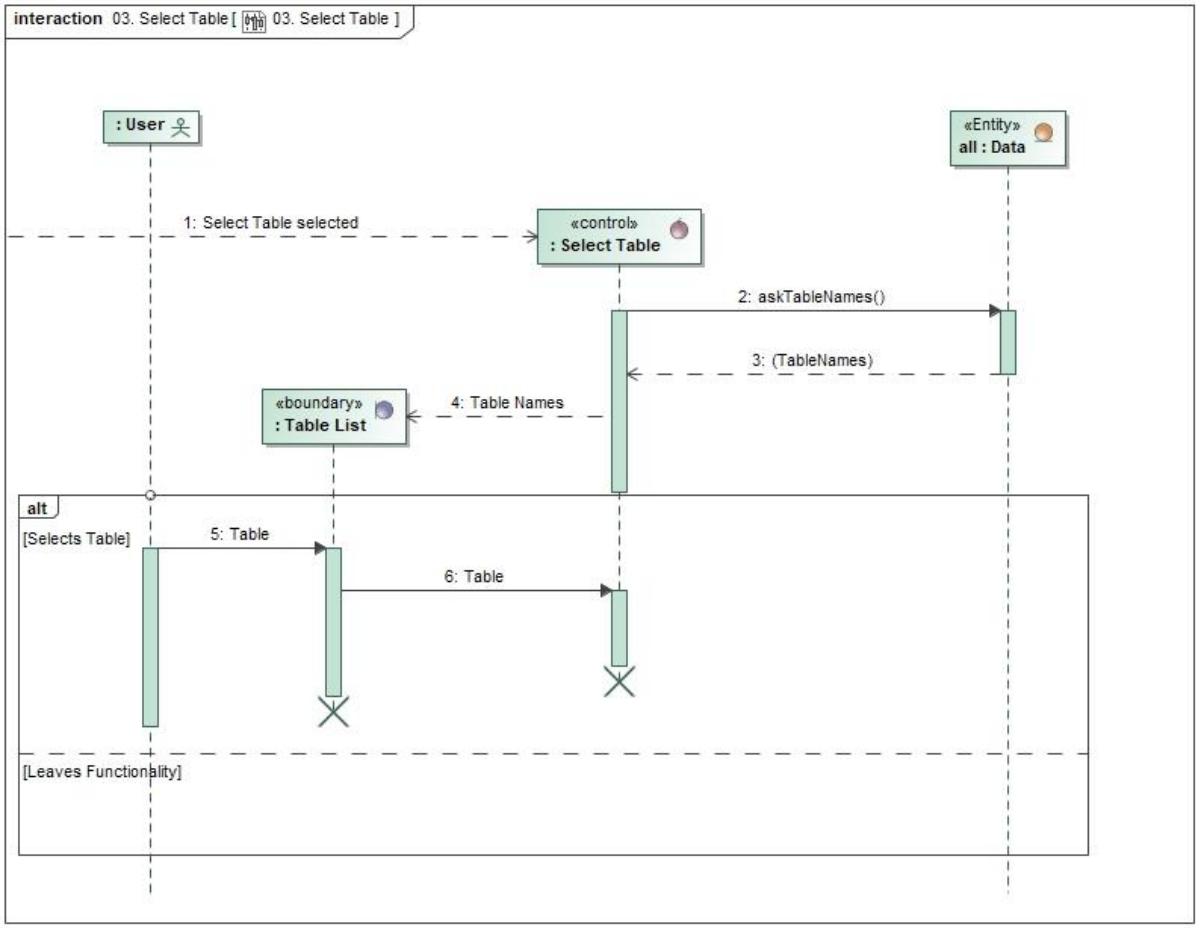


Figure 12. Sequence diagram UC 03. Select Table

UC 04. Logout

Activity diagram

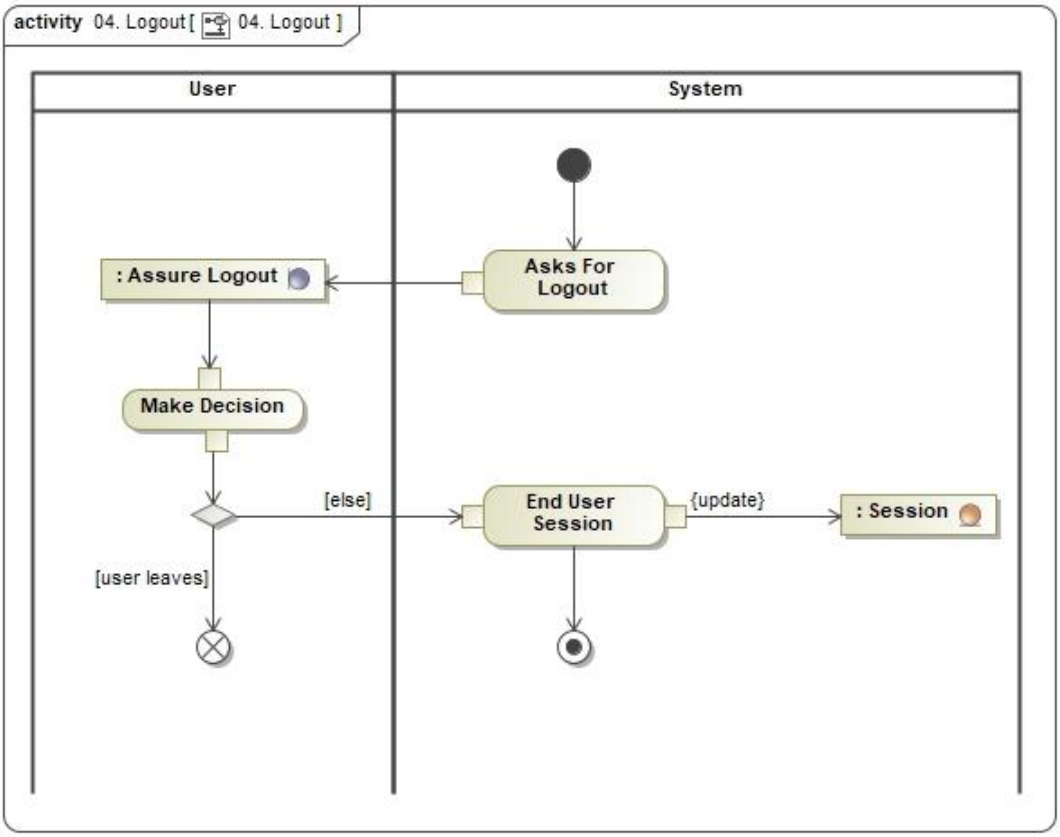


Figure 13. Activity diagram UC 04. Logout

Class diagram

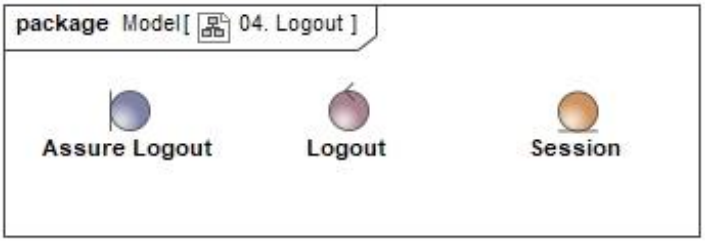


Figure 14. Class diagram UC 04. Logout

Sequence diagram

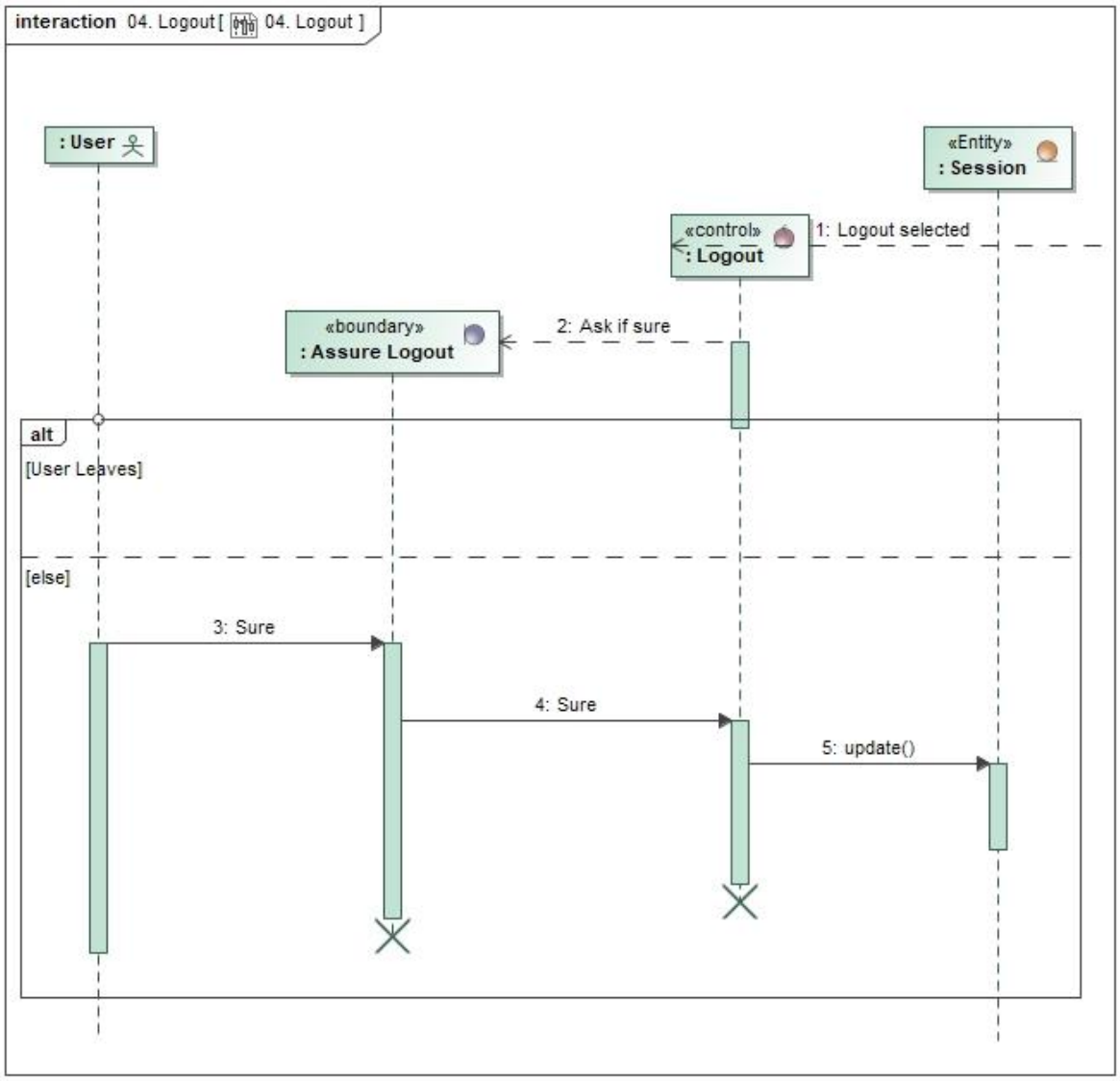


Figure 15. Sequence diagram UC 04. Logout

UC 05. Import Data

Activity diagram

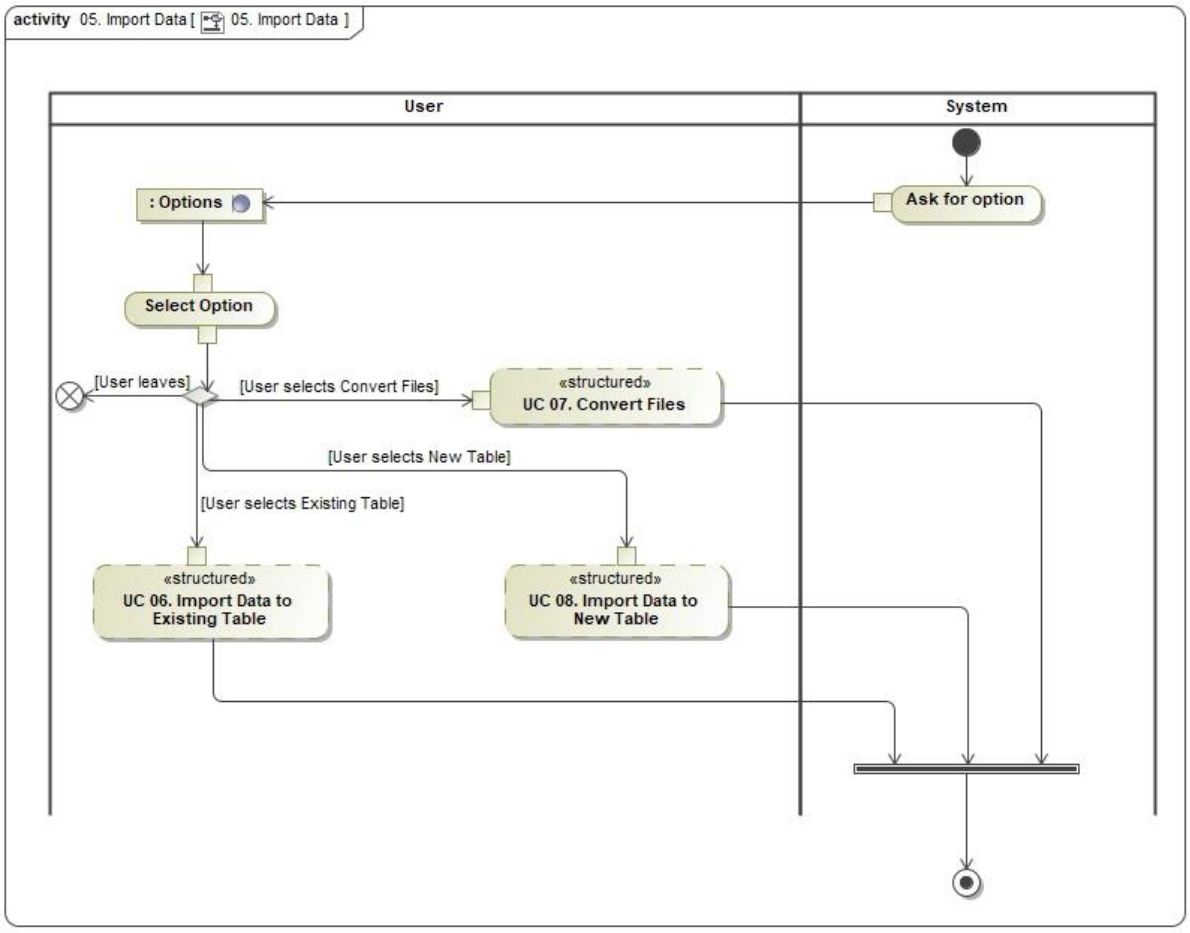


Figure 16. Activity diagram UC 05. Import Data

Class diagram

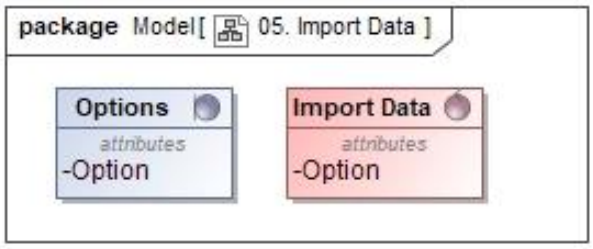


Figure 17. Class diagram UC 05. Import Data

Sequence diagram

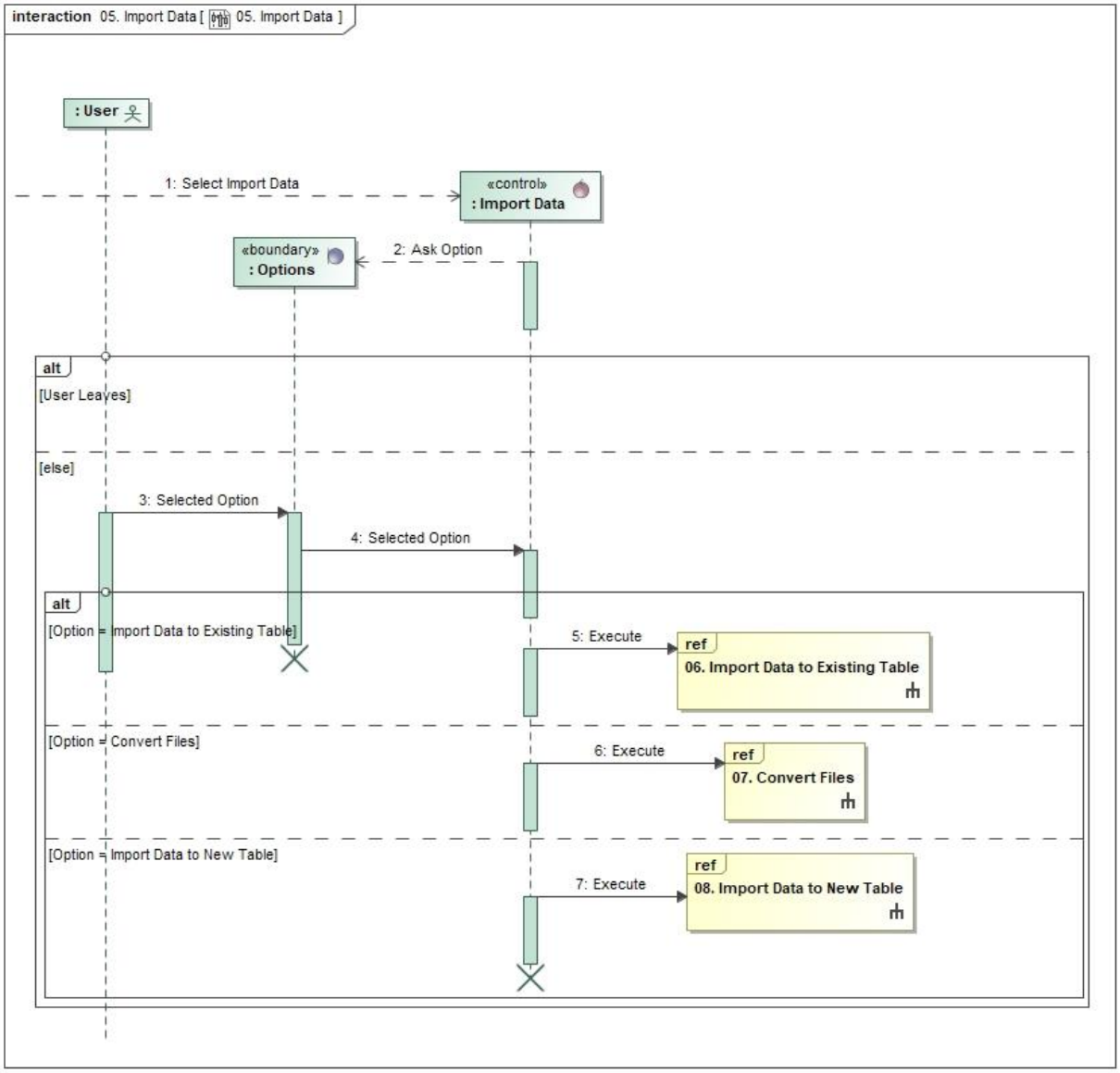


Figure 18. Sequence diagram UC 05. Import Data

UC 06. Import Data to Existing Table

Activity diagram

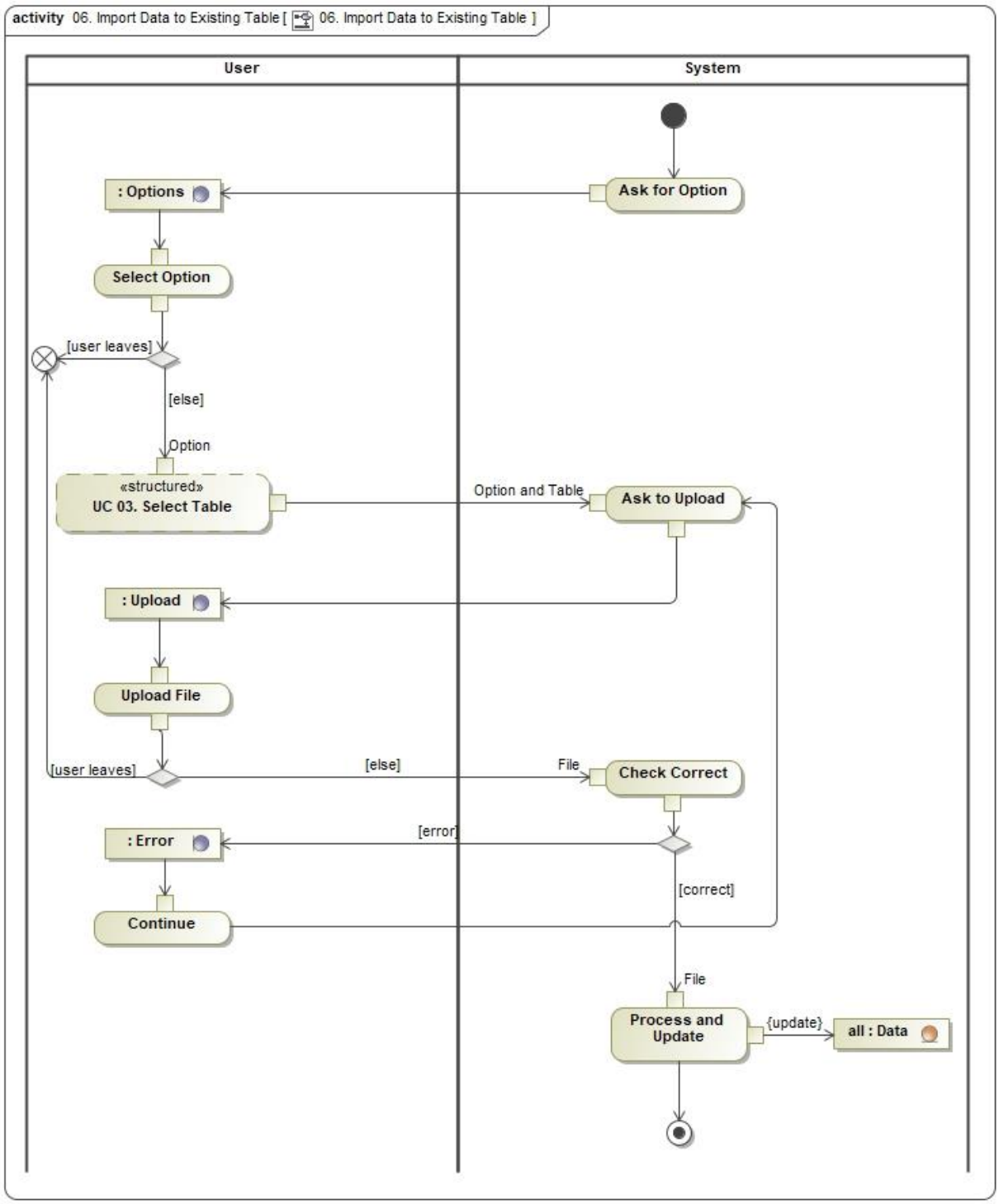


Figure 19. Activity diagram UC 06. Import Data to Existing Table

Class diagram

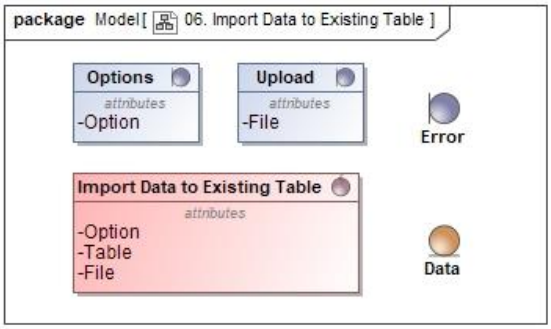


