Library System Architecture

Team 35 - André Morais, Miguel Rodrigues, Paul Rodrigues

The Architectural Proposal

The following figure depicts the architecture proposal. In the rest of the document we will better detail the components, discuss additional considerations and walk through some usage scenarios.

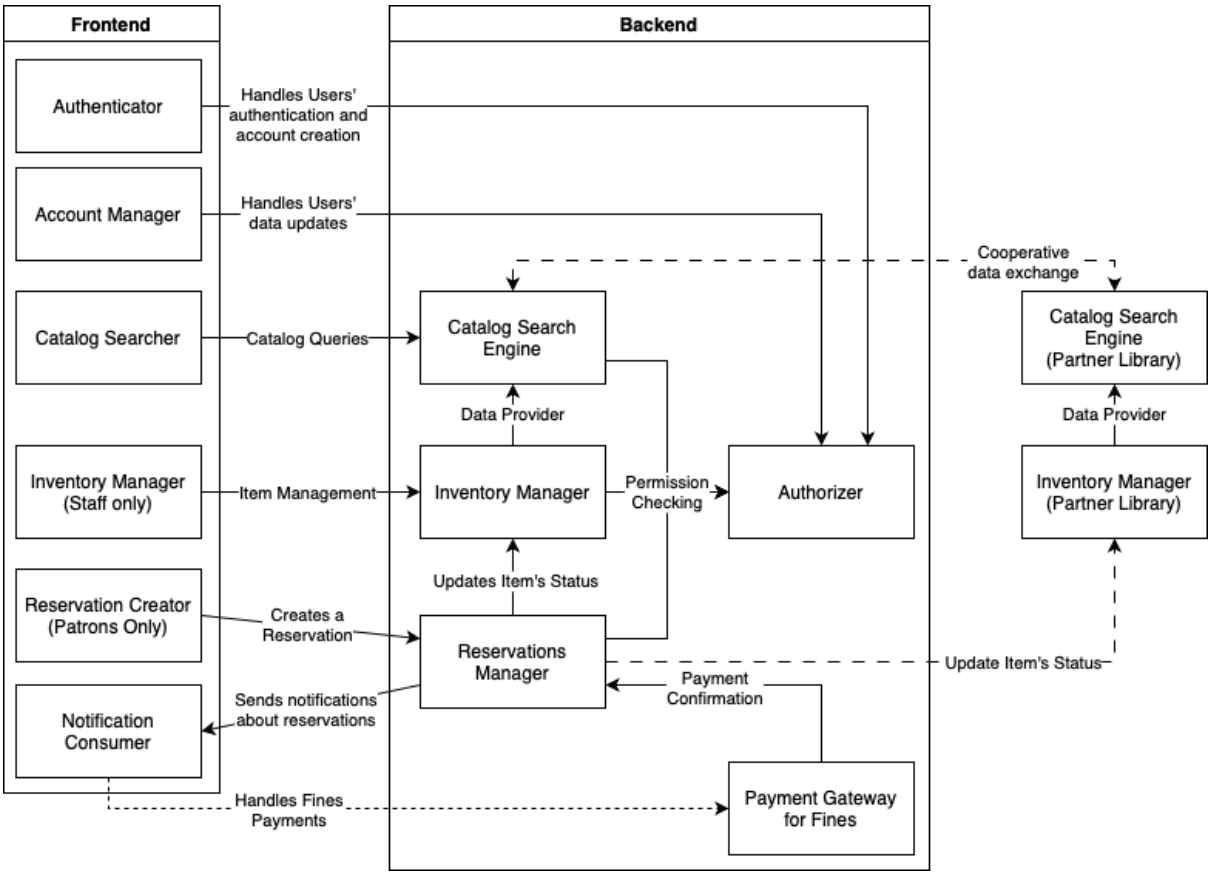

Figure 1 - Logical diagram containing our architectural proposal. Each box represents a component of the system. The arrows' direction represents the flow of data. The normal lines represent the interaction between components, dashed lines represent a flow that is optional, whereas the dotted arrow means an integration with a third-party service.

Architectural Components

An Overview of the Frontend

The frontend container in the diagram represents the components that interface with the staff and the patrons. This frontend system is a Web Application that runs on a browser. This decision ensures that the system is cross-platform, therefore it executes on machines with

very different characteristics, ranging from the connected terminals available *in loco* at the libraries to the patrons' mobile devices. The communication between the frontend and the backend is, usually, done through REST APIs.

