Faculdade de Engenharia da Universidade do Porto

Software Systems Architecture

Homework #05 - Library System Architecture

Team T23:
Anete Pereira (up202008856)
Bárbara Carvalho (up202004695)
David Fang (up202004179)
Milena Gouveia (up202008862)
Pedro Correia (up202006199)

Introduction

The design of a modern library system architecture is crucial to meeting the evolving needs of library patrons and optimizing library operations. In response to the demand for seamless access to library resources, our architecture focuses on providing patrons with user-friendly functionalities and efficient management tools. This report outlines the key decisions and considerations, the functional requirements and the chosen architectural styles for creating a modern and efficient solution that meets both the staff and the patron needs in an Automated Library System.

Ambiguities and Decisions

When designing the Library System Architecture, we identified multiple ambiguities in the problem's specification and in order to address these uncertainties within the system, we made several decisions based on our understanding of the project requirements. Below are the aforementioned ambiguities along with the decisions we made in order to address them.

Data types

- Notifications - due dates and fine alerts
- User accounts
- Media - books, dvd's, electronic books etc
- Fines - value needed to be paid for a specific book and other related information

System Usage

People can use the system two different ways:

- In the library there are stations that everyone can use, as long as they are logged in, to see the available media in the library catalog.

- Via a web application where users, after logged in, can search for media in a library, see the books they borrowed, the due dates for each, their notifications, pay fines, and if the media is online they can use it in the app.

Collaborations

For this we thought that our library belongs to a chain of libraries from a single enterprise. Like this we could have the enterprise manage a global database that contains the information of all the libraries. With this, when library A and B collaborate, A would ask the global database the media catalog of B.

Major Modules

We decided to separate the program into different controllers, for example Authentication Controller, Search Controller, where each controller is responsible for what the name entails. With this we wanted to make a system where we can develop a controller without having to worry about the other controllers.

Typical Scenarios

We outlined typical scenarios that users may encounter while interacting with the system. These scenarios include:

1. Borrowing Books from a Library Station:
    a. A library patron approaches a library station and logs in to the system.
    b. They search for a specific book using the station interface, browsing through available categories or using the search function.
    c. Upon finding the desired book, they select it and proceed to borrow it through the station interface.
    d. The system updates the patron's account to reflect the borrowed book and notifies them of the due date.
2. Searching for Media and Accessing Online Resources:
    a. A patron logs in to the web application and navigates to the search functionality.
    b. They enter keywords or use filters to search for specific media, such as books, journals, or online resources.
    c. Upon finding relevant resources, the patron accesses them directly through the application if available online.
    d. If the resource is physical, the patron can check its availability in nearby libraries and reserve it for pickup.
3. Paying Fines and Notifications Handling:
    a. A patron logs in to the web application and accesses their account dashboard.
    b. They notice a notification alerting them of an overdue book and associated fine.
    c. The patron proceeds to view the details of the fine and pays it in the front desk of the library, in person
    d. The fine is then eliminated from the patron profile.
4. Library Collaboration and Global Database Access:
    a. A patron accesses the library's web application and logs into their account
    b. The patron initiates a search for a specific book in their local library's catalog.
    c. After searching, the patron doesn't find the desired book available in their local library's catalog.
    d. Leveraging the collaboration feature, the patron explores catalogs from other libraries within the enterprise's network through the web application.
    e. The patron finds the desired book available in another library's catalog and proceeds to reserve it for pickup at their local library.
5. Inventory management - add a new book:
    a. The staff member accesses the library's administrative portal and logs into their account using their credentials.
    b. Upon logging in, the staff member navigates to the inventory management section of the administrative portal.
    c. The staff member initiates the process of adding a new book to the inventory by selecting the "Add New Book" option.
    d. The staff member enters the necessary details of the new book, including title, author, ISBN, genre, publication date, the book cover image and any other relevant information.