Figure 20. Class diagram UC 06. Import Data to Existing Table

Sequence diagram

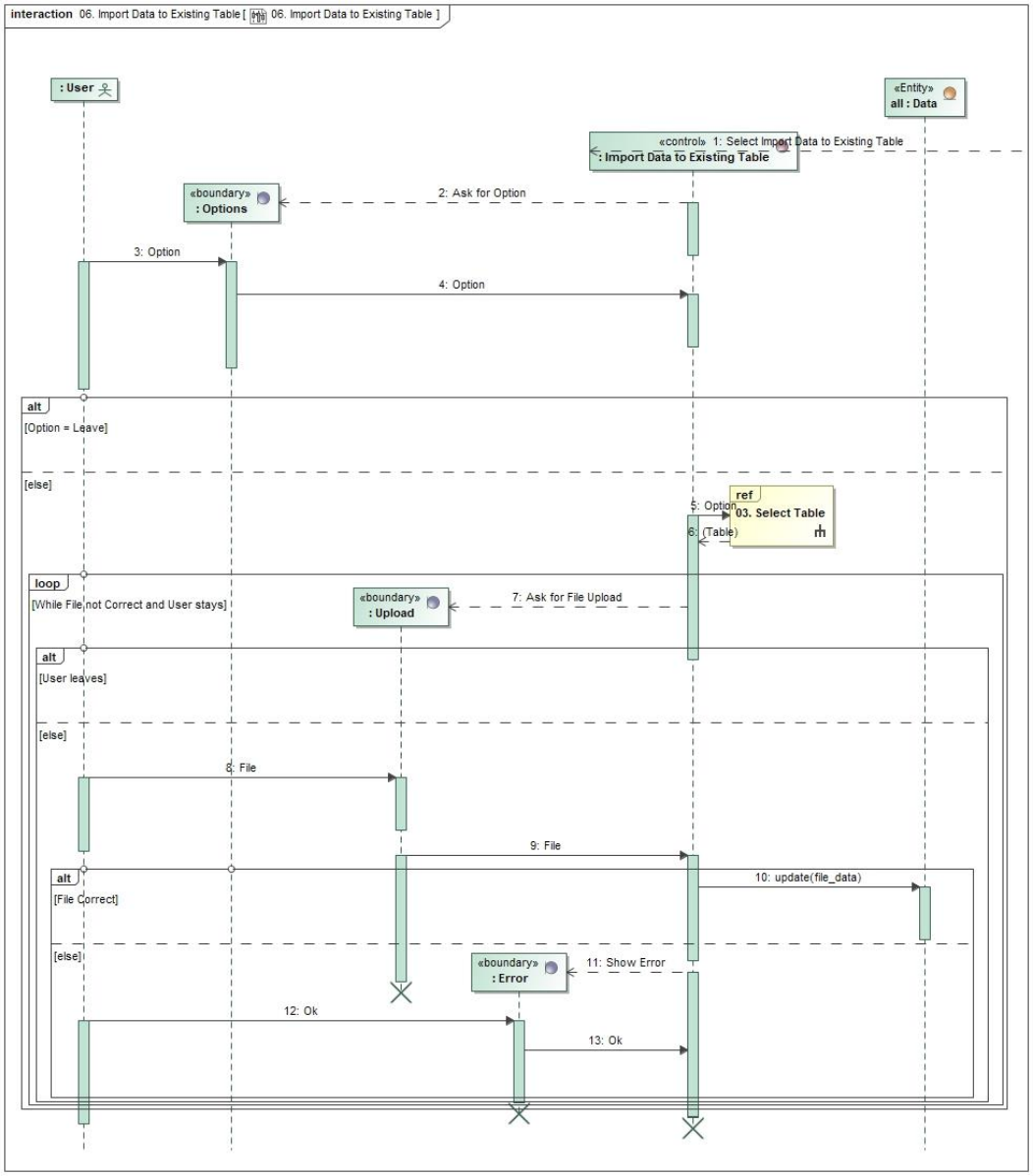


Figure 21. Sequence diagram UC 06. Import Data to Existing Table

UC 07. Convert Files

Activity diagram

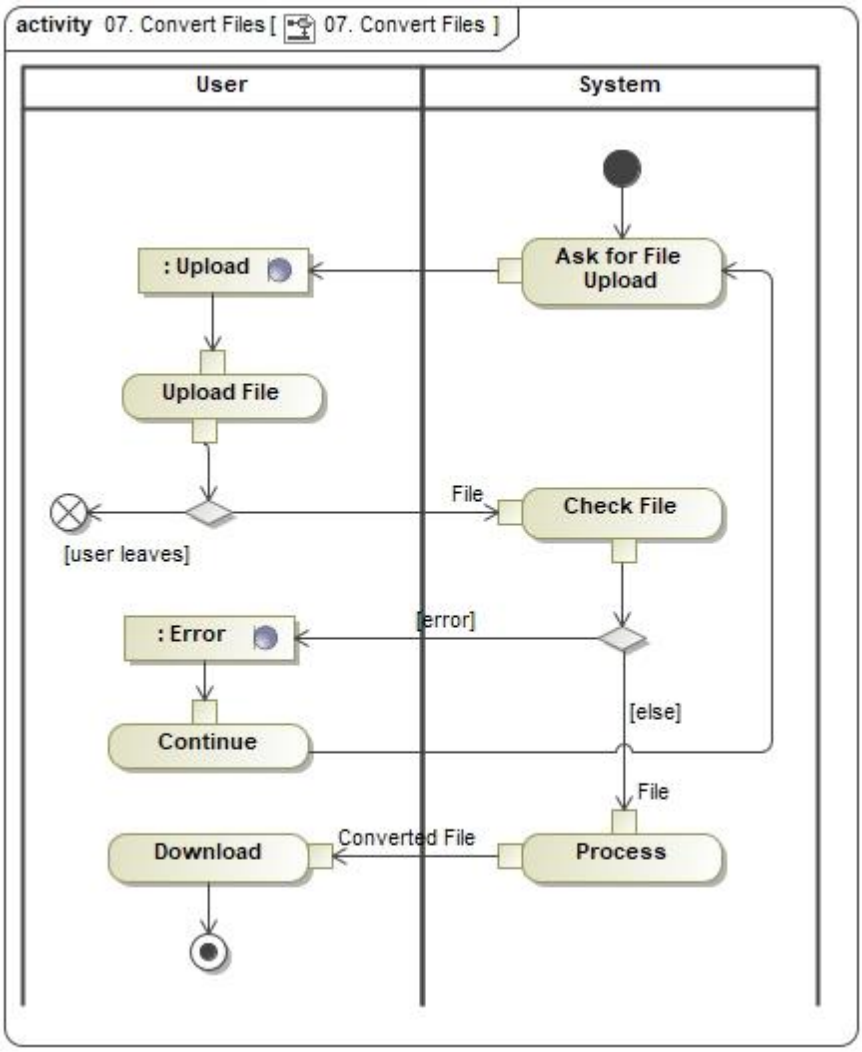


Figure 22. Activity diagram UC 07. Convert Files

Class diagram

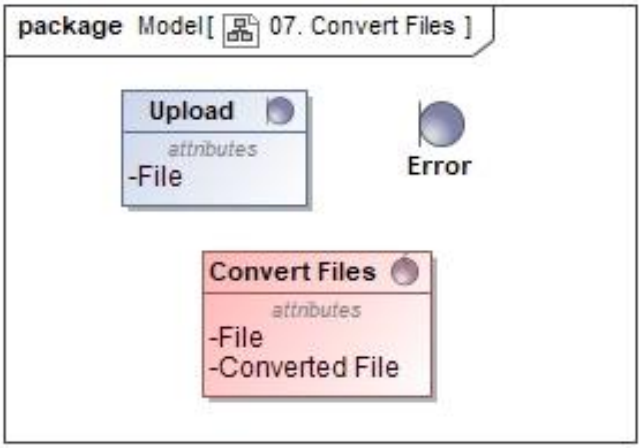


Figure 23. Class diagram UC 07. Convert Files

Sequence diagram

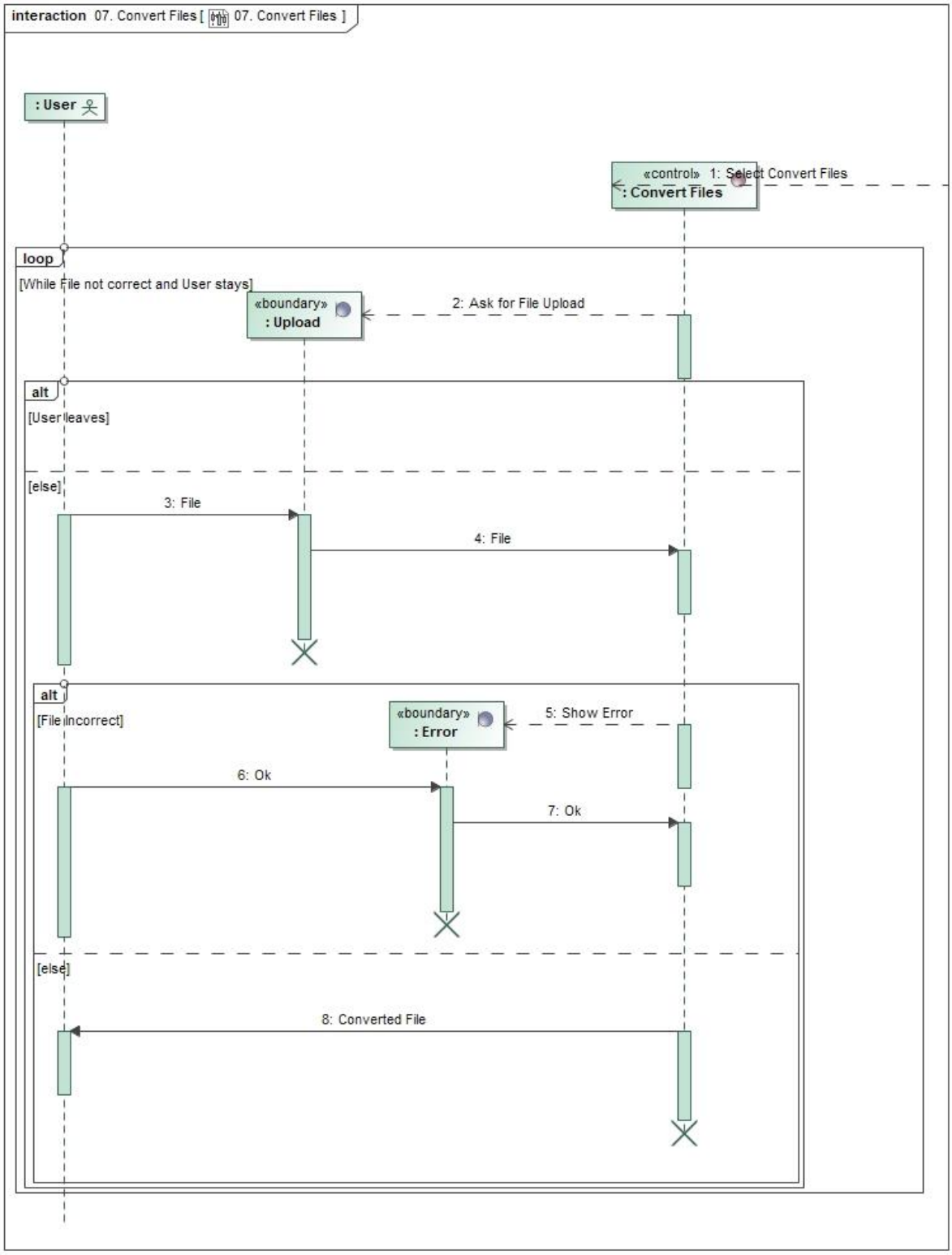


Figure 24. Sequence diagram UC 07. Convert Files

UC 08. Import Data to New Table

Activity diagram

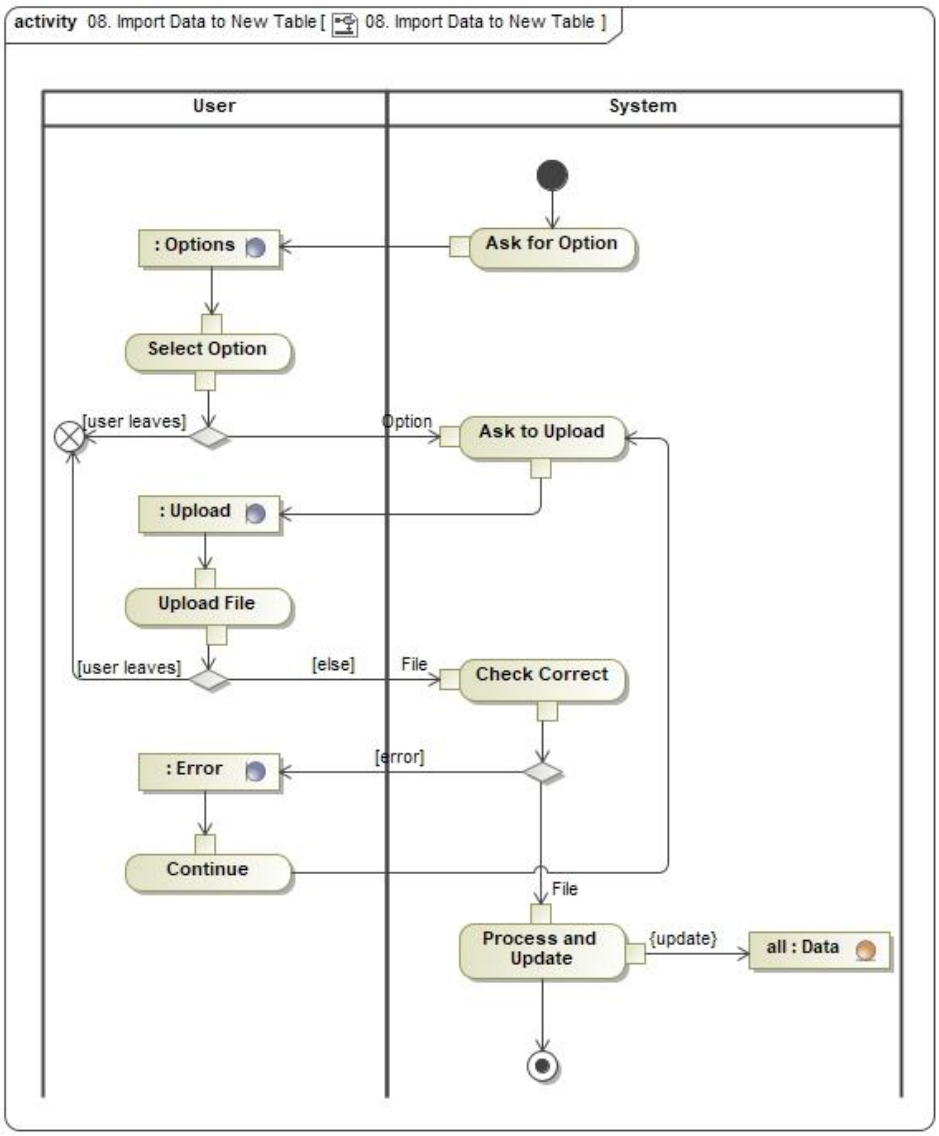


Figure 25. Activity diagram UC 08. Import Data to New Table

Class diagram

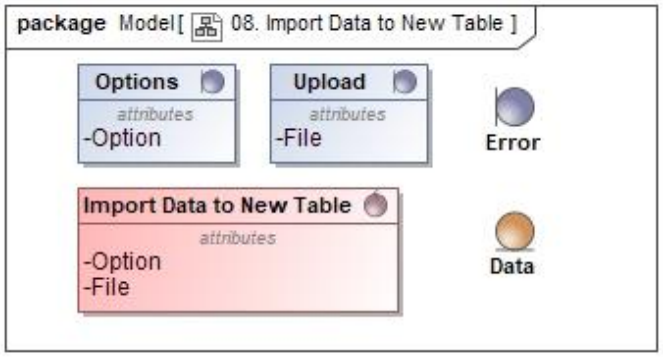


Figure 26. Class diagram UC 08. Import Data to New Table

Sequence diagram

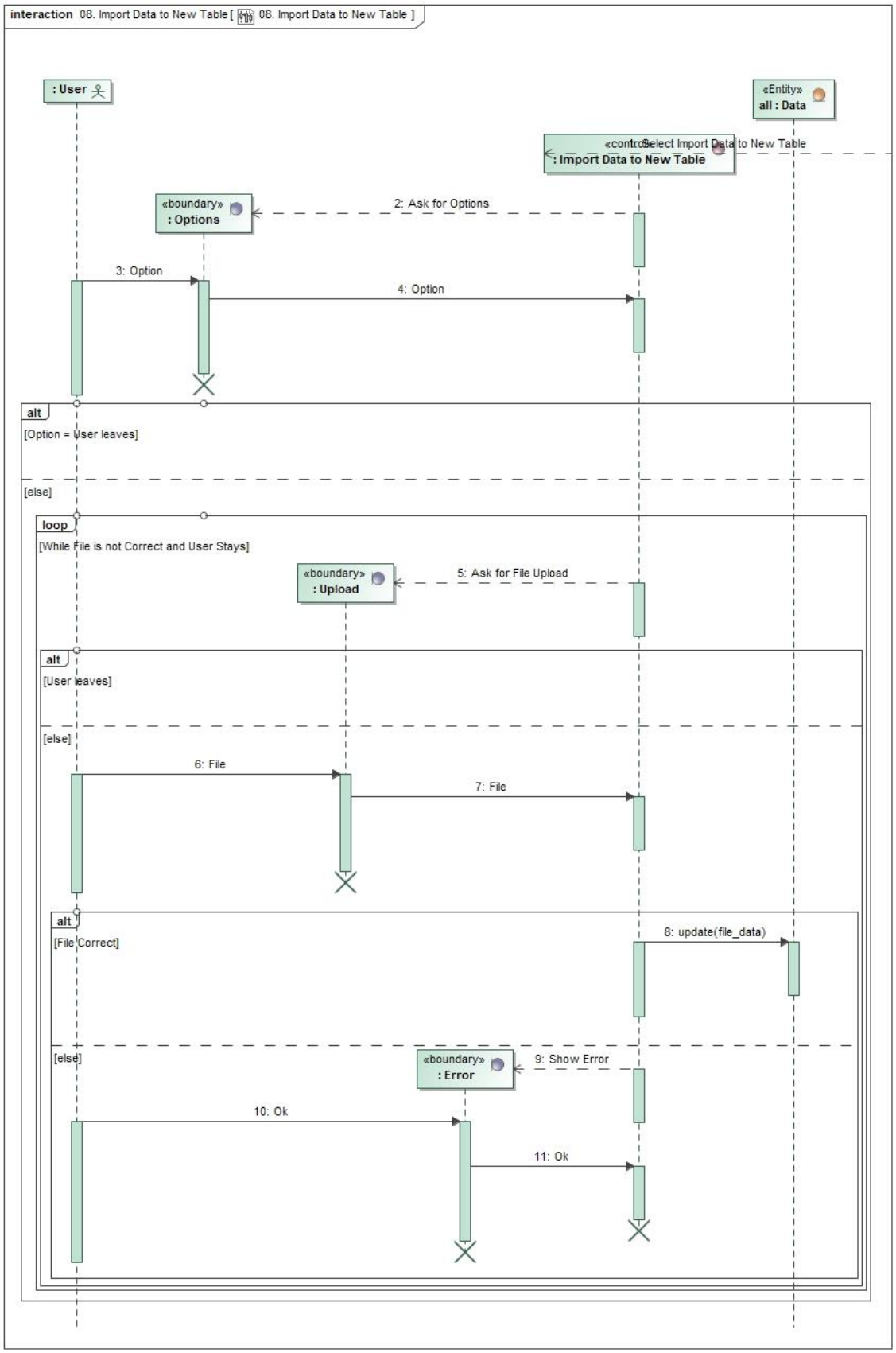


Figure 27. Sequence diagram UC 08. Import Data to New Table

UC 09. Visualize Table (CRUD)