Authenticator

This service is, at the business logic level, simple. It just handles users' authentication, both patrons and staff. This service only contacts the authorizer, in the backend, upon the start or shutdown of a users' session. The creation of patrons' accounts is only available for the staff members, whose accounts are created directly by the system administrator.

Account Manager

The account manager service is, like the authenticator, quite simple. It handles updates to the patron's account details. Whenever an update occurs, the service communicates with the authorizer, which reflects the change. Both the patrons and the staff have access to these features, however, the former must only be able to change its own data, whereas the latter may change data from the patrons.

Catalog Searcher

This frontend service essentially comprises the graphical interface available to the end-users, i.e., patrons and staff, to search through the catalog. The main concern here is that it may present different details according to the user role. It communicates with Catalog Search Engine in the backend and it is only available to users logged in in the system.

In the libraries there are search stations that only permit searching. To address this requirement a special kind of account is introduced. This kind of account is only obtainable by the system administrator. Given this, the same module that runs on the users' phones or in the staff member's computers is used. These machines are deployed with a special version of the system that only loads the authenticator and the catalog searcher module.

Inventory Manager Client

This is a frontend service only available for the staff. It allows the typical CRUD operations on top of the Inventory Manager in the backend, e.g., add or remove books, DVDs, etc. In the case of a digital asset, it is possible to add a blob of its contents that can later be distributed to the patrons who check it out.

The Inventory Manager also permits a staff member to verify the undergoing check outs on each item.

Reservation Creator

This service, as the name suggests, creates and visualizes reservations issued by a patron. A reservation consists of a set of items present in the catalog that become available to a patron for a determined period. The patron can schedule a reservation, this way it is possible to check out the items immediately or later.

It is integrated with the Catalog Searcher mentioned above, and it is only available to patrons. Its core functionalities are: item check out, and reservation – if a given item is unavailable. It communicates with the reservation manager in the backend to coordinate the past, ongoing and upcoming reservations.

Notification Consumer

This service, as the name suggests, sources events from the reservation in the backend. Then it displays the received notifications to the patrons. Since such notifications are related to reservations, this module also presents the details associated with the appropriate reservation. If a reservation contains an item whose media type is digital, the corresponding blob is provided.

Furthermore, it communicates directly with the payment gateway backend whenever a fine for late return is pending.

An Overview of the Backend

The backend presented in Fig. 1 is the large central component that processes all the patrons' and staff members' data, the inventory changes, the reservations of media, and the authorization of users. This comprises a set distinct of responsibilities, hence the importance of splitting them into the appropriate modules.

Authorizer

The Authorizer acts, mainly, as a middleware to the methods that require some sort of authorization to succeed. It keeps a view of all the users logged in the system and their corresponding permissions.

The permissions are attributed to a group, and then users are attributed to a group. Such an indirection permits the creation of complex permissions in a simpler manner.

Furthermore, the Authorizer also tracks updates to the users' data. Such composition is cohesive and exposes three interfaces: authentication, authorization and account details handling.

Reservations Manager

The reservation manager service, as the name suggests, is responsible for handling everything related to the patron's reservations. Upon the reception of a request, the service communicates with the Authorizer to assert that the author is logged in the system, hence fulfilling the request. If it succeeds, this service produces an event to update the appropriate item's status at the Inventory Manager service.

The data layer associated with this service has the characteristic of being append-only. It keeps track of all the past, ongoing and upcoming reservations. It also keeps track of all the updates made to a reservation, similar to a git history. Moreover, this system produces events to be consumed in the frontend when a given condition is met, e.g., a reservation finishes in 48 hours.

Whenever, there is a fine, for late media return, a payment request is forwarded to the Notification Consumer service in the frontend. Meanwhile the user is blocked from creating new reservations. When the payment confirmation arrives from the Payment Gateway, the user is unblocked from creating new reservations. In our architecture proposal, we assume that the payment gateway service is implemented and provided by a third party, hence simplifying the overall system design.

Inventory Manager

The inventory is a simple service which exposes a REST API. Beyond the addition or removal of items by the authorized staff members, this service also sources events from the reservation manager upon a reservation start (or ending) and updates the items' status accordingly. On a different domain, this service provides the data to the catalog search engine.