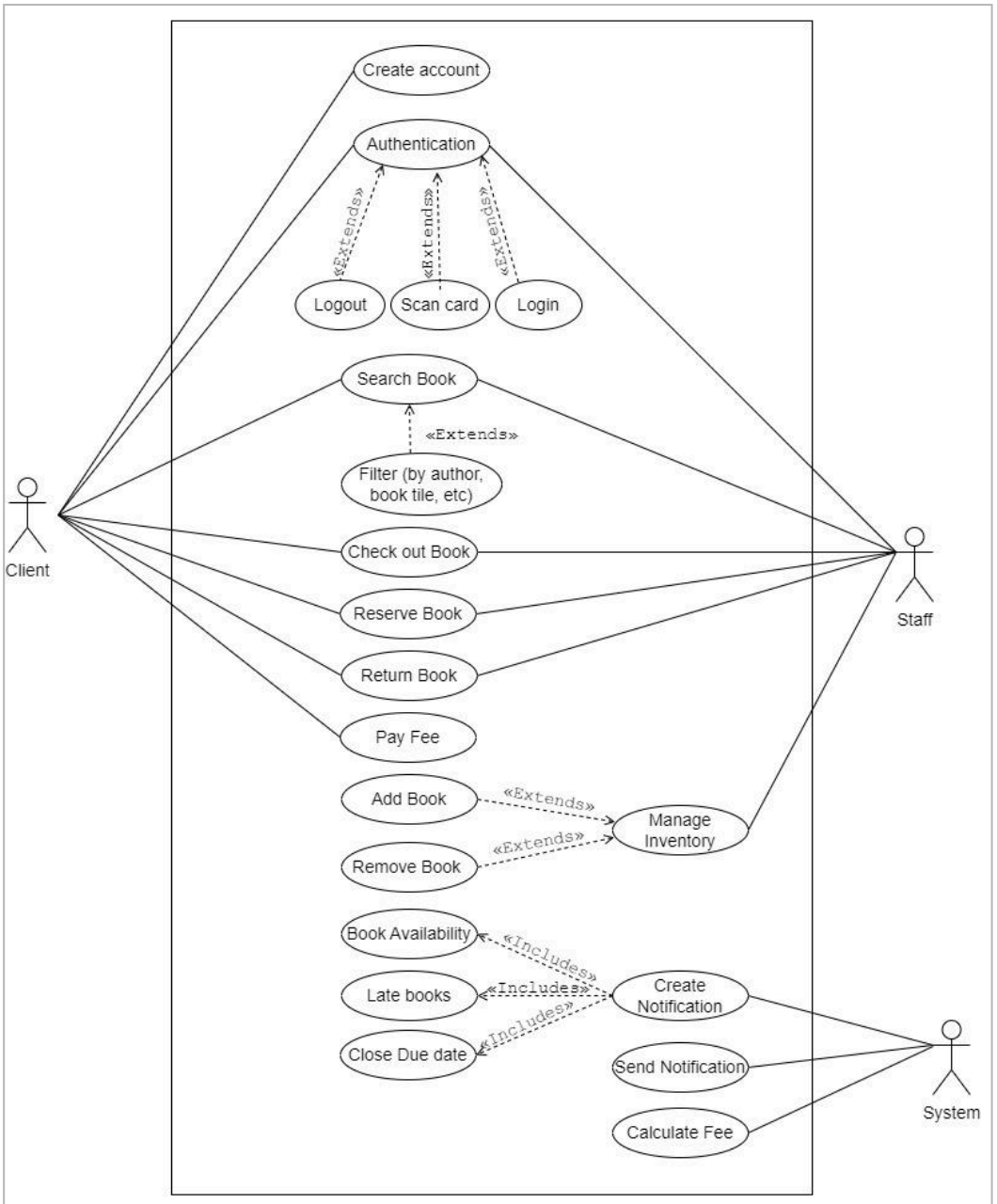
e. After reviewing and confirming the entered details, the staff member adds the new book to the library's catalog and inventory.
f. The system updates the library's inventory database with the new book entry, ensuring that it is now available for patrons to discover and borrow.