Activity diagram

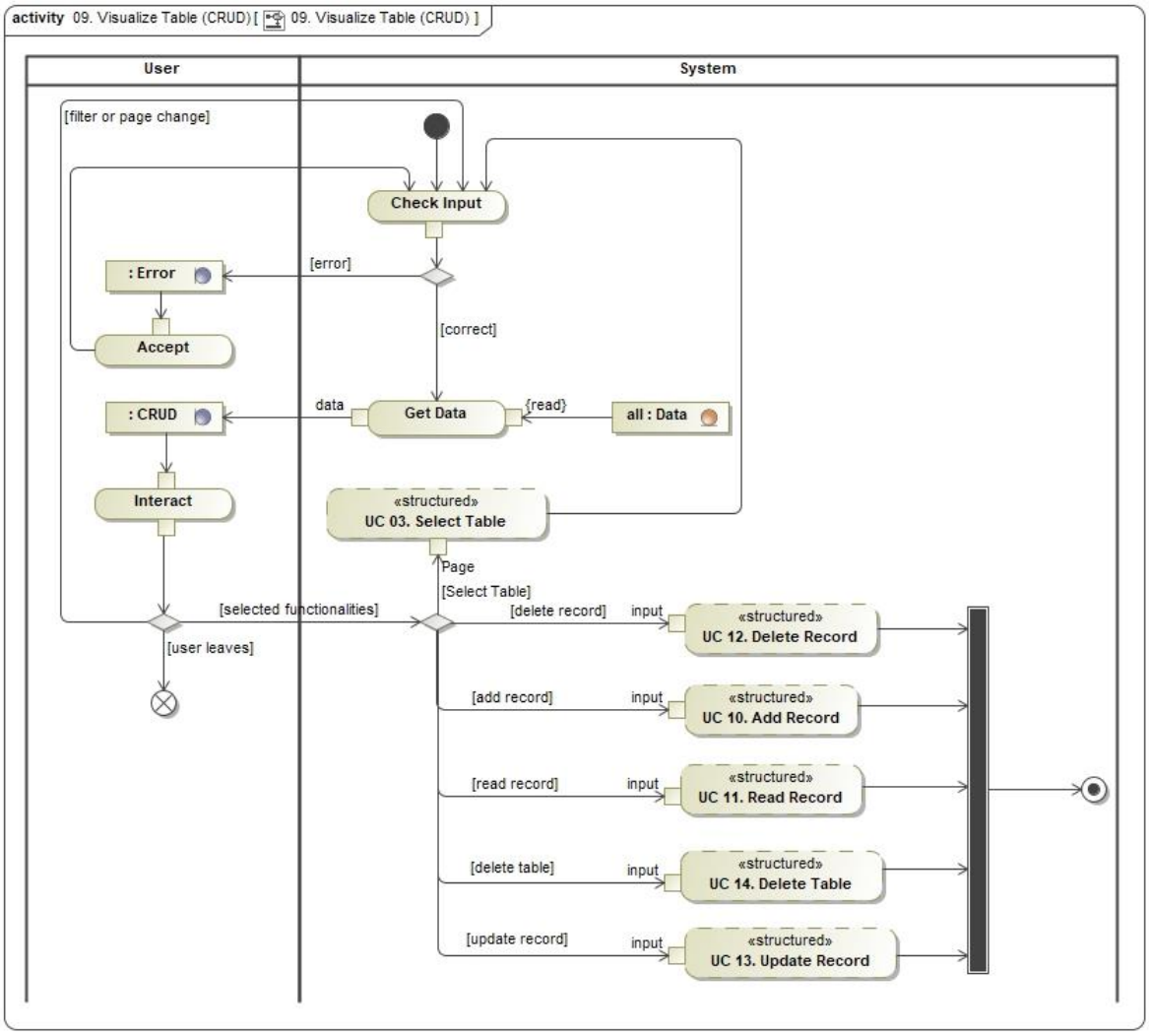


Figure 28. Activity diagram UC 09. Visualize Table (CRUD)

Class diagram

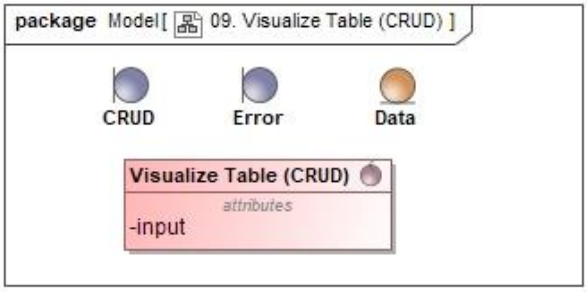


Figure 29. Class diagram UC 09. Visualize Table (CRUD)

Sequence diagram

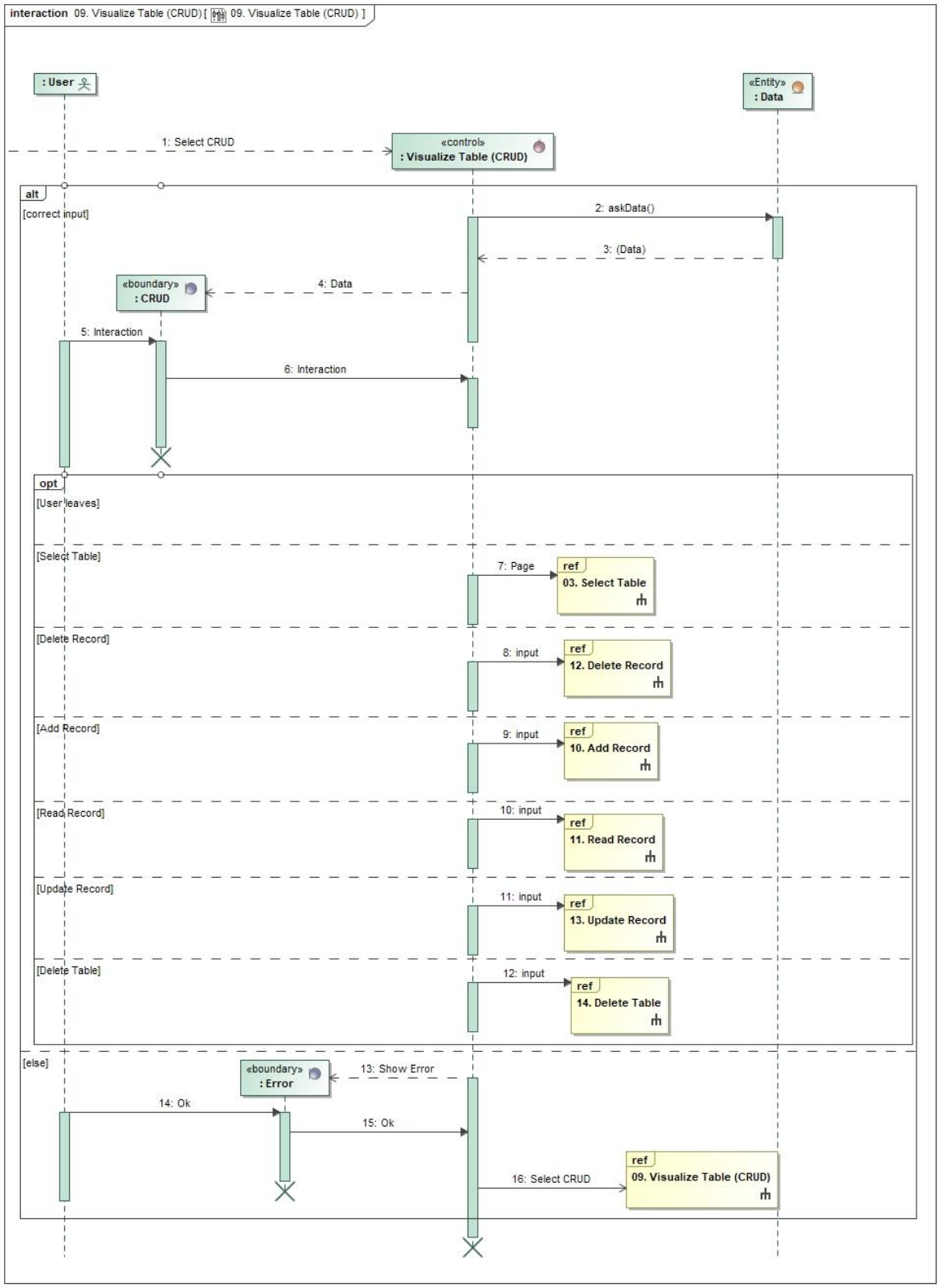


Figure 30. Sequence diagram 09. Visualize Table (CRUD)