Catalog Search Engine

The Catalog Search Engine is a service that receives data from the Inventory Manager. It exposes a REST API to the Catalog Searcher in the frontend with all the desired search functionality.

Additional Considerations

Library Collaboration

Our proposal tries to address the requirement related to collaboration between libraries based on our personal interpretation. Collaboration, in this context, is defined as the possibility of a patron registered at library 'A' searching over an "unified" catalog. Moreover, we believe that it should be possible to make reservations of items available at any library's 'A' partners. This reservation must occur in an automated and transparent manner to the patron.

To address this, our architecture proposes the development of a distributed protocol that is able to provide cooperative data exchange, i.e., an eventually consistent catalog. Given this, it becomes crucial to track the origin library of each item, to ensure that the Reservation Manager resolves to the partner library's Inventory Manager and updates it correctly.

As a downside, the architecture in this area remains foggy, due to the fact that there is no more available information to go deeper.

Technological Stack

We have considered the software tools available to build this system. Our architecture is flexible enough to be implemented as separated services, i.e., microservices, or as a modular monolith. The advantage of the former is that each service can have a different technological stack, whereas the latter is easier to build and maintain due to its lower complexity, namely, concerning the communication between services.

In the frontend, as this is thought to be a Web application, the most obvious choice is to use Web technologies, such as Javascript together with a component library to achieve an intuitive and consistent UI/UX.

On the backend side, there are more options. The Catalog Search Engine can be powered by an Elastic or Solr cluster. The remaining services can be written, virtually, in any programming language; the key here is to define the interfaces for communication between them.

Moreover, developers are aware of the existence of numerous frameworks that ease the implementation process.

Payment Gateway

The last remark on this topic is the payment gateway. In our architecture, we assume that this is a third party service due to its complexity and challenges.

System Usage Scenarios

From our architecture proposal, we can visualize a couple of scenarios of usage of our system that are relevant to describe. This way, we will present two that could help better understand the functionality of our system's architecture.

Scenario 1: Physical or Digital Item Checkout with Notification

In this scenario, a patron utilizes the Catalog Searcher to reserve a book, prompting the Reservation Manager to update the item's status. Upon processing, the Notification Consumer sends a notification to the patron, indicating immediate online access to the e-book and the option for in-person pickup of a physical copy. The patron interacts with the notification, accessing the e-book instantly online and opting to collect the physical copy. Subsequently, the patron visits the designated local library to collect the physical item, successfully completing the checkout process.

User Reserves Item Online:

- A Patron searches for a book using the Catalog Searcher and proceeds to reserve it.
- The Reservation Manager processes the request and updates the status of the item.

Notification for Item Availability:

- Notification Consumer sends a notification to the Patron, informing him that the e-book is immediately available for access online
- And, if chosen, that could be picked up in person in a library with that item available.

User Interaction:

- The patron clicks on the notification to view details.

- The digital item is accessible instantly, and the patron can start reading it online.
- Also, if the user chooses to do so, a physical copy of the item reserved can be collected in person.
- The user chooses to also get a physical copy of the book.

In Person Pickup:
- The patron visits the local library indicated in the app to collect the physical item reserved.
- The patron successfully completes the checkout process for the physical item.

Scenario 2: Fine Payment Confirmation and Reservation Access

In this scenario, a patron returns past-due items to the library, triggering the calculation of fines by the Reservation Manager. The Patron finds their reservation access blocked, upon notification of the fine. Settling the fine, they initiate payment through the integrated gateway. Upon successful confirmation of payment, the Patron receives a notification, prompting the Reservation Manager to clear the fine and restore reservation access, enabling the Patron to make new reservations.

User Returns Past-Due Items:
- A Patron returns past-due items to the library.

Fine Calculation and Notification:
- The Reservation Manager calculates the fine for late return and sends a notification to the Patron.

Blocked Reservation Access:
- The Patron tries to make new reservations but is blocked due to the pending fine.

Payment Initiation:
- The Patron decides to pay the fine using the integrated payment gateway.

Confirmation Notification:
- Payment Gateway confirms the payment, and Notification Consumer sends a confirmation to the Patron.

Fine Clearance:
- The Reservation Manager receives confirmation of the payment and clears the fine associated with the Patron's account.

Reservation Access Restored:
- The Patron can now make new reservations as the fine restriction is lifted.