Architectural Style

1. Use case diagram

We opted to start the architecture implementation for the Automated Library System with a use case diagram, which offers a succinct portrayal of the system's functionalities and interactions from diverse actors' perspectives, encompassing clients, staff, and the system itself. This provides a comprehensive understanding of the primary objectives and actions undertaken by each entity within the system. Additionally, the diagram serves as a pivotal instrument for requirement analysis and validation, enabling the identification and prioritization of crucial use cases. This ensures that the system's design remains aligned with the users' needs and expectations, while also uncovering any potential gaps or inconsistencies in the system requirements early in the design phase.

Moreover, the use case diagram fosters effective communication and collaboration among project stakeholders by providing a shared vocabulary for discussing the system's functionality and requirements. This facilitates enhanced coordination between developers, designers, and end-users, ultimately promoting a more cohesive and user-centric system design. By commencing with a use case diagram, we establish a robust groundwork for the architecture of the automated library system, ensuring its efficacy in meeting the diverse needs of its users and stakeholders.



Name	Element	Description
Association	_____	A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
Include	«Includes» --------->	A line between use cases. One use case includes another use case means that the included use case is a part of the main use case and is essential for its execution.
Extend	«Extends» --------->	A line between use cases. One extended use case indicates one additional behavior of the main use case. It is optional.
Actor		Each actor has a unique name that describes the role of the user who interacts with the system.

Figure 1: Use case diagram

2. C4 model for visualizing software architecture

We opted for the C4 model for visualizing software architecture, due to its capacity to provide a structured and clear representation of the system's design across various levels of abstraction. This model allows us to create context, container, component and code diagrams, catering to both high-level overview and detailed component-level understanding. Similar to the use case diagram, the C4 model offers a systematic approach to understanding the system's structure and behavior, addressing diverse stakeholders' perspectives. While the C4 model also allows for the creation of code-level diagrams, we chose to represent only the first three levels as we aimed to maintain a balance between comprehensiveness and simplicity in our architectural documentation to avoid going into excessive detail. Overall, the adoption of the C4 model laid a solid foundation for discussing, analyzing, and evolving the architecture of the automated library system, enhancing its adaptability to meet the evolving needs of users and stakeholders.