UC 10. Add Record

Activity diagram

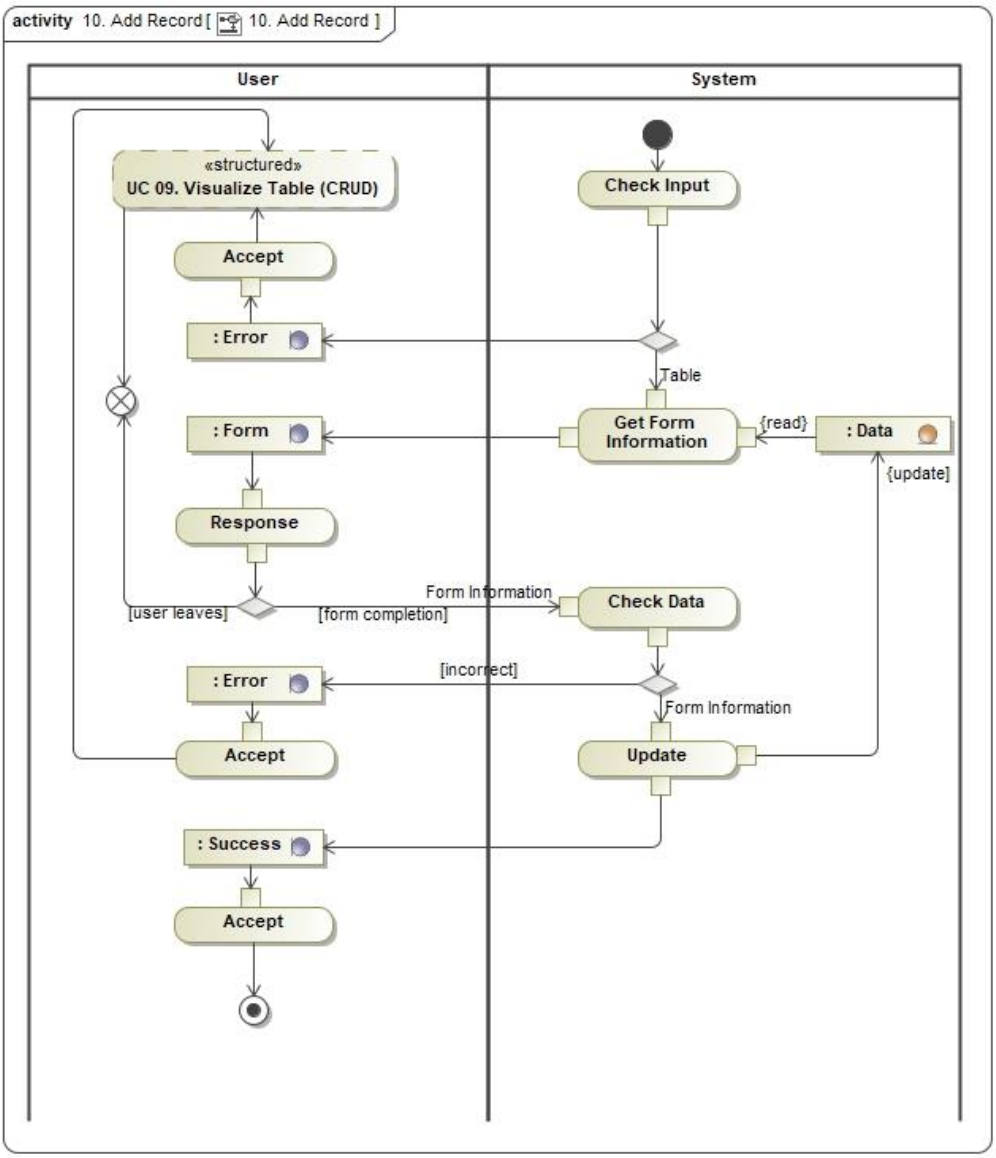


Figure 31. Activity diagram UC 10. Add Record

Class diagram

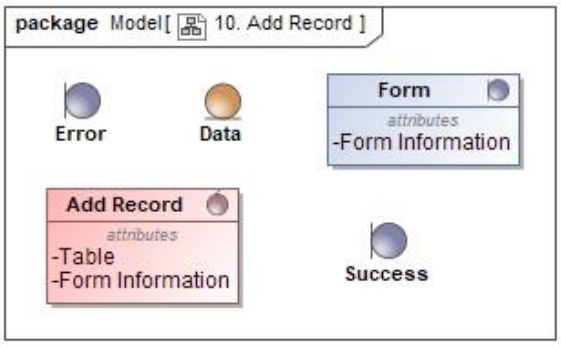


Figure 32. Class diagram UC 10. Add Record

Sequence diagram

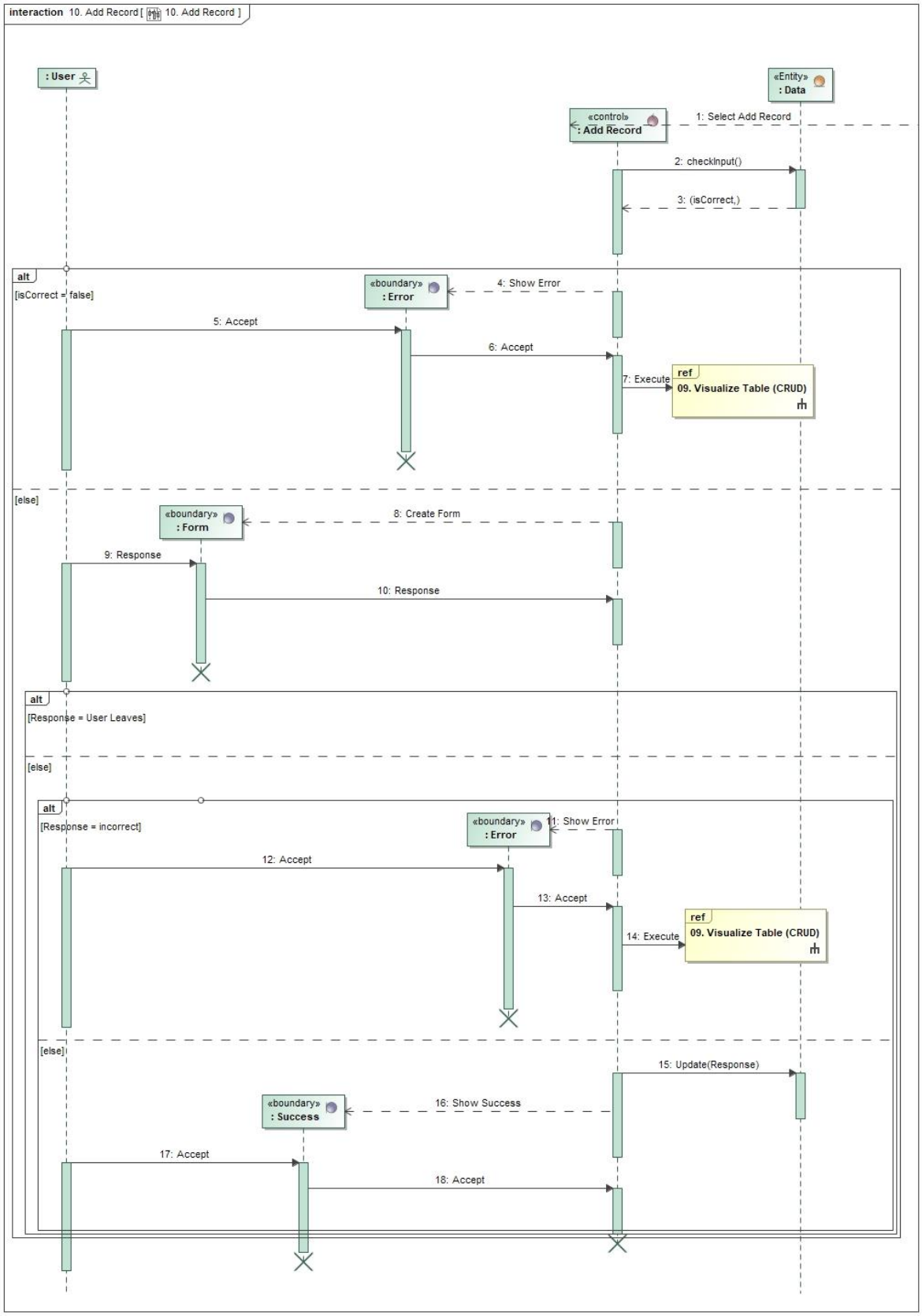


Figure 33. Sequence diagram UC 10. Add Record

UC 11. Read Record

Activity diagram

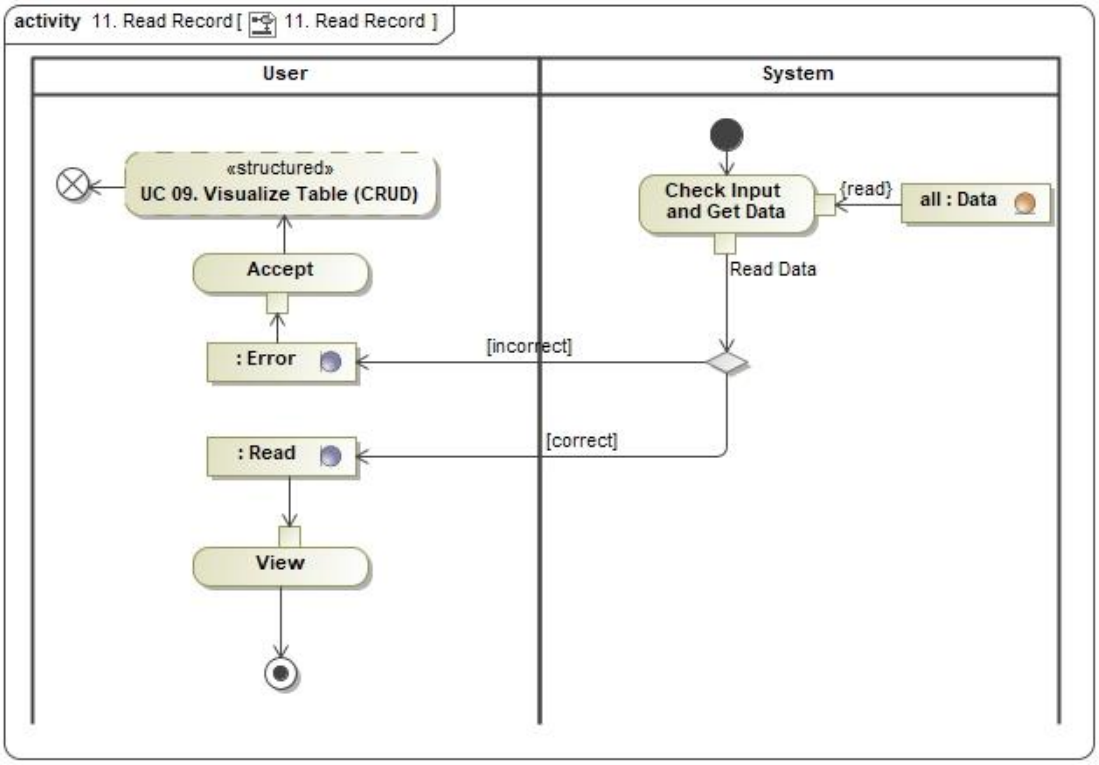


Figure 34. Activity diagram UC 11. Read Record

Class diagram

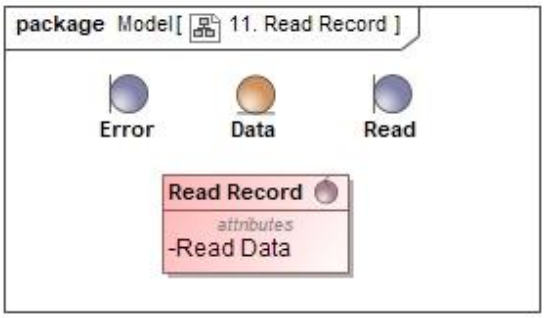


Figure 35. Class diagram UC 11. Read Record

Sequence diagram

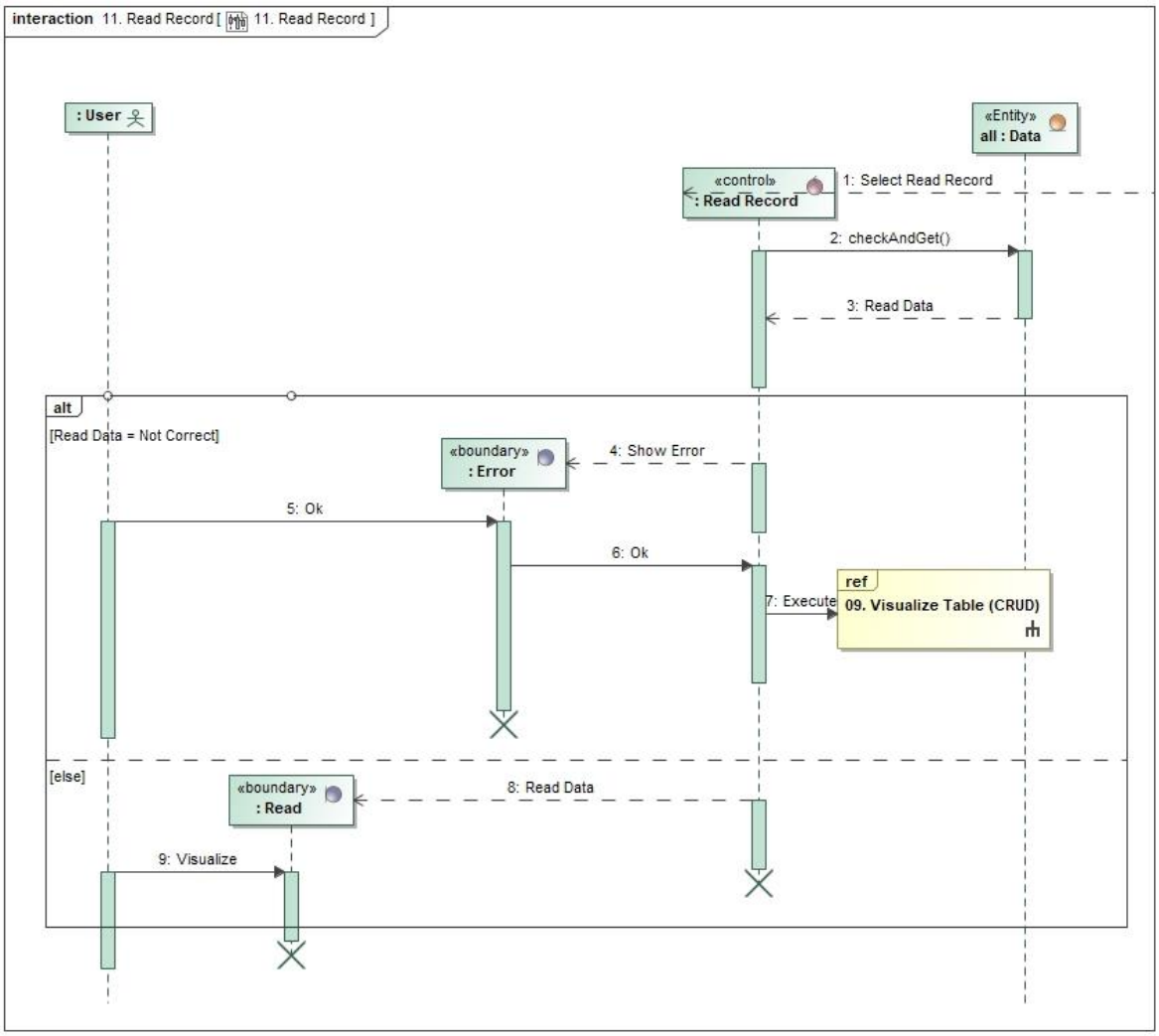


Figure 36. Sequence diagram UC 11. Read Record

UC 12. Delete Record

Activity diagram

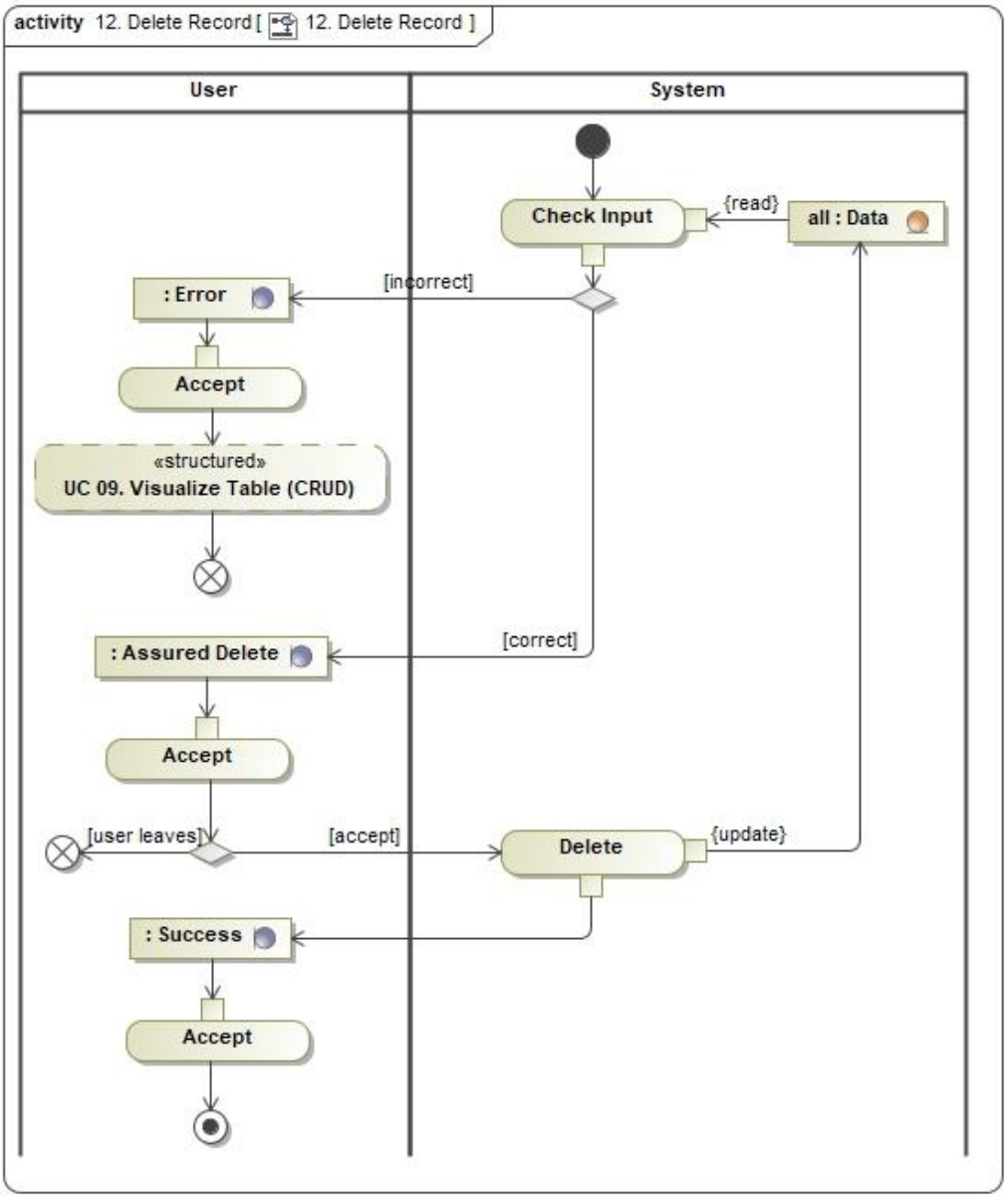


Figure 37. Activity diagram UC 12. Delete Record

Class diagram

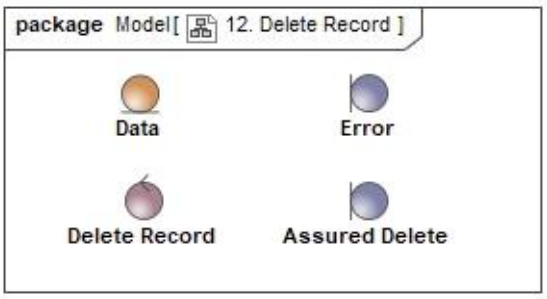


Figure 38. Class diagram UC 12. Delete Record

Sequence diagram

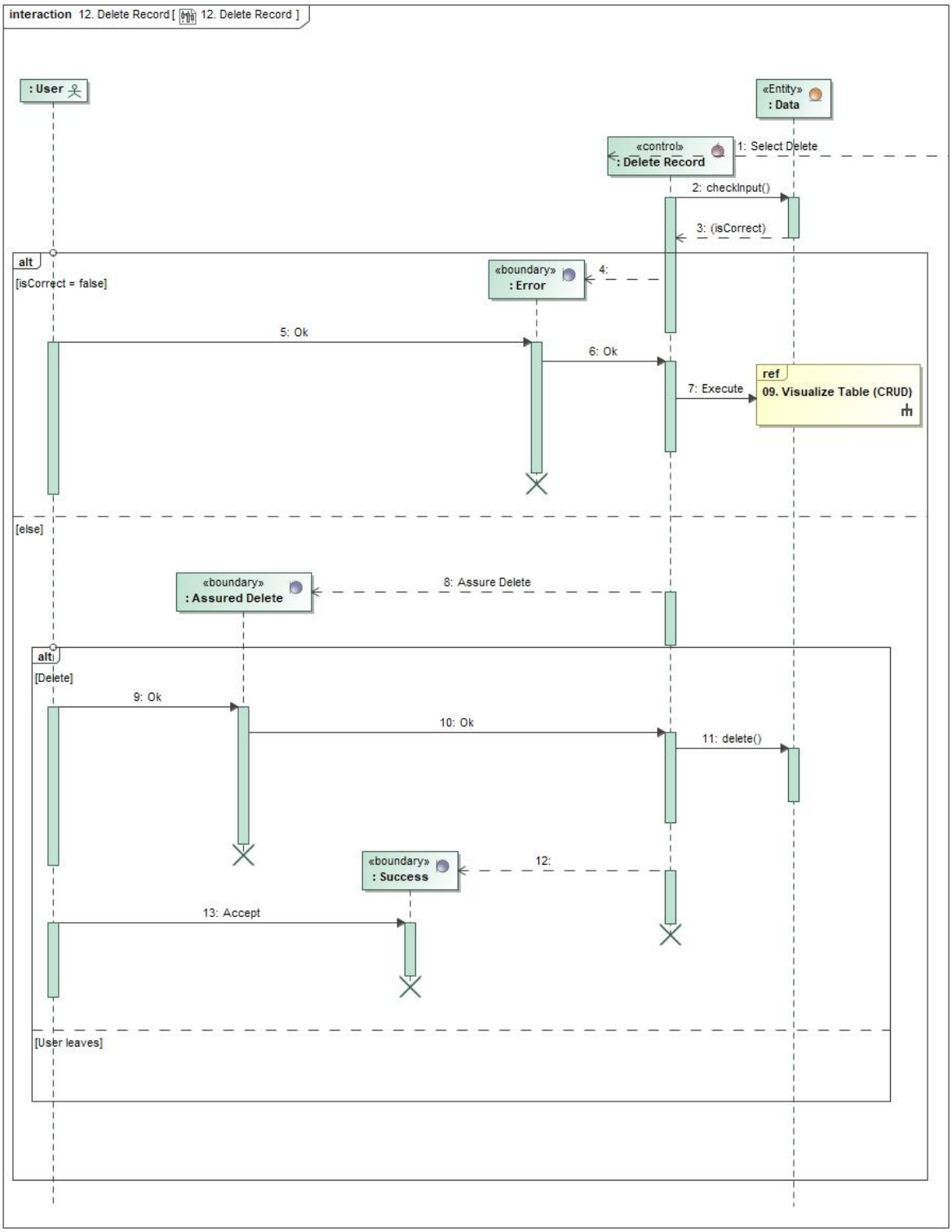


Figure 39. Sequence diagram UC 12. Delete Record

UC 13. Update Record

Activity diagram

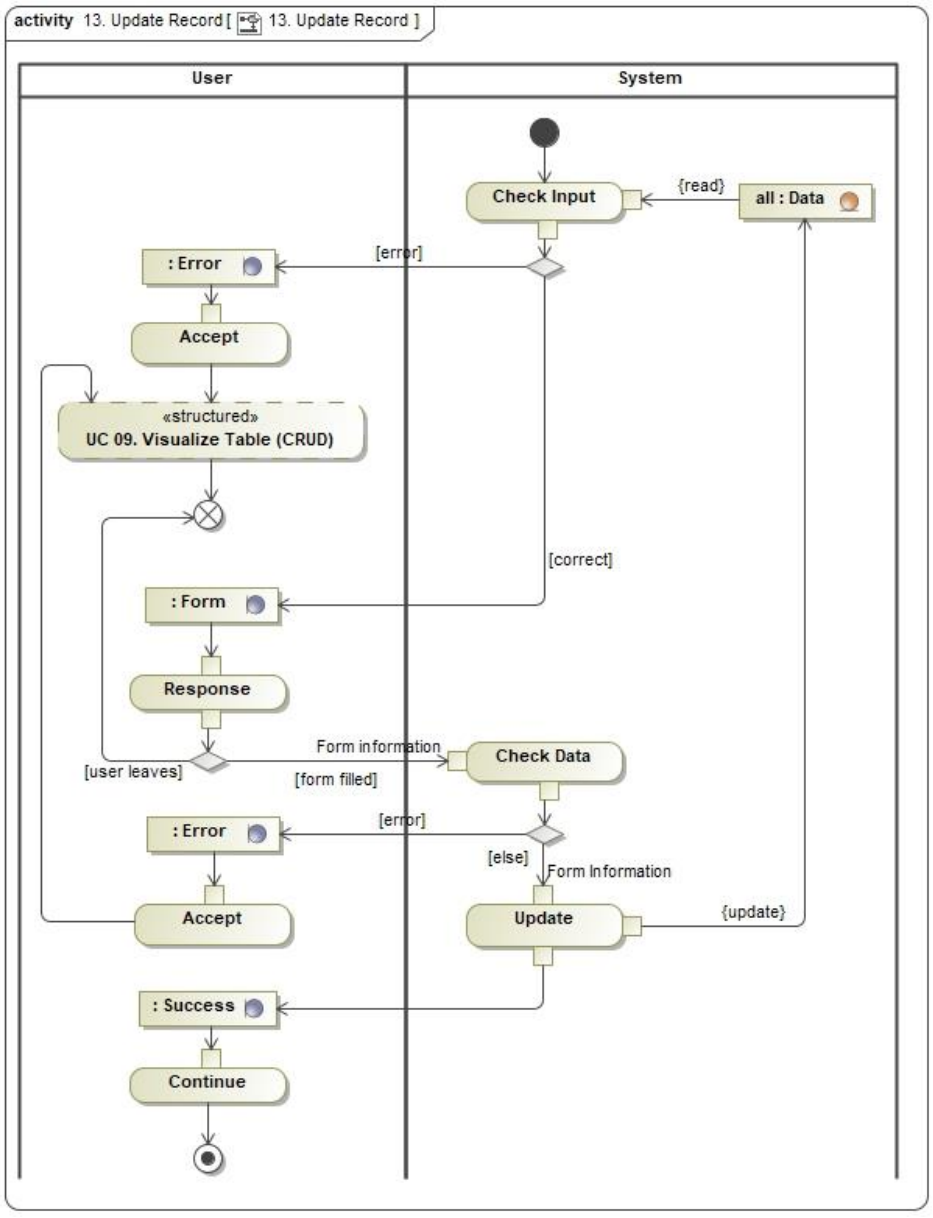


Figure 40. Activity diagram UC 13. Update Record

Class diagram

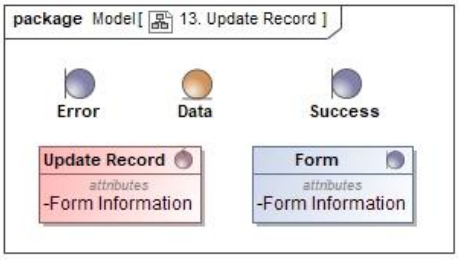


Figure 41. Class diagram UC 13. Update Record

Sequence diagram

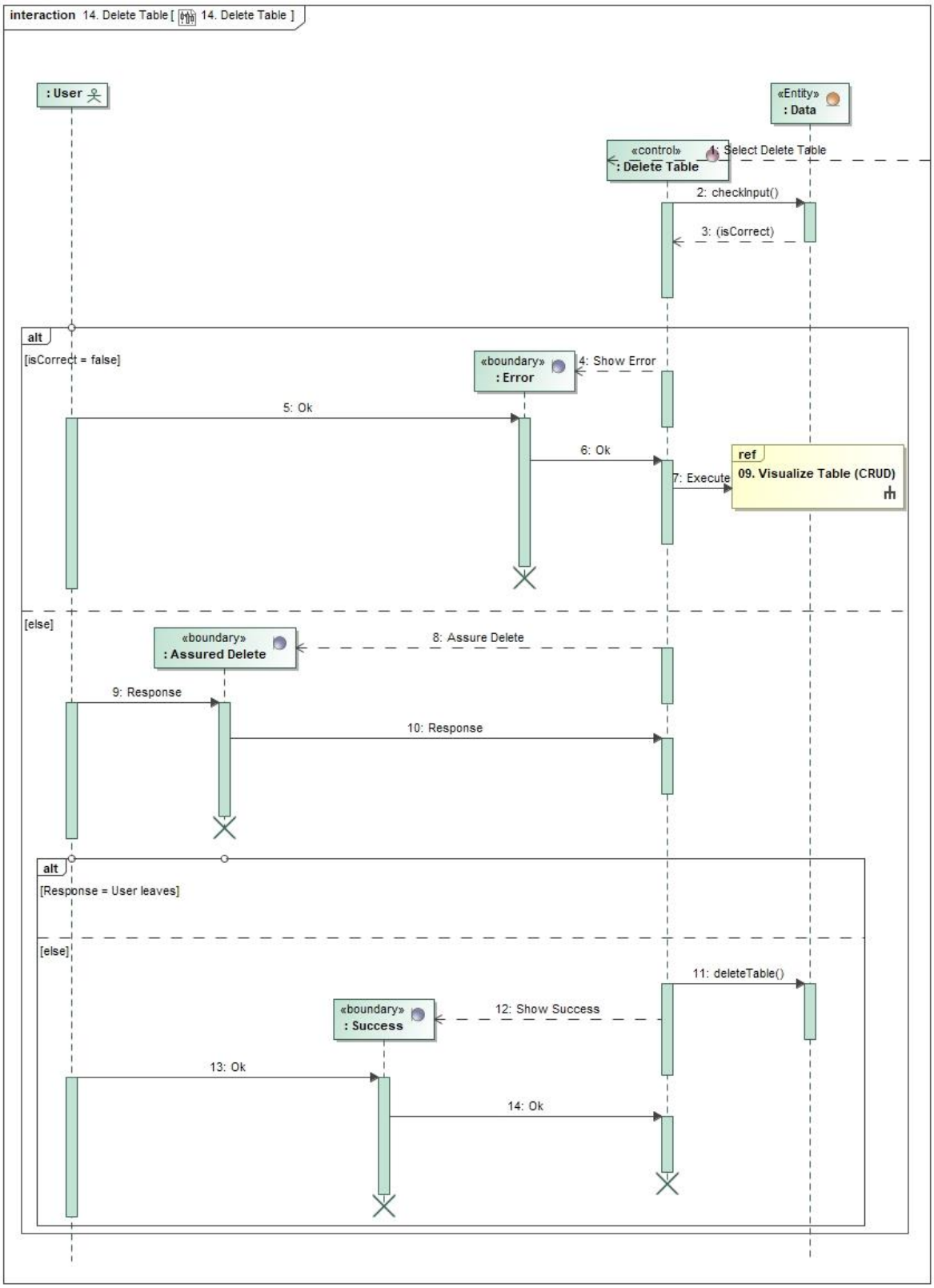


Figure 42. Sequence diagram UC 13. Update Record

UC 14. Delete Table

Activity diagram

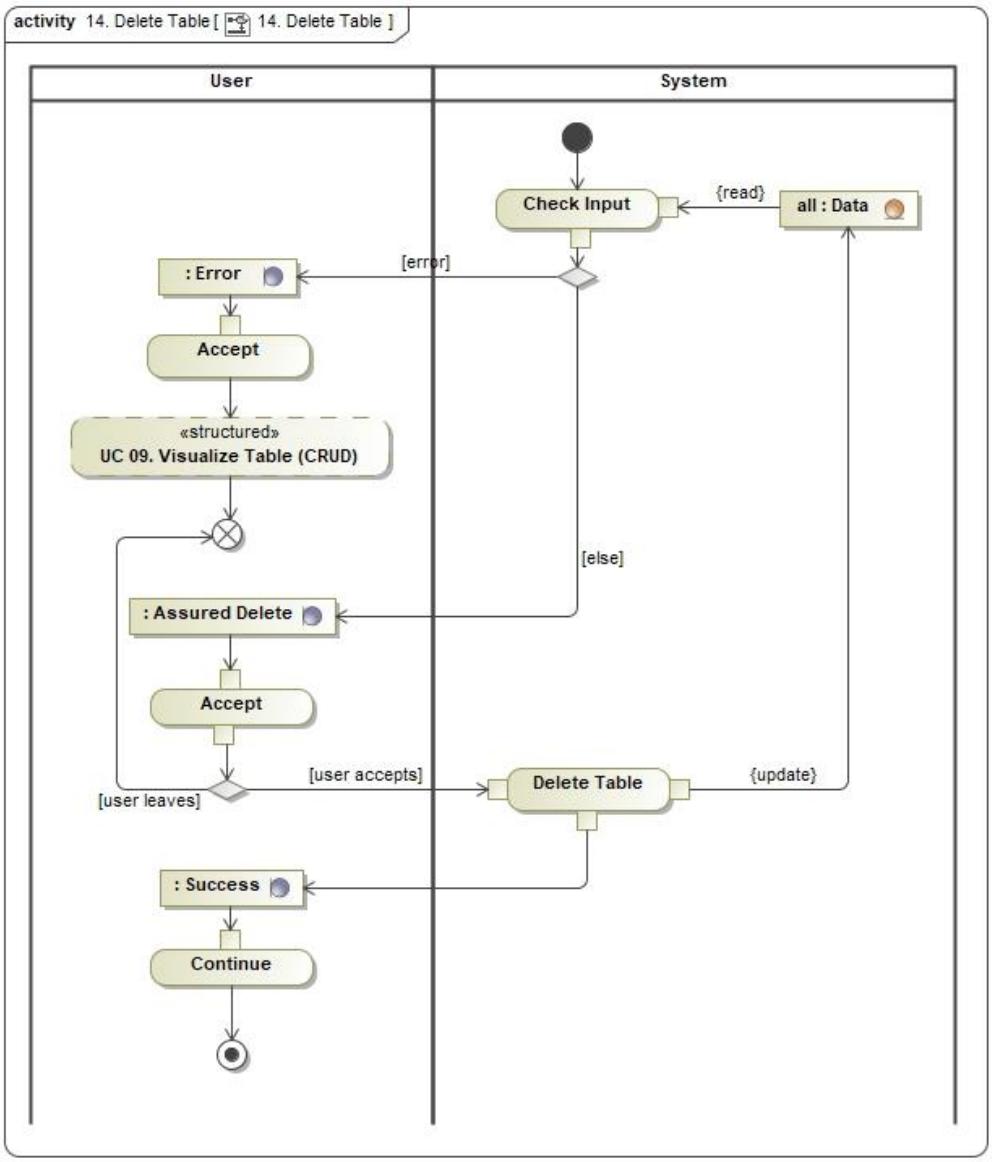


Figure 43. Activity diagram UC 14. Delete Table

Class diagram

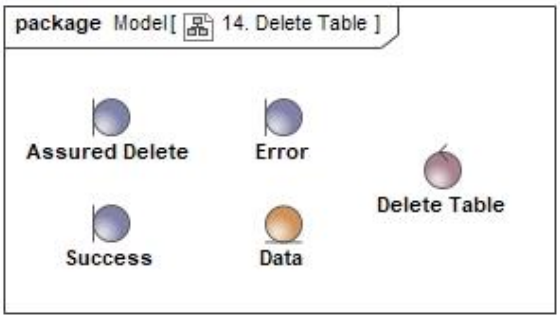


Figure 44. Class diagram UC 14. Delete Table

Sequence diagram

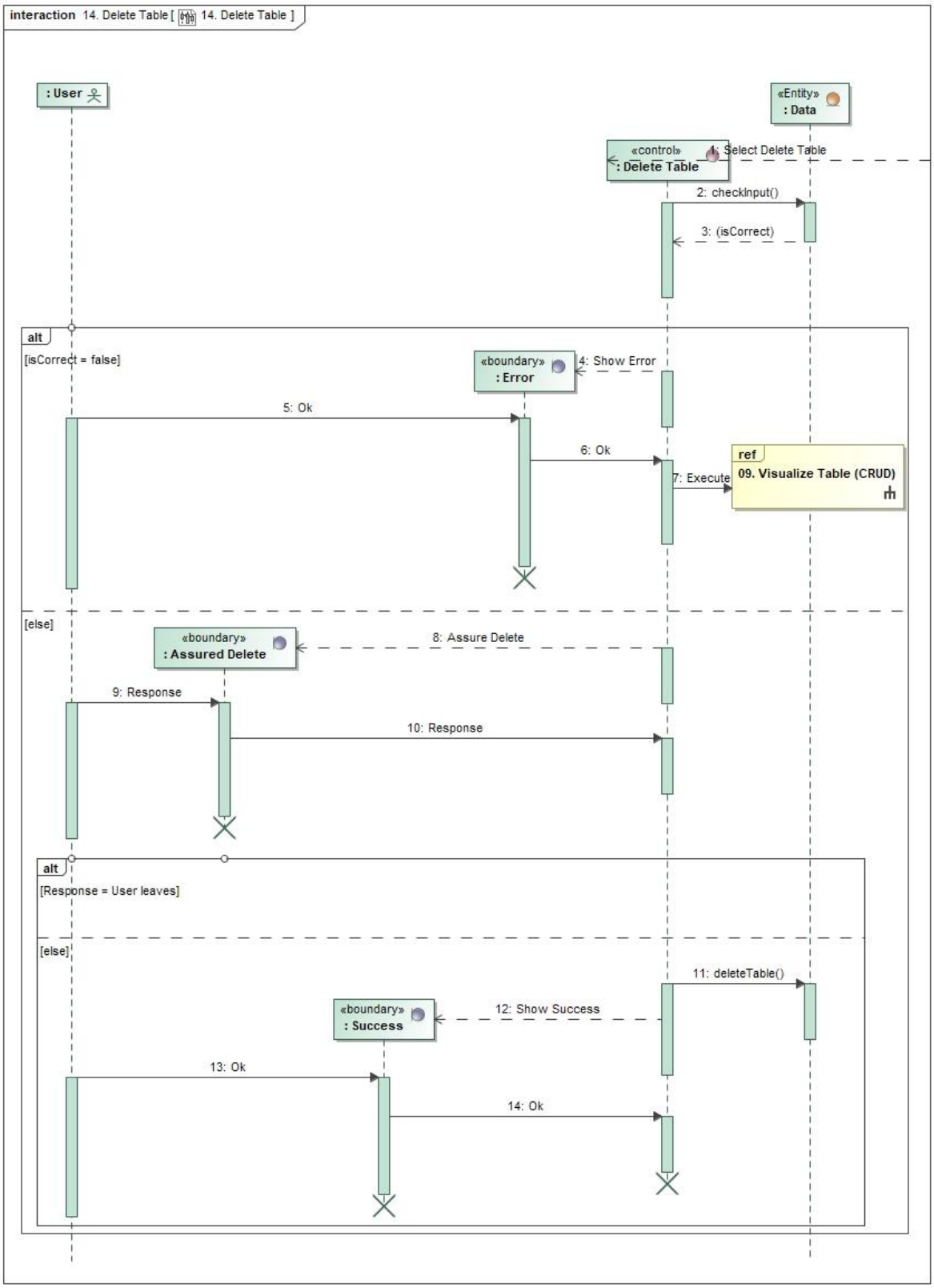


Figure 45. Sequence diagram UC 14. Delete Table

7 Design

In this section, we will provide details on various design elements of the application. These details will cover important information on the application's architecture, directory structure, user interface, and database.

7.1 Application Architecture

Our project has established a robust and scalable application structure to handle the extensive use of databases. The web application follows a server-side rendering approach using PHP to generate HTML content dynamically. This content is based on data retrieved from a MariaDB database.

For efficient handling and interaction with data, every functionality within the application utilizes dedicated PHP Data Objects (PDO) that represent the database. These objects work as an abstraction layer that encapsulates the logic of connecting to the database and executing queries. These database objects establish secure and efficient communication with the MariaDB database.

By utilizing database objects, every functional module can retrieve and manipulate the necessary data from the database. This approach ensures the separation of concerns and allows each functionality to have dedicated database-related operations. It also promotes code reusability and maintainability.

Once the functionality retrieves the data, the same PHP code that interacted with the database will dynamically generate the HTML code based on the information obtained. This dynamic generation allows for the scalability of functionalities with the expansion of the database.

7.2 Application Structure

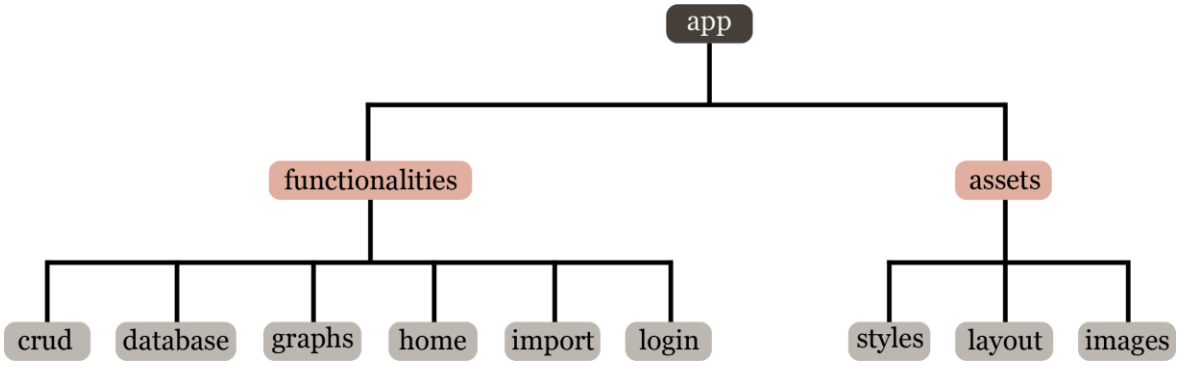


Figure 46. Directory structure of the web application

We structured the web application with a clear division into two main directories: "functionalities" and "assets." This organizational approach promotes modularity within the application. The "functionalities" directory houses individual modules containing the code to manage the specific tasks related to its designated functionality. On the other hand, the "assets" directory serves as a repository for static files required by the application, including images, stylesheets, and layout files. By separating functionality-specific code and static resources, the application achieves a well-organized structure that enhances maintainability, code reusability, and the overall scalability of the system.

7.3 User Interface Design

The graphical interface design in our web application is guided by fundamental principles, emphasizing simplicity and intuitiveness. We aimed to create an interface that allows users to easily navigate and utilize the various functionalities without feeling overwhelmed or confused. Recognizing that our users, predominantly non-tech-savvy, may need more experience with complex web applications, we placed great importance on making the interface as straightforward as possible.

By leveraging design patterns and familiar user interface conventions, we aimed to provide a user experience that feels natural and familiar, akin to the tools they use daily. Additionally, we ensured consistency throughout the interface by employing a general CSS file that establishes a unified visual style and layout for all HTML elements. This approach promotes a cohesive and professional look and allows for easy modifications and updates across the entire application.

7.3.1 Layout and Navigation

The layout of our web application consists of three key elements consistent throughout the page: the header, footer, and tab. These elements provide a cohesive structure to the interface.

The header acts as the primary navigation bar, allowing users to navigate between different application functionalities, with the logo acting as a home button. It is displayed at the top of the page, ensuring easy access to other sections at any point.

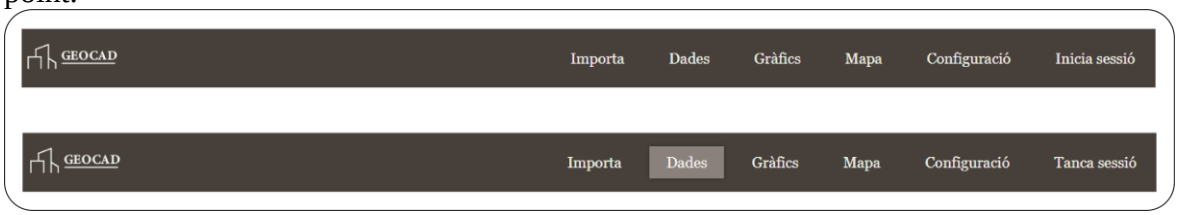


Figure 47. Header as the navigation bar

The bar highlights the current section with multiple visual cues like color and shadows to inform users of their location in the web application. Additionally, the functionality of the login and logout options dynamically adjusts depending on whether the user logged in or not.

The footer and tab elements are style elements but do not contain essential information. The footer element is at the bottom of each page, maintaining consistency in style with the header.

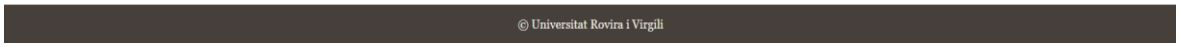


Figure 48. Footer element

The tab element in our web application is separate from the navigation tabs within the application itself. Instead, it refers to the browser tab, which represents the web page in the browser window. The tab element is located at the top of the browser window and displays the application logo and name.

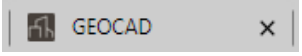


Figure 49. Tab element

7.3.2 Visual Elements

Our web application utilizes various visual elements to create an engaging interface. These elements include colors, typography, and imagery, which collectively contribute to the overall aesthetic and usability of the application.

Regarding typography, we employ a combination of Georgia and Arial fonts. We use Georgia primarily for text in titles or sections that do not contain numbers, as it offers an elegant and distinctive look. On the other hand, we employ Arial for its enhanced readability, particularly in contexts where numbers are present, like in the CRUD functionality. This careful selection of fonts ensures the text is visually appealing and easily legible, enhancing the overall user experience.