a) System context diagram

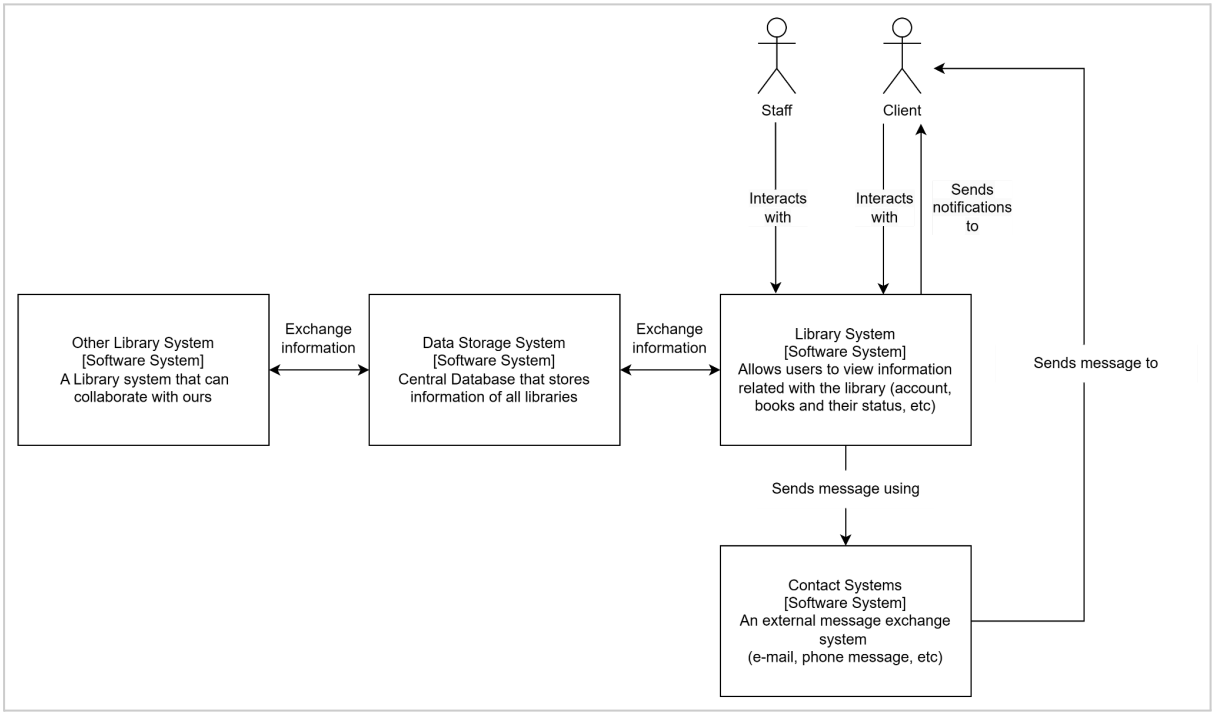


Figure 2: Context diagram

The System Context diagram serves as an initial step in designing the architecture of a software system, providing a broad overview that allows for a comprehensive understanding of the system's scope.

In this diagram, the systems are represented as a box, surrounded by its users and other interacting systems. The emphasis lies on identifying actors, roles, as well as software

systems, rather than delving into technical intricacies such as technologies and protocols. This zoomed-out view offers a high-level perspective suitable for communication with non-technical audiences.

In our context, there are two primary actors: the staff and the client. Both actors interact directly with the library system, which serves as the central hub for managing that specific library. The system is further linked to a central data storage system, providing storage backup and retrieval of various resources. The central data storage system is interconnected with other library systems, fostering collaboration and resource sharing across multiple platforms.

Additionally, the library system interfaces with a contact system, enabling several other ways to communicate with the clients besides the notification system incorporated in the library system itself.

b) Container diagram

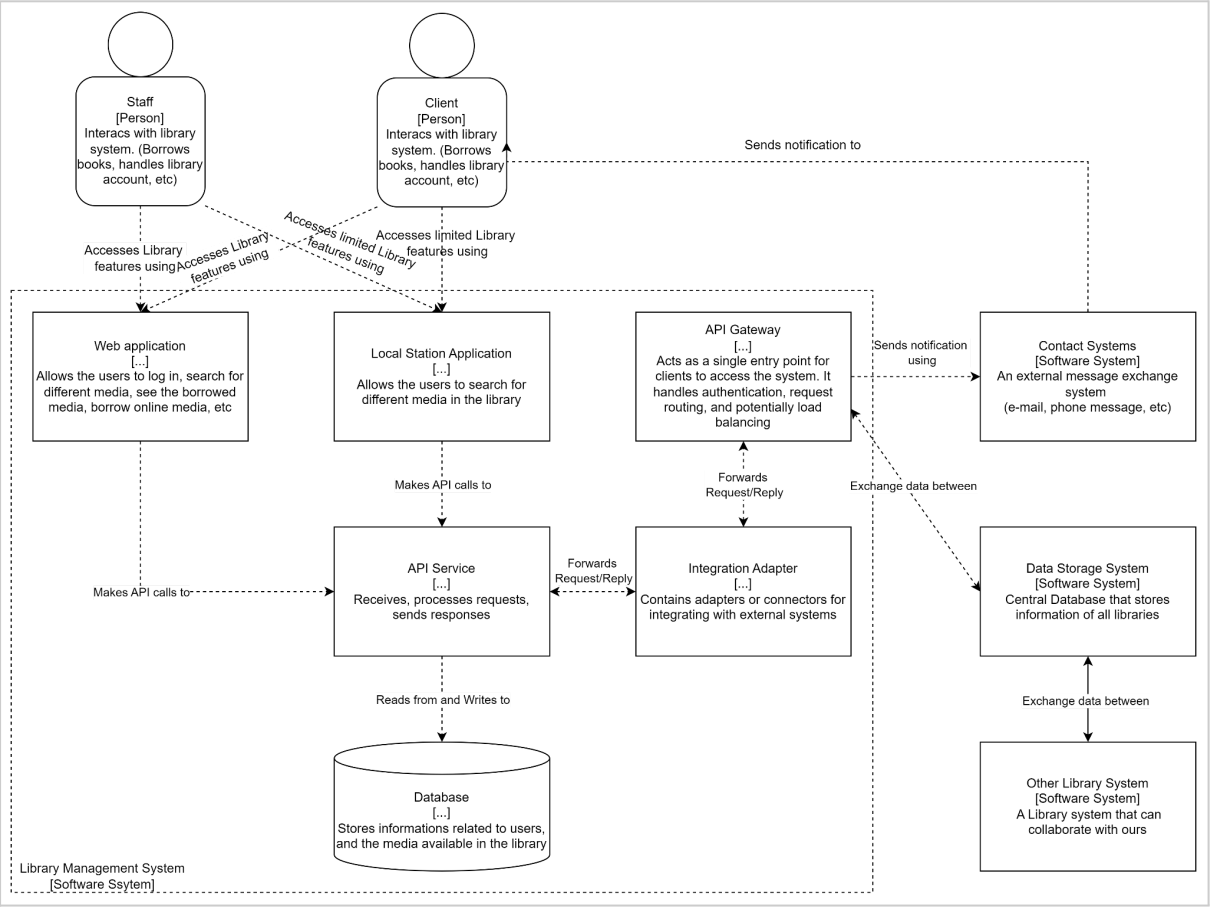


Figure 3: Container diagram 1

This diagram is meant to show a slightly more detailed view of how the system is meant to work. Basically, both ways to interact with the system, either via the web application or the local station, are going to be using the same system, the “API Service”. This service uses the library local database to solve the requests sent.

Moreover, having an integration adapter that converts and matches different types of requests or replies in the system, opens the opportunity for integration of several external applications, promoting versatility of our system.