Concerning imagery, our application incorporates a logo as the primary iconographic element. The logo represents the brand identity and serves as a visual identifier for the application. We enhance the overall visual appeal by utilizing a recognizable and distinctive logo.

The color scheme employed in our application follows a carefully curated palette that balances professionalism and visual appeal. The selected colors are neither too bright nor dark, ensuring optimal readability and creating a pleasant visual environment for users.

7.3.3 Responsiveness

In our web application, we have worked on responsiveness to ensure that the interface adapts to various screen sizes and devices. The navigation bar is designed to be flexible to size changes. As the screen size changes or when viewed on smaller devices, the navigation bar dynamically adjusts to provide optimal usability, ensuring it always remain accessible and usable.

Figure 50. Header responsiveness

7.3.4 Feedback and Interactivity

Our web application incorporates various interactive elements, such as buttons, forms, and validation messages, to provide users feedback and enhance the interface's overall usability.

We have implemented dynamic button behaviors to enhance user engagement and provide visual cues. When users hover buttons, these undergo visual changes to indicate interactivity and responsiveness, including color changes, shading effects, or animations, depending on the button's purpose.

Figure 51. Button interactivity (Normal – Hovered)

In our web application, form fields also incorporate interactive features. When a user selects a form field, such as clicking or focusing on it, it changes visually to indicate the selection; this may include highlighting the field or adding a border to differentiate it from other fields.

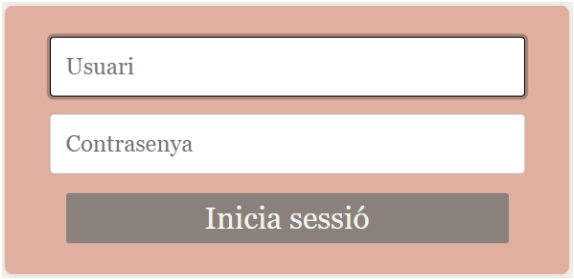


Figure 52. Form interactivity

As mentioned earlier, the header navigation bar contains buttons corresponding to different sections of the application. We have implemented highlighting the active section's button to provide clear visual feedback and assist users in orienting themselves. When users are in a particular section, the corresponding button in the navigation bar changes its appearance.

7.3.5 Cohesive Look

To achieve a visually pleasing web application, we created cohesion by using the same colors and styles for each function's elements. This cohesion helps users easily remember the purpose of each element, resulting in a more intuitive, user-friendly experience. Here we can see a table visualization via the CRUD system:

Figure 53. Interface of CRUD visualization

7.4 Database Design

In the Entity-Relationship Model, the database consists of three essential tables at the start of the application, each serving a specific purpose.

The first table is the "users" table, which stores user information for authentication purposes. The table has two columns, "user" and "password," both of type varchar(255) and marked as NOT NULL. The "user" column serves as the primary key for this table. This table does not have indexes, as we expect it to contain a manageable number of users. The passwords are stored securely using hashing techniques.

Next is the "hidden_tables" table, which lists table names that should not be accessible through the CRUD functionality. The table has a single column, "table_name," of type varchar(255), marked as the primary key. This table ensures that specific tables remain hidden from certain operations to maintain data integrity and security.

The main table, "estate," is optimized for efficient querying based on location proximity. It has several columns, including "id" (varchar(14)), "province" (varchar(30)), "size" (int(10)), "location" (point), and "size_class" (varchar(10)). The "id" column serves as the primary key. The "location" column is also marked as a unique key and has a spatial index for spatial queries. This design allows for quick retrieval of entries based on their geographic location using latitude and longitude coordinates.

In order to maintain data consistency and prevent errors, we implemented a cascading delete mechanism for the user-created tables linked to the main "estate" table. The mechanism used is the "ON DELETE CASCADE" statement.

Cascade is a mechanism present in mariaDB that causes the propagation of changes to child tables [4]. In our case, when a row is deleted from the main table, for example, using the CRUD functionality, all corresponding rows in the user-created tables referencing the deleted row will also be automatically deleted.

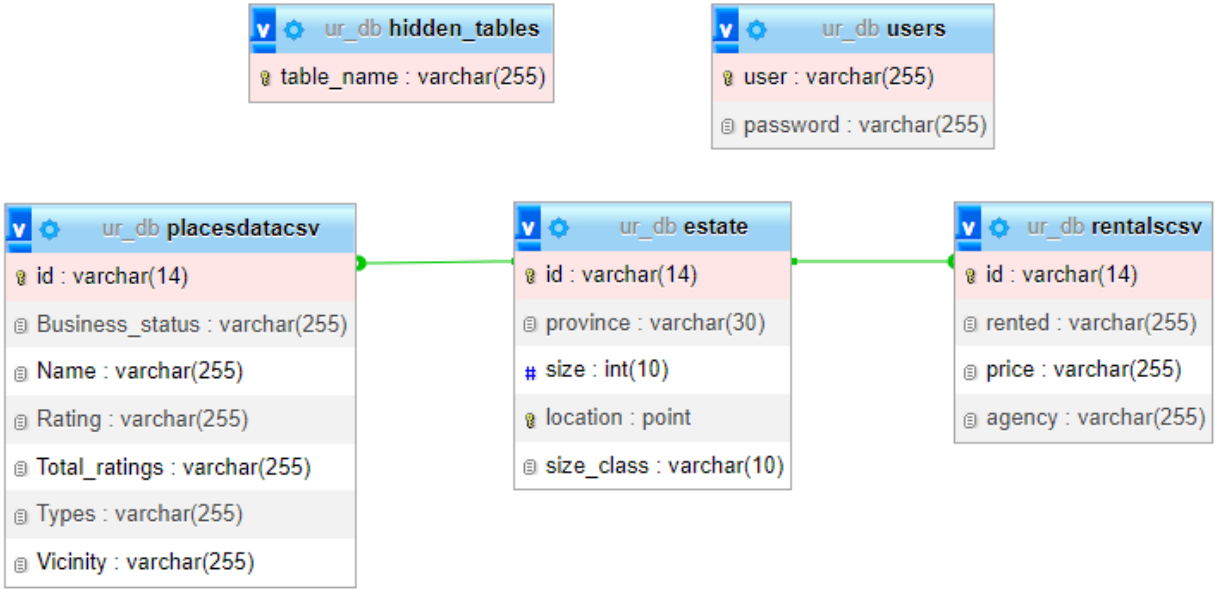


Figure 54. Database diagram with two user-created tables

7.4.1 Optimization

As we mentioned, the main table of the database (estate) is optimized for spatial queries. Classical database indexing structures are inappropriate for this kind of query as working with location requires range search instead of exact value matches like in hash tables. That is why we use a dynamic index structure called spatial index.

A spatial index is implemented as an R-tree index structure. The R-tree is a data structure that arranges objects in a tree-like manner akin to a family tree.

Each tree node represents a region of space and has a bounding rectangle that encloses all objects within it. The child nodes of a parent node have smaller bounding rectangles that fit inside the parent's bounding rectangle. The tree starts with a root node representing the entire indexed space, and at the leaf level of the tree are the actual objects, along with their associated information. The leaf nodes form the lowest level of the tree.

When inserting a new object into the R-tree, the algorithm places it in the node with the smallest increase in bounding rectangle area, ensuring that objects with overlapping or nearby locations are grouped together. When searching for an object in the R-tree, the algorithm starts from the root node and recursively traverses the tree based on the query criteria. Suppose the query is looking to find objects in a specified area. In that case, the algorithm will eliminate tree branches that do not intersect with that area by comparing the bounding rectangles.

The R-tree facilitates efficient spatial queries by organizing objects in this hierarchical structure. Instead of comparing every object in the collection, large portions of the search space can be eliminated. The algorithm will only consider nodes or objects intersecting with the search area or satisfying the query condition, reducing the time needed to find the desired objects [5].

7.4.1.1 Metrics

We performed a series of measures to analyze how the use of an R-tree structure affected the efficiency of spatial queries. We measured the execution time of fifty spatial queries looking for the closest point in at least ten meters from a random latitude and longitude for different data sizes.

	Time (s)	
Size (rows)	Expected	Correct
100	0,019	0,008
697	0,009	0,030
3028	0,007	0,080
11538	0,014	0,400
21131	0,008	0,725
35464	0,008	1,172
51705	0,010	1,724

Table 1. Time comparison between optimized and unoptimized spatial search

As we can see, the time increase is noticeable when using a non-optimized search algorithm to find the closest point to another. However, in the case of using an optimized R-tree, we cannot see significant growth even after taking the row number of the data set to 51705.