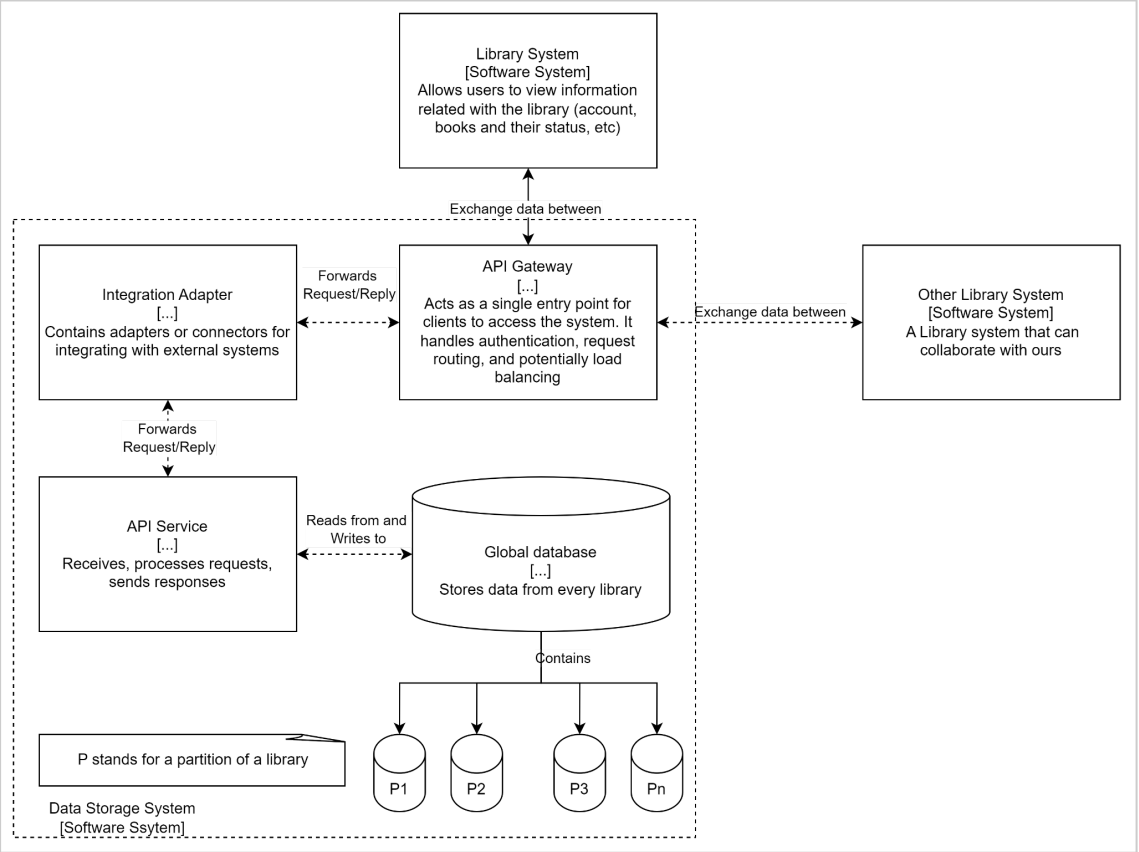


Figure 4: Container diagram 2

This diagram illustrates how the Data Storage System operates. It supports a global database containing information from every library. In other words, each library's local database serves as a partition of this global database. The exchanges that can happen with each library system can be, updating the database, reading another library media catalog etc.