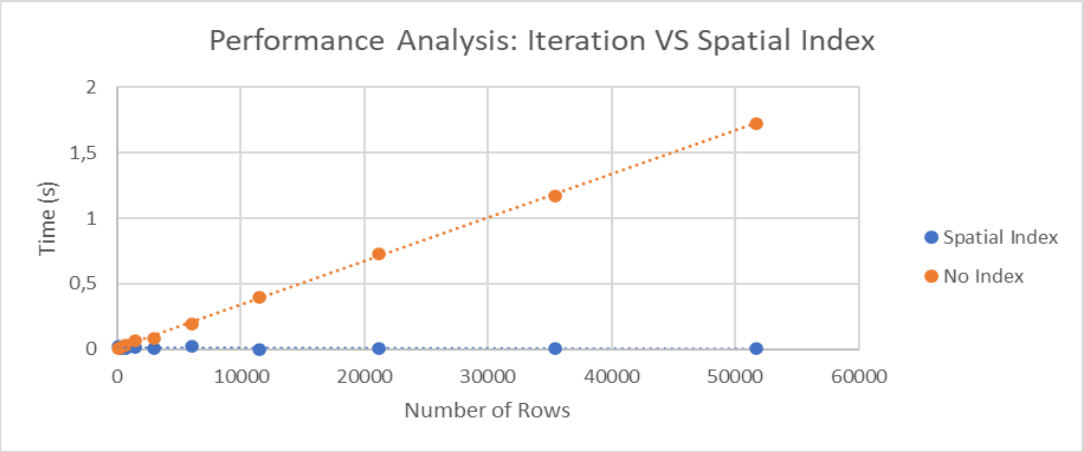


Figure 55. Graph representing execution time growth with data increase

8 Implementation

In this section, we will delve into the implementation choices made during the development of the web application, providing insights into the system's inner workings. We will start discussing our choice of technologies and then highlight key points of the code developed for the application.

8.1 Selection of Technologies

For several reasons, HTML, PHP, and CSS were chosen as the primary languages for developing the web application. One of the critical factors is the flexibility that coding directly in these languages provides. By doing so, we had complete control over the web application's structure, design, and functionality. This level of control allowed us to tailor the code to meet our specific requirements without being limited by the conventions and constraints of a framework.

Besides, coding directly in HTML, PHP, and CSS offered performance advantages. Frameworks often come with bundled features and libraries, which can increase an application's overall size and complexity. By coding directly, we could keep the codebase lightweight, resulting in faster load times and improved performance.

Another reason for this decision was the opportunity to deepen our understanding of the underlying technologies. Working directly with these languages gave us a comprehensive knowledge of web development fundamentals. This hands-on experience enhanced our skills and expertise in the field.

8.2 Implementation of Functionalities

In this section, we will focus on the critical components of our code that support the features we have built. Specifically, we will discuss how the code enables database access, login functionality, visualizing and managing data with CRUD operations, and importing new data.

8.2.1 Database Access

The database functionality in this web application plays a crucial role due to its intensive use throughout the system. A dedicated functionality was implemented to ensure a streamlined and reusable approach to accessing the database.

At the core of this functionality is a static class called "Database", which provides methods to establish and manage the database connection. The key method, connect(), is responsible for returning an object that serves as the connection to the database. This object can then perform the execution of the queries and interact with the database.

Here is an example of how to use the Database class:

```
$pdo = Database::connect();
$query = "SELECT * FROM users";
$result = $pdo->query($query);
```

Code 1. Usage of Database class

By encapsulating the database connection logic within the Database class, we ensure the connection is established consistently across the application. This class follows the singleton design pattern, allowing only one instance of the PDO connection to exist at a time.

Opting for a single connection provides several benefits, including efficient resource management and improved performance. It eliminates the need to create multiple database objects and ensures that all application parts share the same connection, promoting data consistency.

Here is the code snippet that demonstrates how the connect() method works:

```
public static function connect()
{
   // If there is no connection, create one; else reuse
   if ( null == self::$cont )
   {
      try
      {
         self::$cont =  new PDO(
            "mysql:host=".self::$dbHost.";"
            ."dbname=".self::$dbName,
            self::$dbUsername,
            self::$dbUserPassword);
      }
      catch(PDOException $e)
      {
         die($e->getMessage());
      }
      return self::$cont;
}
```

Code 2. Database connect() code

8.2.2 Login

In order to provide secure access to the application's functionalities and restrict certain operations that can modify the data, we implemented a login functionality.

We implemented the login process using a standard password hashing technique supported by PHP. Specifically, we chose the password_verify() function. Here is how the process works: when a user submits their credentials, the server verifies if the user exists and then compares the provided password against the stored hashed password.

Once we could check credentials, we implemented a mechanism to check if a user logged in and to make them have an automatic timeout. We implemented this mechanism using php built-in $_SESSION functionality, which includes session management and timeout features. By leveraging sessions, the application can keep track of the user's login status and maintain it until a logout action, or a predefined timeout period is reached.

Here is the code that compares the passwords and starts the user session:

```
if (password_verify($_POST['password'], $result['password'])) {
   $_SESSION['logged_in'] = true;
   $_SESSION['username'] = $_POST['username'];
   $_SESSION['timeout'] = time();
   header('location: '.$page);
}
```

Code 3. Login session start

As the code shows, we use session variables to manage the user's login status and enable specific functionalities. Concretely we observe these three setting variables:

1. $_SESSION['logged_in']: This variable determines whether the user logged in. It serves as a flag to indicate the user's authentication status within the application.
2. $_SESSION['username']: This variable stores the logged-in user's username. It provides a way to identify the user throughout their session and enables personalized functionalities or user-specific operations.
3. $_SESSION['timeout']: The timeout mechanism is implemented using the time() function, which returns the current timestamp. Setting $_SESSION['timeout'] = time(); causes the session's timeout to initialize or reset.

The timeout functionality operates as follows: with each interaction or page refresh, the session's timeout is refreshed to the current time. If the user remains inactive for a specified period, exceeding the defined timeout threshold, the session is terminated automatically, logging the user out and clearing the session data. Timeouts enhance security by closing idle sessions to prevent unauthorized access.

In order to validate the access to restricted functionalities for users that logged in, a modular and reusable approach was adopted. We created a PHP file, “logged_in.php”, that encapsulates the code for determining whether the user logged_in and sets a variable, $logged_in, accordingly.

By including this file in the relevant PHP files, the functionalities can check the value of the $logged_in variable to determine the user's authentication status. This approach eliminates the need to duplicate code across multiple files and ensures consistency in determining user access.

The following code snippet demonstrates how the authentication check occurs:

```
require_once("../login/logged_in.php");
if(!isset($logged_in) or !$logged_in)
{
    header('location: ../login/login.php?page=
       '.htmlspecialchars($_SERVER['PHP_SELF']));
}
```

Code 4. Login check

As we see from the code snippet, it performs the check to determine whether the user logged in. If the check results in a non-logged user, the code redirects them to the login page.

Here is the code of “logged_in.php” that the functionalities use:

```
session_start();
if(isset($_SESSION['logged_in']) && $_SESSION['logged_in'] == true){
   $logged_in = true;
}else{
   $logged_in = false;
}
```

Code 5. “logged_in.php” code

Finally, if the user decides to log out, the pertinent $_SESSION variables are unset, and therefore they lose access to restricted functionalities.

8.2.3 CRUD

Implementing a dynamic CRUD functionality within a web application posed significant challenges, requiring careful consideration. Two major hurdles emerged during the development process, each demanding a unique solution.

The first significant challenge stemmed from the dependency of these CRUD functionalities on user-provided data. Although we implemented measures to restrict user interaction with the page, malicious users could send unexpected data and access restricted tables. That made it crucial to implement robust data validation and verification mechanisms on the server side. The information sent by the user needed to be carefully validated to prevent data corruption, unauthorized access, or potential security vulnerabilities.

As most of the CRUD functionalities work with similar input, the solution code must be reusable. That is why we made multiple functions to check the input and used them in all the required functionalities. One example is the following function "is_valid_table()":

```
function is_valid_table($pdo, $table, &$table_info) {
   /* Check if table exists */
   $exists_query = 'DESCRIBE ' . $table;
   $table_info = $pdo->query($exists_query);
   if($table_info == false) return false;
   /* If table exists, check it is not hidden */
   $is_hidden_query = "SELECT *
      FROM hidden_tables WHERE table_name = ?";
   $stmt = $pdo->prepare($is_hidden_query);
   $stmt->execute([$table]);
   if ($stmt->rowCount() > 0) return false;
   return true;
}
```

Code 6. “is_valid_table()” code

As shown by the code, two checks are made in the function if the table exists in the database and is not restricted. If both conditions are met, then the table is valid and true will be returned.

Furthermore, when performing operations in concrete records, like update or delete, as each table has a different structure, the information of the name of the primary key column and the value of the record for that column are passed by the user. That is why we need to check whether the column is a primary key in the table and if the record exists.

To perform these validations, we created functions that check if a column is a primary key and generate a dynamic select query adapted to each table. Here is the code that performs the primary key validation:

```
function is_primary_key($columns, $col_name) {
   foreach($columns as $col) {
      if($col['name'] === $col_name) {
         if($col['key'] === 'PRI') return true;
         return false;
      }
   }
   Return false;
```

```
}
```

Code 7. “is_primary_key()” code

The second challenge arose from the requirement to create CRUD operations that could seamlessly adapt to any data stored in the database. Unlike traditional static CRUD implementations, this dynamic approach called for a flexible system capable of handling various data structures and adapting its functionality accordingly.

In order to successfully construct this adaptable function, we need to dynamically construct the queries to the tables. So, for that, the general process followed is to do a DESCRIBE query that will inform the server of the table structure and then construct the queries with the column information. For example, this query generation is present in the viewing functionality, where the adaptation is not only to the table that the user is viewing at the moment but also to the page and the filters for each column the user selects.

Here is the code that dynamically generates a select query given the names and types of the columns:

```
function get_select_query($columns, &$filt_cols, $table, $filter) {
   $select = ""; $filt_cols = array();
   foreach($columns as $col) {
      if(strpos($col['type'], 'point') === false){
         $select .= $col['name'] . ", ";
         $filt_cols = array_merge($filt_cols, array($col['name']));
      } else {
         $select .= "ST_X(" . $col['name'] . ") AS " .
            $col['name'] . "_lat, ST_Y(" . $col['name'] .
            ") AS " . $col['name'] . "_lon, ";
         $filt_cols = array_merge($filt_cols, array($col['name'] .
            "_lat", $col['name'] . "_lon")) ;
      }
    }
    $select = "SELECT " . substr($select, 0, -2) .
       " FROM " . $table . $filter;
    return $select;
}
```

Code 8. “get_select_query()” code

A similar process is followed to generate other queries, like update or filter.

8.2.4 Import Data

Even though there are no significant changes for the different options for importing data in the user view, the processing code for each file type differs significantly. Given these differences, we will delve into each of their implementations separately.

8.2.4.1 Importing data to the Main Table via CAT Files

CAT files organize data into multiple records, each occupying a single line and consisting of a fixed number of characters. However, what sets CAT files apart is that each record type uses different characters to store information.

One of the significant downsides of CAT files is their inherent inefficiency. Regardless of the data they need to store, each record occupies a fixed number of

characters. This design choice wastes space, as many characters within a record are left blank. Consequently, CAT files tend to occupy larger sizes than necessary, consuming more storage resources.

We studied their multiple record types and dispositions. When we determined what record types and sections were of interest, we just needed to process the file, iterating over each line, and extracting the characters of importance. Here is a snippet code of the function in charge of obtaining all the essential fields:

```
function generateInfo($geo, $chars, &$id, &$place, &$superficie,
   &$x_cord, &$y_cord, &$utm, &$lat, &$lon)
{
    $id = implode("", array_slice($chars,30,14));
    $place = implode("", array_Slice($chars,52,25));
    $superficie = intval(implode("", array_slice($chars,295,10)));
    $x_cord = floatval(implode("", array_slice($chars,333,7)) . "."
       . implode("", array_slice($chars,340,2)));
    $y_cord = floatval(implode("", array_slice($chars,342,8)) . "."
       . implode("", array_slice($chars,350,2)));
    $utm = intval(implode("", array_slice($chars,674,2)));
    $easting = $x_cord;
    $northing = $y_cord;
    $zone = "31T";
    $geo->setUTM($easting, $northing, $zone);
    $geo->convertTMtoLL();
    $lat = $geo->Lat();
    $lon = $geo->Long();
}
```

Code 9. “generateInfo()” code

As we see in the code, it receives the line and cuts the characters that contain the information. One of the interesting sections of this code is transforming geographic information from the UTM setting to latitude and longitude for better compatibility with other systems.

The second challenge was that, given the size of CAT files, uploading them to the webpage surpassed the file size limit. To solve this problem, we designed a Python executable that accepts a CAT file and removes all the unnecessary information, considerably reducing its size. The program is freely accessible through the webpage.

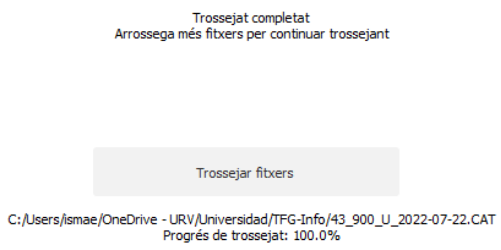


Figure 56. Program that reduces CAT files size

8.2.4.2 Import Data referencing the Main Table

The most complex part of this functionality is the option to connect the table data via geographical location instead of a simple id connection. As we mentioned in the database design section, we created an optimized table that would allow us to search estate records given latitude and longitude efficiently. Thanks to this design, we can execute queries in the following format:

```
SELECT id, ST_Distance_Sphere(location, POINT(41.13147287,
   1.279994949)) AS distance
FROM estate
WHERE ST_Distance_Sphere(location, POINT(41.13147287, 1.279994949))
   < 10
ORDER BY distance ASC
LIMIT 1;
```

Code 10. SQL query obtaining the closest location to another

Another part to consider in this importing process is how the application will react if a record does not match the main table. To avoid errors, we first check the existence of the match, and if there is not one, we do not perform the insertion.

8.2.4.3 General Challenges

Between most of the import options, we found that we need to control aspects like repeated records, incorrect file types, or null values. That is why all import options implement specific error handle mechanisms to avoid possible fatal errors.

9 Evaluation

In the evaluation process of the web application, two distinct types of tests were conducted to assess its performance comprehensively. The first category involved testing the application's functionality while emulating a regular user's actions. This set of tests aimed to ensure that the web application operated as intended.

However, evaluating the web application solely based on its intended usage would not provide a complete picture of its robustness and security. Therefore, a second category of tests was devised, focusing on assessing the application's resilience against potential malicious users. The GET values were deliberately modified in these tests to simulate unauthorized attempts and exploit potential vulnerabilities. By intentionally tampering with the input values, the objective was to identify any weaknesses that could lead to data breaches, unauthorized access, or other security risks.

9.1 Functional Testing

00. Login		
Test	Expected	Correct
User introduces valid credentials	The system takes the user to the page he came from or the logout page after a successful log in.	Yes
User introduces invalid credentials	An error message is shown, and the system asks the user to try again	Yes

Table 2. Test Set for UC 00. Login

01. Visualize Graphics		
Test	Expected	Correct
User enters the functionality	The system shows the graphs correctly	Yes

Table 3. Test Set for UC 01. Visualize Graphics

03. Select Table		
Test	Expected	Correct
User enters the functionality and selects one table	The system takes the user to the page they came from and sends the table information.	Yes

Table 4. Test Set for UC 03. Select Table

04. Logout		
Test	Expected	Correct
User logs out of the web application	The system removes the access of the user to restricted functionalities	Yes

Table 5. Test Set for UC 04. Logout

05. Import Data + extended UCs 06, 07 and 08		
Test	Expected	Correct
User starts an import data process and uploads a correct file	The system processes the file, updates the database, and informs the user of the changes	Yes
User starts an import data process and uploads an incorrect file type	The system shows the user an error message and asks them to try again	Yes
User starts an import data process selecting the import to new table with id reference to the main table, but the references do not match	The new table is created but none of the rows are added, so the table becomes empty	Yes
User starts an import data process selecting the import to new table with latitude and longitude reference to the main table with some references matching	The references that match are added to the new created table	Yes
User starts a file conversion with a JSON file generated from a query to Google Places API	The server forces the download of the converted CSV file	Yes

Table 6. Test Set for UC 05. Import Data and their extended UCs

09. Visualize Data (CRUD) + extended UCs 10, 11, 12, 13, 14		
Test	Expected	Correct
User filters the data with no matches	The system processes the filters and shows no rows	Yes
User changes to a page greater than the actual max page	The system shows no rows	Yes
User uses filter, matching multiple rows and navigate in allowed pages	The system remembers the filters and shows the correct data for each page	Yes
User changes table	The system shows the data for the new table	Yes
User creates new row	The system correctly creates the form, processes the data, and shows the success message	Yes

User deletes a row	The system deletes the row and shows the success message	Yes
User updates a row	The system updates the row values in the database and shows the success message	Yes
User reads a row	The system shows the row values	Yes
User deletes a table	The system removes the table from the database	Yes

Table 7. Test Set for UC 09. Visualize Data (CRUD) and their extended UCs

9.2 Security Testing

Tampered Data Tests		
Test	Expected	Correct
GET values modified to access restricted table users in CRUD visualization	The system shows the user an error message	Yes
GET values modified to update a row in Update functionality with incorrect primary key	The system shows the user an error	Yes
GET values modified to delete restricted table estate in Delete Table functionality	The system shows the user an error message	Yes
GET values modified to access inexistent records in read, update and delete functionalities	The system shows the user an error message	Yes

Table 8. Test Set for functionalities whose GET values can be modified

10 Conclusions

In conclusion, this web application project has provided valuable personal growth and learning opportunities in web development. Throughout the project, I have gained practical experience in various aspects of the development process, from conceptualizing the application's requirements to implementing its functionalities. By working on this project, I have achieved the following:

1. Enhanced Technical Skills: The project strengthened my proficiency in HTML, PHP, CSS, and database management. I gained a deeper understanding of these technologies by implementing them in a real-world application.
2. Problem-Solving Abilities: Developing the web application required me to overcome numerous challenges and find solutions. I honed my problem-solving skills by identifying issues, researching potential solutions, and implementing effective resolutions. This experience has improved my ability to analyze problems and think critically.
3. Understanding of Web Development Best Practices: Throughout the project, I followed established best practices in web development, such as code organization, security measures, and optimization techniques. Adhering to these practices has given me insights into industry standards and ensured the application is efficient, secure, and maintainable.
4. Collaboration and Project Management: While working on the project, I collaborated with teammates, solicited feedback, and incorporated suggestions for improvement. This experience enhanced my teamwork and communication skills and my ability to manage project timelines and prioritize tasks effectively.
5. User-Centric Approach: Developing a user-friendly interface and incorporating user feedback allowed me to focus on user experience and usability. By considering the needs and expectations of the target audience, I developed an application that provides a seamless and intuitive experience.

Overall, this web application project has been a valuable learning experience, equipping me with practical skills and knowledge that will serve me well in my future endeavors. It has allowed me to apply theoretical concepts, explore industry best practices, and gain hands-on experience in a real-world development scenario. The lessons learned from this project will significantly contribute to my growth.

11 References

[1] Rajabifard, A., Williamson, I., Steudler, D., Binns, A., & King, M. (2007). Assessing the worldwide comparison of cadastral systems. Land Use Policy, 24(1), 275–288.

[2] https://www.sedecatastro.gob.es/Accesos/SECAccDescargaDatos.aspx [Accessed May 15, 2023]

[3] https://developers.google.com/maps/documentation/places/web-service/search-nearby [Accessed May 15, 2023]

[4] https://mariadb.com/kb/en/foreign-keys/ [Accesed June 1, 2023]

[5] Ruijie Tian, Huawei Zhai, Weishi Zhang, Fei Wang, and Yao Guan, "A Survey of Spatio-Temporal Big Data Indexing Methods in Distributed Environment," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 15, no. 2022, pp. 4132-4155, 2022.

12 Appendices

12.1 Appendix A: User Manual

Introducción

Este es el manual de usuario para la aplicación web GEOCAD enfocada en la administración de datos del catastro.

La aplicación está diseñada para facilitar el trabajo con datos del catastro permitiendo la importación de datos tanto oficiales (proporcionados por el estado con el formato .CAT) como de otras fuentes de datos como Google API.

Toda la información es guardada en una base de datos optimizada para las funcionalidades implementadas permitiendo la visualización y operación de los datos.

Primeros pasos

El acceso a la aplicación web no tienen ningún prerrequisito. Simplemente es necesario acceder al siguiente enlace https://urdata.cat/app/. Una vez dentro de la web existe la opción de iniciar sesión. Los usuarios registrados que inicien sesión tendrán acceso a las funcionalidades más delicadas que pueden modificar directamente los datos almacenados en la base de datos.

No existe una opción de libre registro, para que usuarios no autorizados no puedan modificar datos de las bases de datos sin permiso.

Interfaz de usuario

La aplicación web cuenta con una interfaz de usuario amigable, diseñada para mejorar la experiencia y trabajo con ella. Los principales componentes son:

Barra de navegación

La barra de navegación está presente en todo momento en el encabezado de la aplicación. Su función es dar libre acceso a las distintas funcionalidades de la aplicación de forma intuitiva.