c) Component diagram

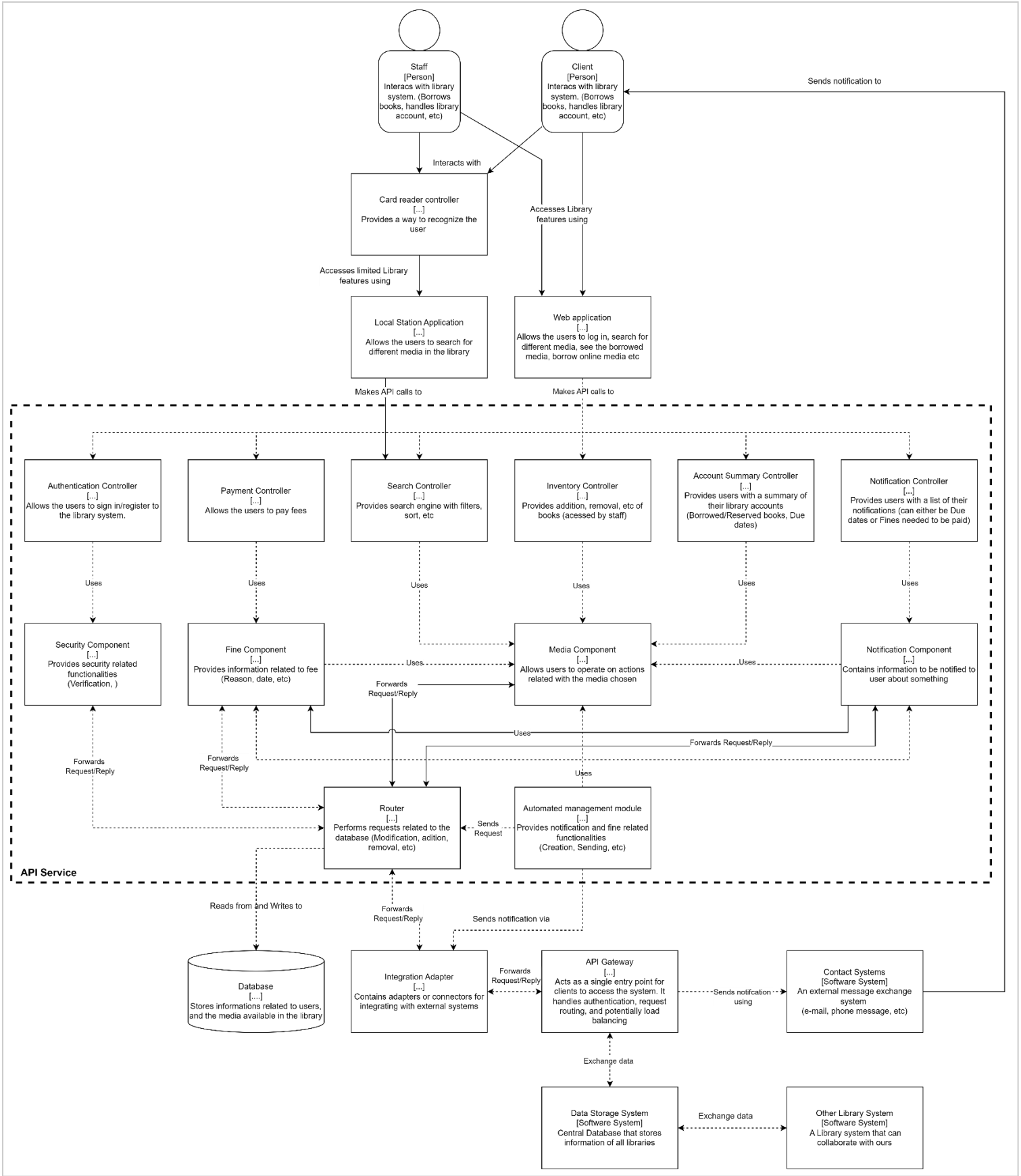


Figure 5: Component diagram

The component diagram provides a visual representation of the structural organization of our system, highlighting the key components that make up the system and how they interact with each other.

Our architecture is a mix of Layered architecture (not shown in the image) and Microservices. The system can be divided into three layers: the “Presentation” layer, where the UI design is developed, the “Logic” layer, where the controllers are located, that deals with most logic of the program, and the “Data” layer that deals with calls to the database, represented by the “Router”.

For Microservices we decided to divide the systems into different modules. For example the controllers are services that only deal with a specific part of the program, without needing to worry about the others. For example, the “Authentication Controller” deals with the creation of a new account, login and logout while the “Search Controller” only deals with the search that a user can make.

The “Automated Management Module” is a component that is meant to see if a user has a piece of media that is close to the due date or has already passed the date. For the first case it sends a notification, and the second case creates a fine that the user needs to pay. Both of them are then sent to the user either via an app notification or an email/phone message.

The “Router” would also request the information of another library system in the case of a collaboration.

We chose this approach because it enables independent scaling of different parts of the program, enhances resilience, flexibility, and simplifies system maintenance and repair in case of failure.

Conclusions

In conclusion, our approach for designing a modern library system architecture emphasizes clarity, collaboration, and user-centricity. By addressing ambiguities in the system specification and making strategic decisions, we laid a solid foundation for development. Adopting architectural styles such as the use case diagram and the C4 model provided structured frameworks for visualizing system functionalities and interactions. While the use case diagram provides a succinct portrayal of system objectives and actions, the C4 model enables a more detailed understanding of system structure and behavior. Moving forward, the implementation of these architectural styles, along with outlined functionalities and requirements, will enable the development of a robust and user-centric Automated Library System, meeting the evolving needs of patrons and staff.