Además, esta busca orientar al usuario en todo momento indicando la funcionalidad en la que se encuentra actualmente mediante cambios de color en el encabezado, y si actualmente ha iniciado sesión o no:

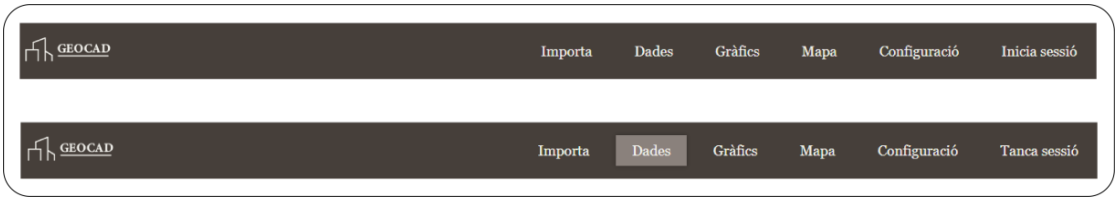


Figura 1. Encabezado como barra de navegación

Área de contenido

El área de contenido es el espacio central dónde los usuarios podrán interactuar con las distintas funcionalidades. Variará dependiendo de las necesidades de cada página, pero siempre manteniendo la estética de la aplicación web.

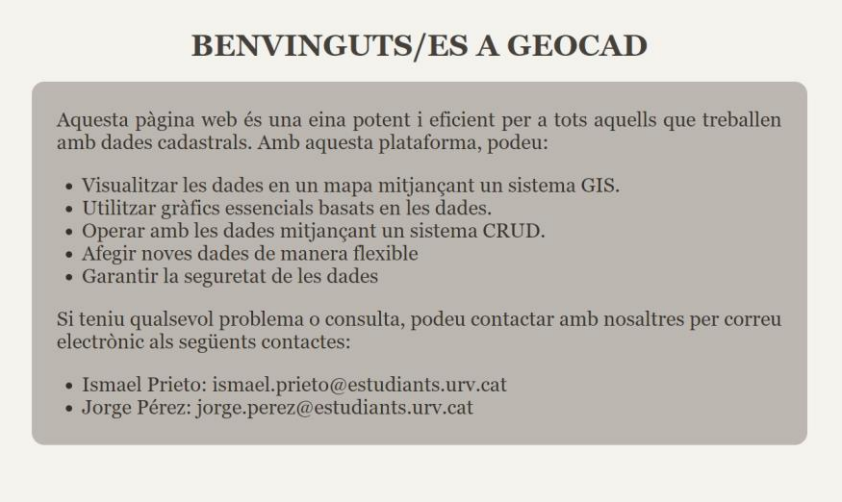


Figura 2. Área de contenido en la página de inicio

Pie de página

La aplicación incluye un pie de página en la interfaz de usuario:

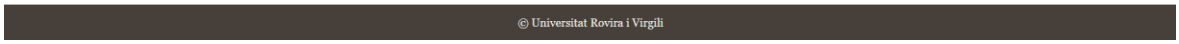


Figura 3. Pie de página

Características y funcionalidades

La aplicación web ofrece un amplio rango de funcionalidades. Ahora nos adentraremos una por una en su utilidad y uso.

Inicio y cierre de sesión

Estas funcionalidades permiten a los usuarios tanto iniciar sesión para acceder a funcionalidades restringidas, como cerrar la sesión una vez abierta para perder este acceso.

Este control de sesiones permite aumentar la seguridad de la base de datos restringiendo el acceso a usuarios no registrados y permitiendo a usuarios registrados cerrar su sesión si han acabado de trabajar en un dispositivo no seguro.

Para acceder a esta funcionalidad habrá que pulsar el último botón de la barra de navegación:

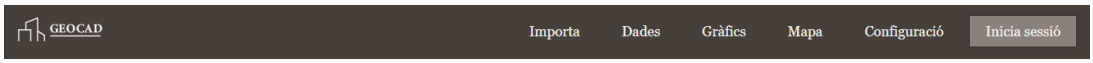


Figura 4. Botón para iniciar sesión

Una vez hallamos pulsado el botón de iniciar sesión (o hayamos intentado acceder a una funcionalidad restringida sin haber iniciado sesión) veremos la siguiente área de contenido pidiendo las credenciales:

Figura 5. Área de contenido para introducir credenciales

En el caso de introducir las credenciales incorrectas, el sistema volverá a pedirlas, informando al usuario de que las introducidas anteriormente son incorrectas.

Una vez el usuario ha iniciado sesión, en el mismo botón podrá finalizarla pulsando en un botón:

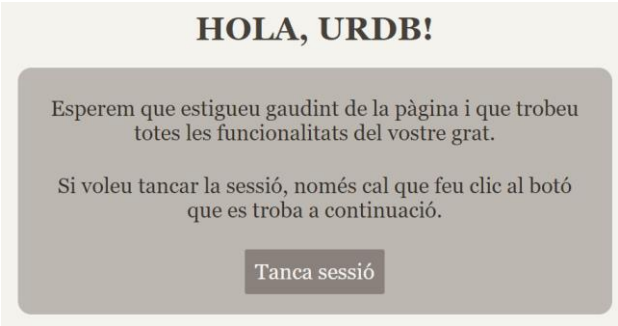


Figura 6. Área de contenido para cerrar sesión

Importación de datos

Esta funcionalidad permite a los usuarios que han iniciado sesión importar nuevos datos a la base de datos a través de la subida de ficheros a la web.

Esta cuenta con múltiples opciones de subida, que van desde la subida de ficheros que buscan añadir información a tablas ya creadas, subidas que buscan crear nuevas tablas o incluso subidas que buscan transformar el fichero a un formato con el que pueda trabajar fácilmente la web. Esto nos permite aumentar la base de datos con información externa.

Para acceder a esta funcionalidad, se pulsará el siguiente botón:

Figura 7. Botón para acceder a la importación de datos

Una vez accedido se nos presentarán las distintas posibles opciones para acceder a la deseada, con áreas de contenido similares en apariencia:

Figura 8. Opciones de importación

Una vez el usuario acceda a la opción deseada, se le presentará el siguiente menú para que pueda subir los ficheros:

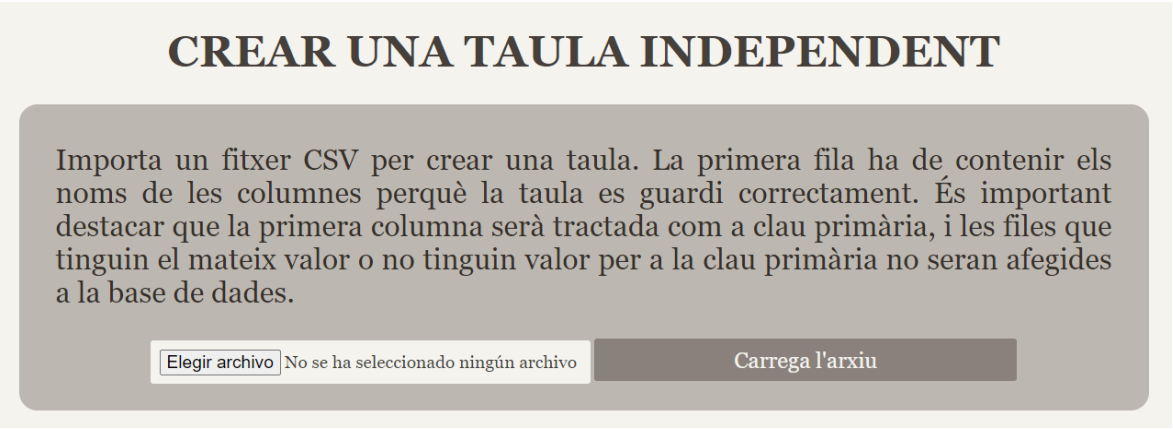


Figura 9. Menú de subida de ficheros

Una vez se suba el fichero, el servidor hará el procesamiento necesario para incorporar los datos a la base. En el caso de que ocurra cualquier tipo de error, se le comunicará al usuario permitiendo reintentar la tarea. Si no ha ocurrido ningún error el usuario podrá ver un resumen del procesamiento del fichero.

Aquí podemos ver un ejemplo de éxito:

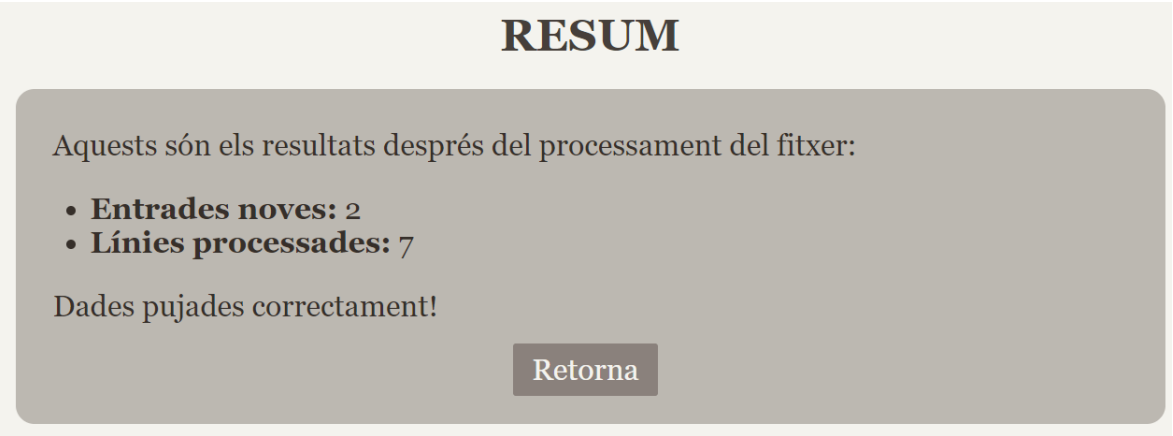


Figura 10. Procesado de fichero correcto

Un ejemplo de fallo:

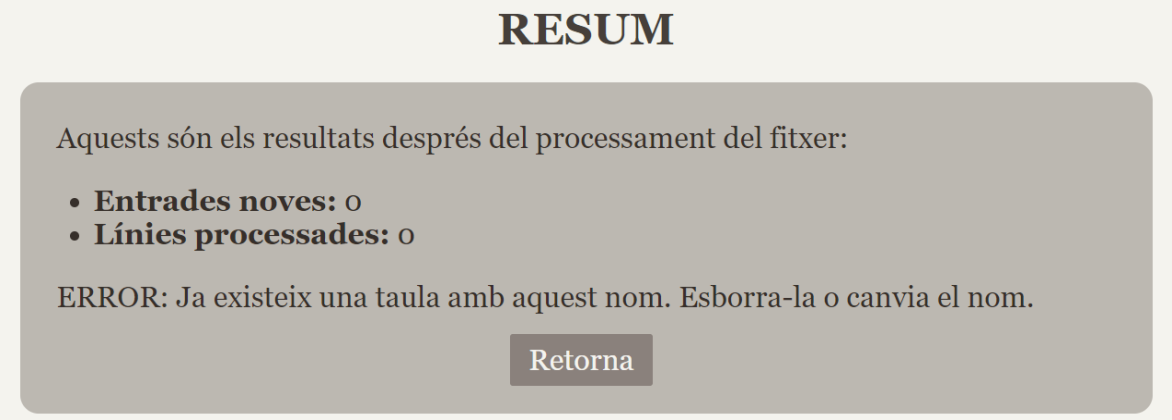


Figura 11. Procesado de fichero incorrecto

Sistema CRUD

Esta funcionalidad permite a los usuarios, una vez iniciada la sesión, realizar operaciones sobre las tablas de la base de datos mediante un sistema CRUD.

Este sistema permite operaciones de creación, lectura, actualización y eliminación de filas. Además, nos permitirá viajar sobre las distintas tablas y eliminar aquellas que no sean esenciales. Todo esto con un sistema de filtrado y de paginación para facilitar el trabajo con las tablas que contengan un gran número de datos.

Para acceder a esta funcionalidad se deberá usar el siguiente botón de la barra de navegación:

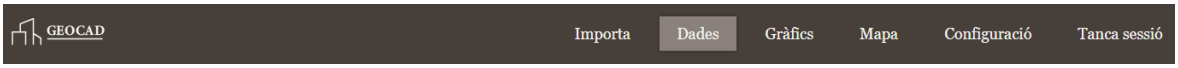


Figura 12. Botón para acceder al sistema CRUD

Una vez dentro nos encontraremos con la tabla principal por defecto. Para ilustrar mejor la visualización completa usaremos una tabla con un solo elemento que nos permita ver todos los componentes:

Figura 13. Área de contenido del sistema CRUD

Como podemos observar tenemos 3 botones iniciales que nos permitirán, crear una nueva fila, cambiar de tabla o eliminar la tabla actual. Luego tendremos dentro de la tabla una sección de filtrado que nos permitirá introducir datos de filtrado para cada una de las columnas. Dentro de cada fila podremos acceder a la lectura actualización y eliminación de las filas. Finalmente vemos el sistema de paginado, al que podremos acceder mediante botones o indicando una página concreta.

Nos adentraremos primero en el cambio de tabla, que nos abrirá una selección para escoger la tabla deseada pulsando en su botón, que luego será la que podremos visualizar:

Figura 14. Selección de tabla

Ahora veremos el menú similar que observará el usuario a la hora de eliminar tanto la tabla como una fila en concreto, donde el usuario recibirá una petición de confirmación y una vez aceptada, se eliminará lo seleccionado:

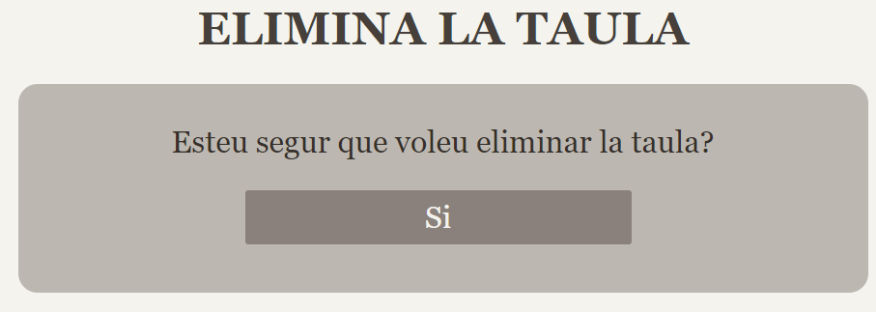


Figura 15. Menú de eliminación

Los menús de creación y actualización también son similares, dándoles a los usuarios múltiples campos para rellenar los nuevos datos y un menú de envío:

Figura 16. Menú de actualización

Finalmente, también podemos observar el menú de lectura que nos enseñará todos los datos pertinentes:

Figura 17. Menú de lectura

Visualización de gráficos

Esta funcionalidad permite observar los datos guardados en la tabla principal mediante gráficos predefinidos.

Para acceder a esta funcionalidad es necesario acceder al siguiente botón:

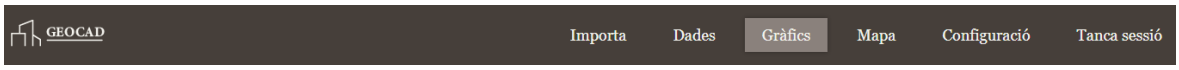


Figura 18. Botón para acceder a gráficos

Esto permite la visualización sencilla de patrones en los datos. Este es su área de contenido:

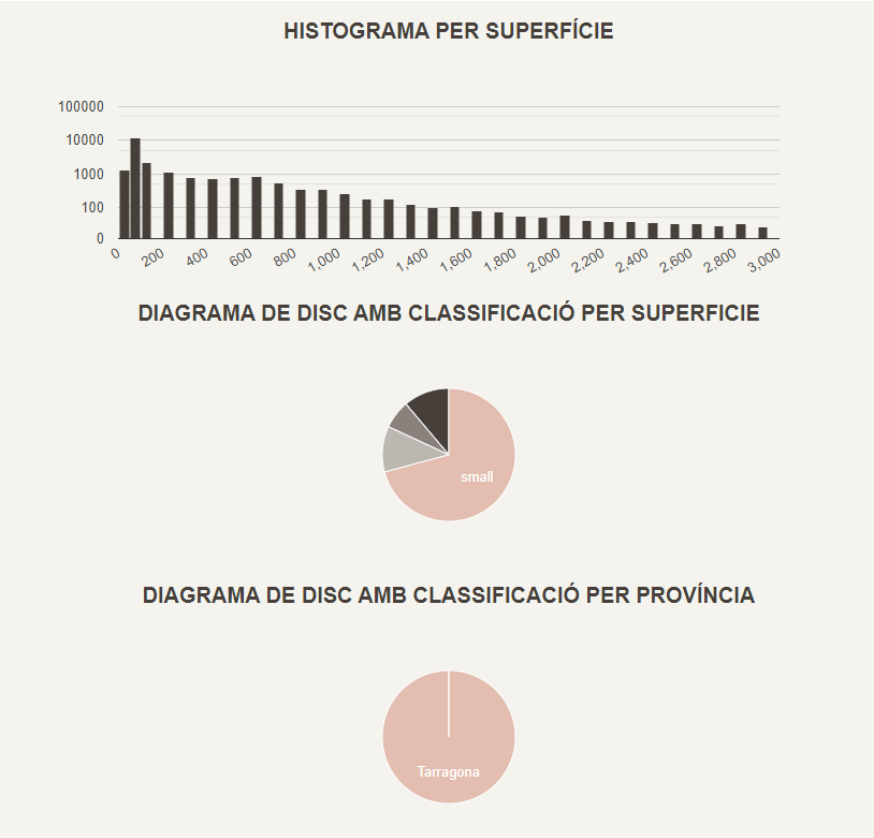


Figura 19. Área de contenido de funcionalidad gráficos

Ajustes

Esta funcionalidad permite modificar los valores predeterminados de la visualización del mapa.

Se accede desde el siguiente botón en el encabezado:

Figura 20. Botón acceso configuración

Desde aquí se podrá ver el siguiente menú que te indica la provincia donde quieres que se inicie la vista predeterminada:

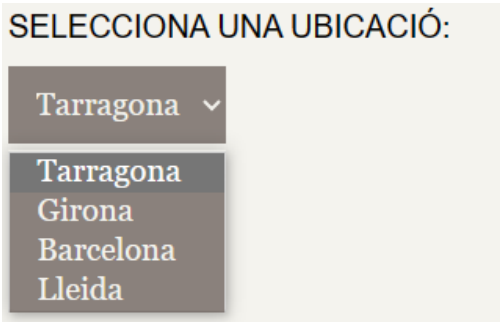


Figura 21. Menú selección vista predeterminada

Una vez seleccionada la opción que se desea y seleccionado el botón de enviar, mostrará información de que se ha realizado el cambio y las coordenadas establecidas:

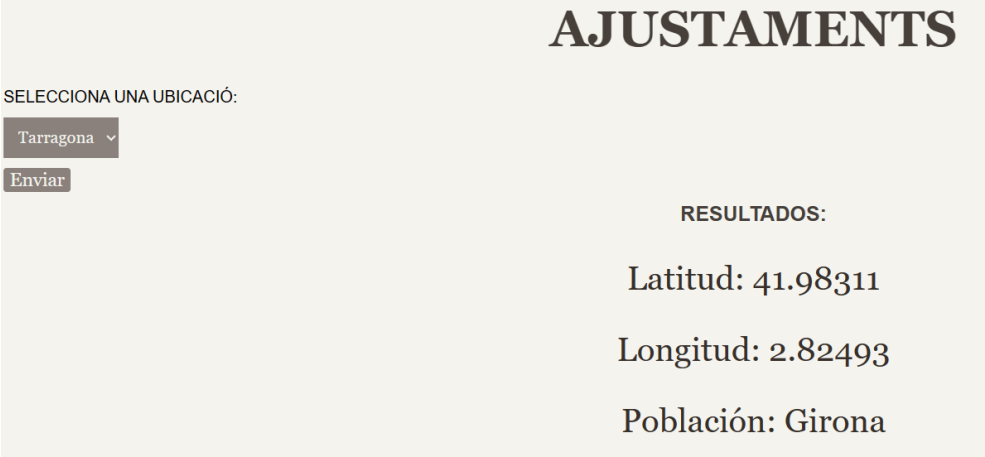


Figura 22. Modificación visión mapa configuración

Realizado este cambio, si volvemos a la página de mapas, veremos que de manera predeterminada nos ha modificado el punto de visión a otra provincia:

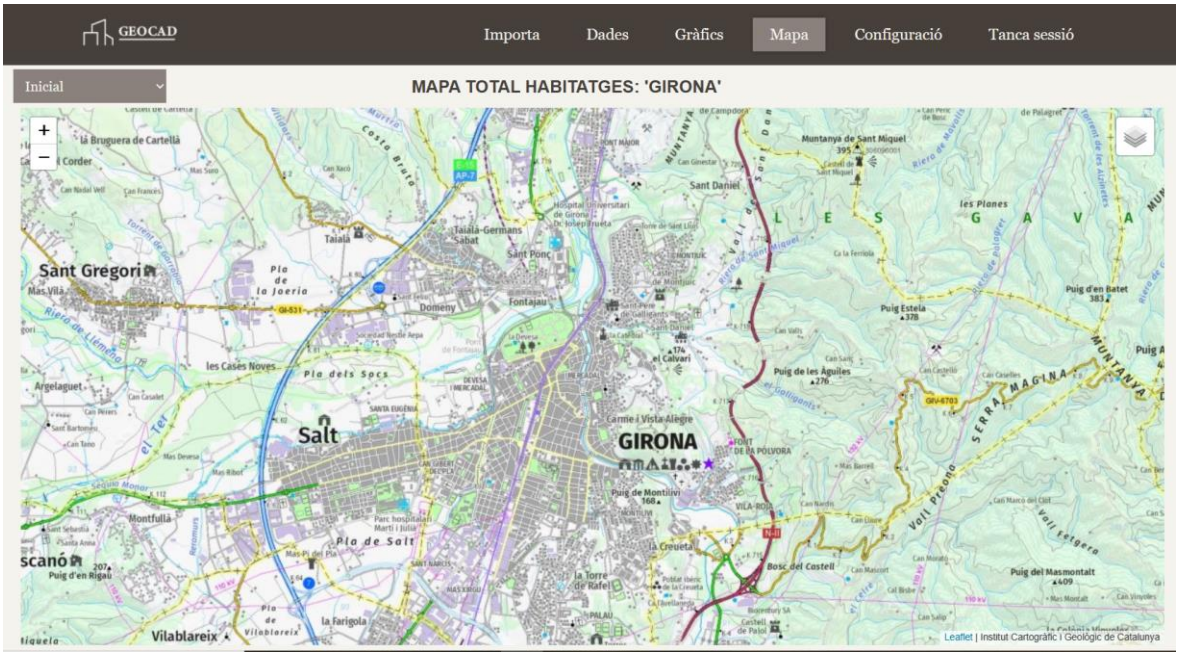


Figura 23. Mapa visualización predeterminada Girona

Finalmente queda la opción de modificar rango, donde se mostrará 3 campos que el usuario puede rellenar y enviar para así modificar los valores que se representarán en el mapa de superficie y el gráfico:

Figura 24. Opción modificar rango

Visualización de mapas

Esta funcionalidad permite al usuario observar y representar los distintos valores en la base de datos representados en el mapa.

El acceso a la publicación se realiza seleccionando la siguiente opción en el encabezado:

Figura 25. Botón acceso a mapa

Tras seleccionar esta opción se visualizará el mapa predeterminado que en un principio no refleja ningún punto seleccionado:

Figura 26. Mapa predeterminado provincia Tarragona

Desde el menú que se muestra en la esquina superior derecha, se podrá interactuar con el mapa, modificando el estilo de ilustración del mapa y la adicción de los puntos procedentes de las referencias catastrales registradas en la tabla principal de la base de datos:

Figura 27. Menú mapa predeterminado visualización topográfica gris

Si en este menú seleccionamos la opción '*CATASTRO'* el mapa mostrará un punto en la posición correspondiente del mapa por cada referencia catastral.

Figura 28. Mapa predeterminado visualización inmueble registrado

En la esquina superior izquierda de la pantalla contamos con un menú, si se selecciona nos permitirá cambiar entre las distintas visualizaciones de datos en el mapa que ofrece la aplicación:

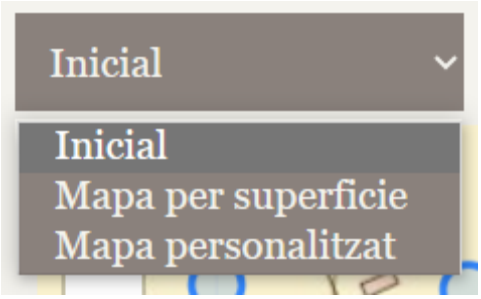


Figura 29. Menu navegación opciones de mapas

Si seleccionamos la siguiente opción '*Mapa per superficie*' vemos como se vuelve a representar el mapa inicial, pero con modificaciones en el menú de la esquina superior derecha:

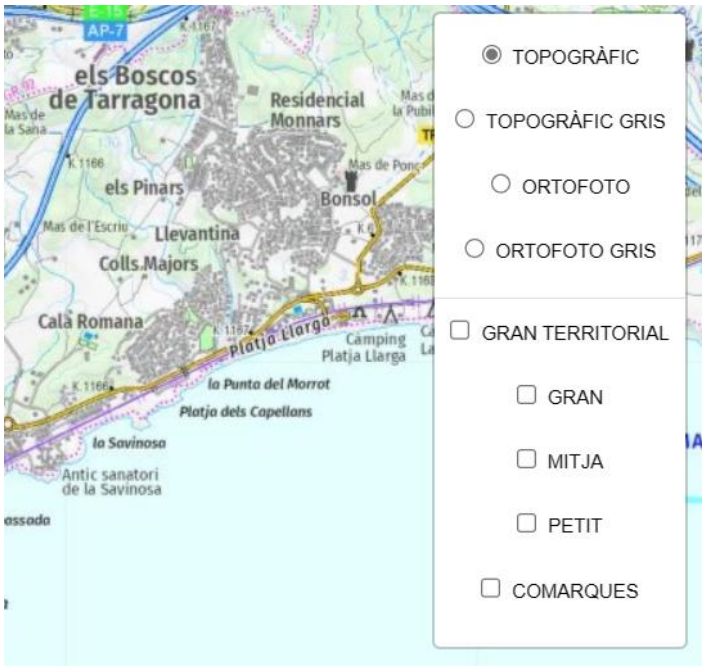


Figura 30. Menú mapa por superficie

En este menú, además de editar el estilo, también podrás decidir qué información dese visualizar según su tamaño registrado. Esta información será diferenciada por colores cuando se ilustren los puntos en el mapa:

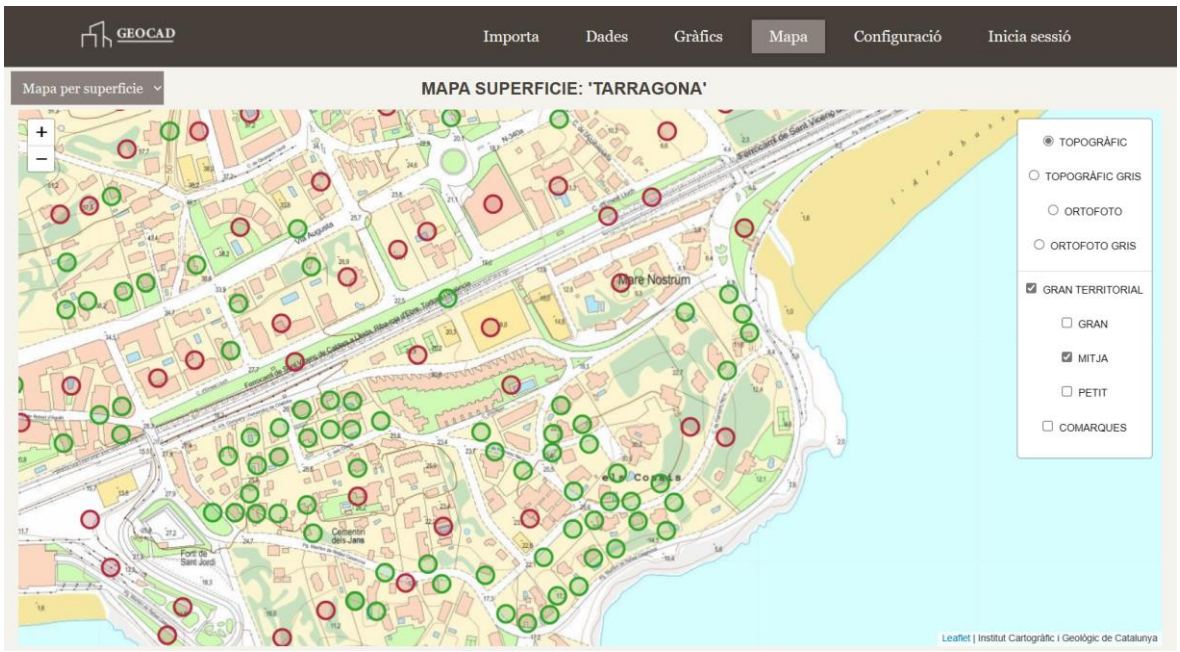


Figura 31. Mapa por superficie mostrando puntos diferenciados

Por último, queda la funcionalidad '*Mapa personalizat*' a la que se accede desde el menú de la esquina superior izquierda el cual te llevará al mismo menñu que se muestra en la Figura 14.

Luego nos llevará a un menú similar que nos solicitará seleccionar el dato de la tabla que queremos emplear para diferenciar los puntos:

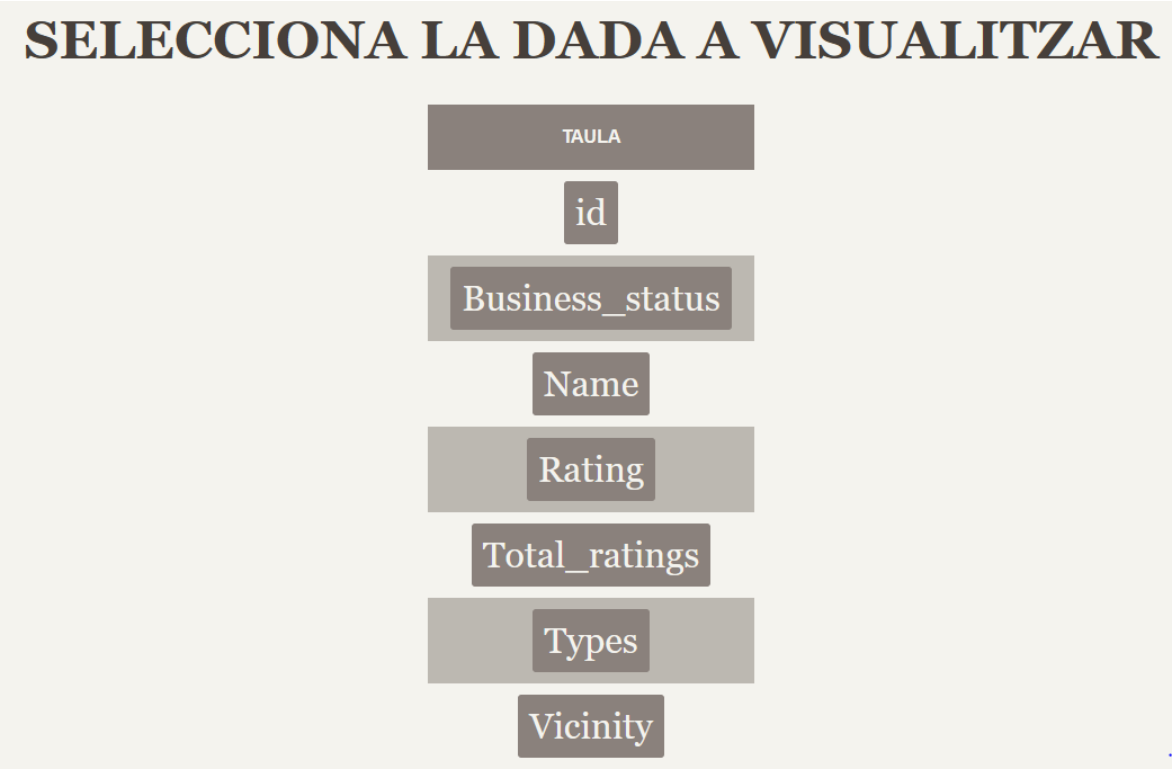
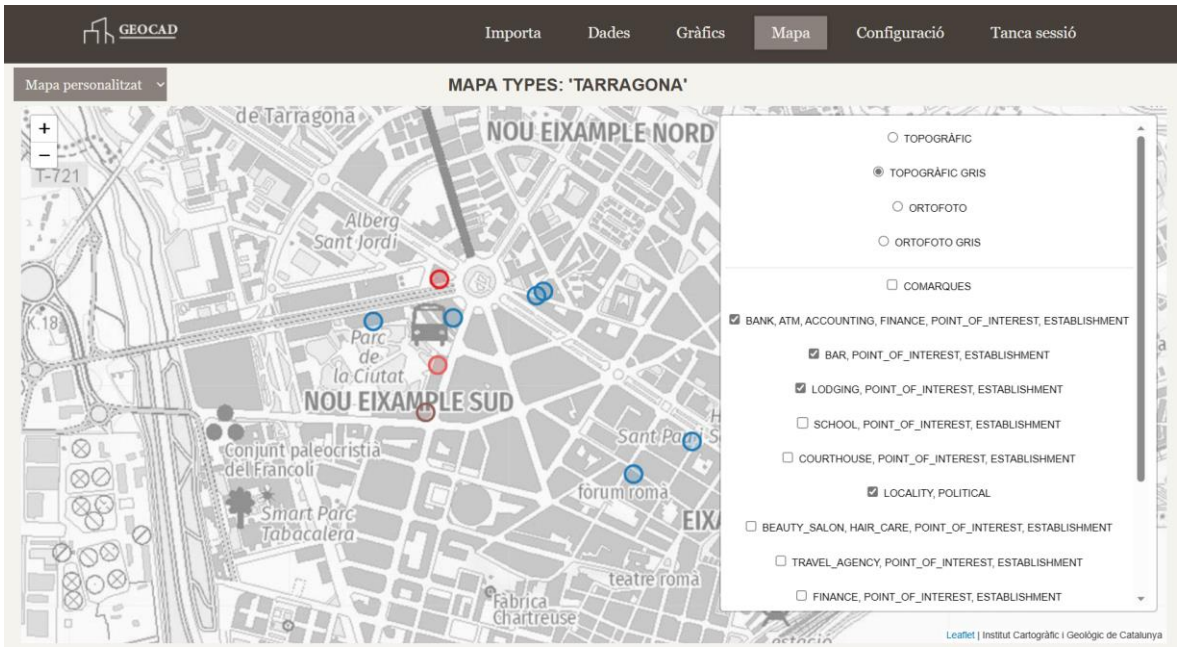


Figura 32. Tabla selección de dato a visualizar

Una vez seleccionado, se volverá a mostrar el mapa, con el único cambio de que en el menú de la derecha se mostrarán las distintas agrupaciones de datos, además de los puntos de cada grupo representados con un color distinto:



En conclusión, esta guía de uso ha presentado las diversas funcionalidades de la aplicación, proporcionando a los usuarios las herramientas necesarias para aprovechar al máximo sus capacidades. Esperamos que esta aplicación les sea de gran utilidad y les permita explorar y visualizar los datos del catastro de manera eficiente y flexible.

12.2 Appendix B: Installation Manual

Introducción

Este es el manual de instalación para la aplicación web GEOCAD, enfocada en la administración de datos del catastro.

La aplicación está diseñada para facilitar el trabajo con datos del catastro, permitiendo la importación de datos tanto oficiales (proporcionados por el estado en formato .CAT) como de otras fuentes de datos como Google Places API. El acceso a la aplicación web se puede realizar sin necesidad de realizar ninguna instalación, simplemente accediendo al siguiente enlace https://urdata.cat/app/.

En caso de que desee migrar la aplicación a otro servidor, a continuación, se explica cómo hacerlo.

Primeros pasos

Para llevar a cabo este proceso, será necesario tener a mano la dirección del servidor, el nombre de usuario y la contraseña. Además, deberá instalar la aplicación FileZilla y descargar el archivo ZIP con la aplicación. También se recomienda tener instalada la herramienta Visual Studio Code, ya que facilitará la edición de documentos.

Puede descargar Visual Studio Code desde el siguiente enlace:

https://code.visualstudio.com/download

Guía de instalación

A continuación, se detallarán los pasos a seguir para poder instalar y ejecutar la aplicación.

Instalación en servidor remoto

1. Descargar el archivo .zip con el código de la aplicación.
2. Descomprimir el archivo .zip en una ubicación segura de su dispositivo.
3. Descargar FileZilla desde el siguiente enlace: https://filezilla-project.org/
4. Abrir la aplicación FileZilla e insertar las credenciales de su servidor en el menú superior de la aplicación como se muestra seleccionado en rojo en la Figura 1.

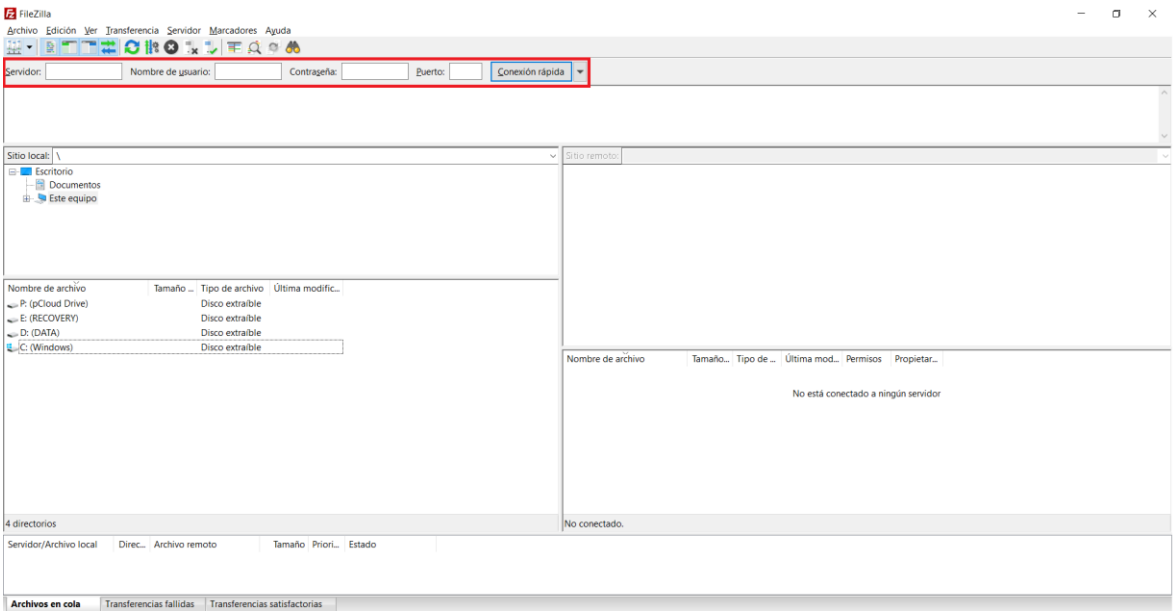


Figura 18. Aplicación FileZilla

5. Después de ingresar las credenciales correctamente, en el menú de la derecha del sitio remoto, accede a la carpeta "www".
6. En el menú de la derecha del repositorio local, busca la ubicación donde descomprimiste el archivo de la aplicación.
7. Arrastra el directorio raíz "app" del archivo descomprimido hasta el menú de la izquierda del repositorio remoto, dentro de la carpeta "www".

Instalación y acceso a la base de datos

El siguiente paso consiste en configurar la base de datos en tu servidor para que la aplicación sea completamente funcional. Sigue estos pasos:

1. Asegúrate de que tu usuario tenga acceso para crear nuevas bases de datos, junto con las credenciales necesarias. Si no tienes estos permisos, consulta con tu proveedor de servicios de hosting.
2. Accede a PhpMyAdmin con la dirección correspondiente nuevo servidor.
3. Una vez en PhpMyAdmin, completa los campos de servidor, usuario y contraseña con tus credenciales.
4. Ahora, selecciona la opción "SQL" en el menú superior. Se abrirá una ventana para ingresar texto donde deberás copiar el siguiente código:

```
CREATE DATABASE IF NOT EXISTS ur_db;
USE ur_db;
CREATE TABLE IF NOT EXISTS estate (
  id varchar(14) NOT NULL,
  province varchar(30) NOT NULL,
  size int(10) NOT NULL,
  location point NOT NULL,
  size_class varchar(10) DEFAULT NULL,
  PRIMARY KEY (id),
```

```
  UNIQUE KEY uq_location (location(25)),
  SPATIAL KEY idx_location (location)
)
CREATE TABLE IF NOT EXISTS ranges (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name varchar(50) NOT NULL,
  lower_limit int(11) NOT NULL,
  upper_limit int(11) NOT NULL
);

CREATE TABLE IF NOT EXISTS users (
  user varchar(255) NOT NULL,
  password varchar(255) NOT NULL,
  PRIMARY KEY (user)
);

CREATE TABLE IF NOT EXISTS hidden_tables (
  table_name varchar(255) NOT NULL,
  PRIMARY KEY (table_name)
);

INSERT INTO hidden_tables (table_name) VALUES ('hidden_tables'),
('ranges'), ('users');
```

5. Después de copiar este comando, selecciona "Continuar". Si tienes los permisos adecuados, la base de datos se creará con todas las tablas necesarias

Actualización aplicación a la base de datos creada

Una vez que la base de datos esté configurada, deberemos actualizar la aplicación para que realice las consultas correspondientes a esta nueva base de datos. Sigue los siguientes pasos:

1. Desde FileZilla, accede nuevamente al servidor remoto como explicamos en anteriormente.
2. Una vez que hayas iniciado sesión, en el menú de la derecha del servidor remoto, accede al archivo "database.php" que se encuentra en la siguiente ruta:

   ‘/www/app/functionalities/database’
3. Después de abrir el archivo, haz clic derecho en "database.php" y selecciona la opción "Ver/Editar". Si es la primera vez que editas un archivo PHP, se te pedirá que selecciones la aplicación para abrirlo. En este caso, recomendamos utilizar Visual Studio Code.
4. Se abrirá el archivo y será necesario actualizar los campos que contienen asteriscos en $dbUsername y $dbUserPassword, como se muestra en la siguiente imagen:

database.php

C: > Users > ijorg > AppData > Local > Temp > fz3temp-2 > database.php

```php
<?php
class Database
{
    public static $dbName = 'ur_db' ;
    private static $dbHost = 'localhost' ;
    private static $dbUsername = '******';
    private static $dbUserPassword = '*******';

    private static $cont  = null;

```

Figura 19. Código dabase.php a actualizar

5. Una vez hayas realizado los cambios, guarda la modificación, regresa a FileZilla y selecciona la opción "Sí" en la ventana emergente que aparece.

Acceso aplicación servidor remoto

Para acceder a la aplicación, debes abrir tu navegador de confianza y escribir la dirección de tu servidor seguida de la carpeta "/app". El resultado final será algo similar a: "dirección_servidor/app". Si has seguido los pasos correctamente, verás la ventana de inicio en tu navegado.

Figura 20. Ventana de inicio resultado final instalación