TicketBoss Architecture

Software Systems Architecture

Group 7 – Class 3

Master in Informatics and Computing Engineering

Dinis R. dos Santos Bessa de Sousa	up202006303@fe.up.pt
João Rola Reis	up202007227@fe.up.pt
Pedro Manuel da Silva Gomes	up202006322@fe.up.pt
Ricardo André Araújo de Matos	up202007962@fe.up.pt

1. Introduction

In this report, we will describe Ticket Boss' architecture. Ticket Boss is a system designed to handle ticket sales for various entertainment events.

TicketBoss system should be able to accomplish the following, according to the requirements given:

- It displays multiple entertainment events that a user can purchase tickets for
- When a user selects an event, it shows seats available. A user can select seats, and see prices for
- those seats.
- When a user selects seats, the seats are held for ten minutes. If the user does not complete the purchase within ten minutes, the seats are released for someone else to buy.
- Purchasing tickets involves the usual stuff: user enters credit card information, billing address, etc. TicketBoss emails a confirmation to the user. (Design note: there are applications available that handle credit card payments.)
- Multiple users can access the same entertainment event at the same time, but of course, only one person can hold a particular seat.
- A user can hold up to 10 seats at a time

We will first look at the architecture of the system in its various views, analyze which architecture patterns are present, and then dive into the quality attributes of the system.

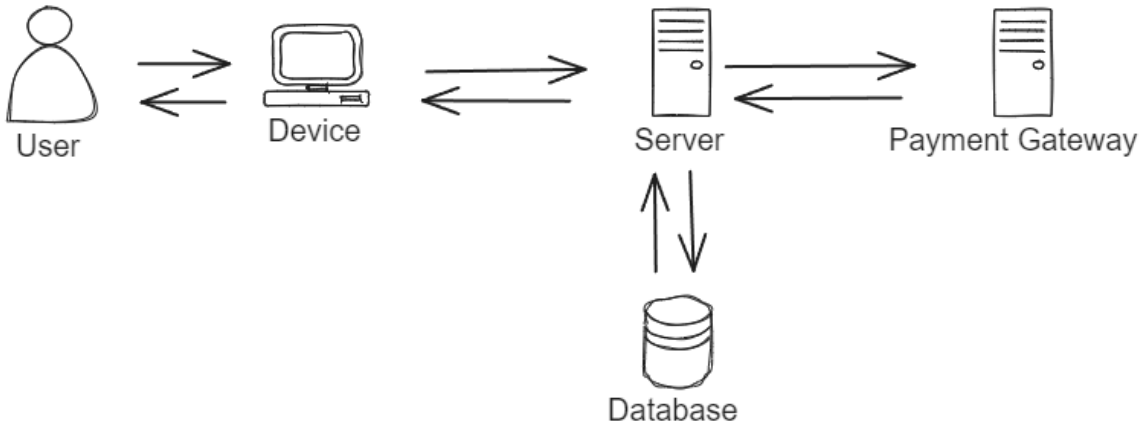
2. Architecture

We will design the architecture of TicketBoss, using different views and their respective diagrams to document the architecture.

2.1 Physical View

The Physical View of the TicketBoss is fairly simple, following, on a high level, a Client-Server architecture.

The arrows represent the communication between the different parts of the system.



User: The one that uses the app. Interacts directly with frontend of the application, using its personal device.

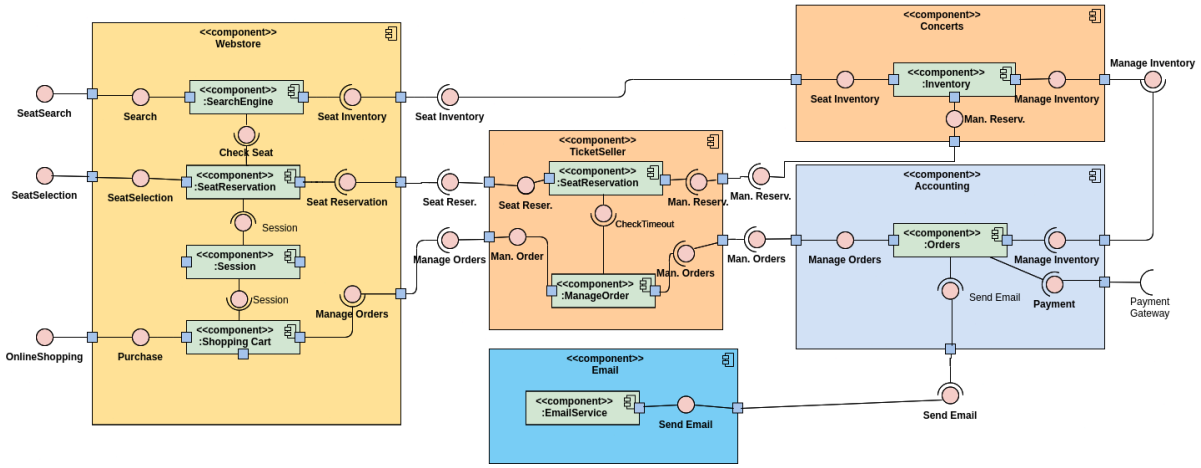
Device: Personal device of the user, serves as interface and intermediary between the user and the server/application.

Server: The server contains the main components of the application logic, from managing orders to reserve seats. The server is also responsible for interacting with the database for getting event information and seat availability and with the payment gateway, a third party application to handle payments. The server is also responsible for sending the email with the confirmation of the purchase and the ticket to the user.

Database: Saves information about the events and the respective seat availability. Also saves information about the reservations.

Payment Gateway: Third Party service that handles the payment.

2.2 Logical View



Concerts Service

- Responsibilities:
    - Manages the concert ticket inventory, overseeing availability, reservations, and purchases.
- Interfaces Implemented:
    - Seat Inventory: Provides information about available tickets.
    - Manage Inventory: Marks concert tickets as purchased.
    - Manage Reservations: Handles the reservation status of tickets.

Accounting Service

- Responsibilities:
    - Processes orders for concert tickets.
    - Manages inventory by marking tickets in shopping carts as purchased.
    - Utilizes an email service to send confirmation emails to clients.
    - Facilitates payment processing through an external payment gateway.
- Interfaces Required:
    - Manage Inventory: Used for managing ticket inventory.
    - Payment: Interface for processing ticket payments.
    - Send Email: Used for sending confirmation emails to clients.

Ticket Seller Service

- Responsibilities:
    - Facilitates ticket reservations and purchases.
- Interfaces Required:
    - Manage Reservations: Implemented by the Concerts service to handle ticket reservations.
    - Manage Orders: Implemented by the Accounting service for buying tickets.
- Components:
    - Seat Reservation: Manages ticket reservations and expiration.
    - ManageOrder: Processes shopping cart checkout.

Webstore Service

- Responsibilities:
    - Provides user interfaces for searching, reserving and buying tickets.
- Interfaces Implemented:
    - SeatSearch: Utilizes the Seat Inventory interface from the Concerts service.
    - SeatSelection: Utilizes the SeatReservation interface from the Ticket Seller service.
    - OnlineShopping: Utilizes the Manage Orders interface from the Ticket Seller service.
    - Session: Implements session management for maintaining user states across different sessions (i.e., maintaining seat reservations after disconnecting and reestablishing a connection).

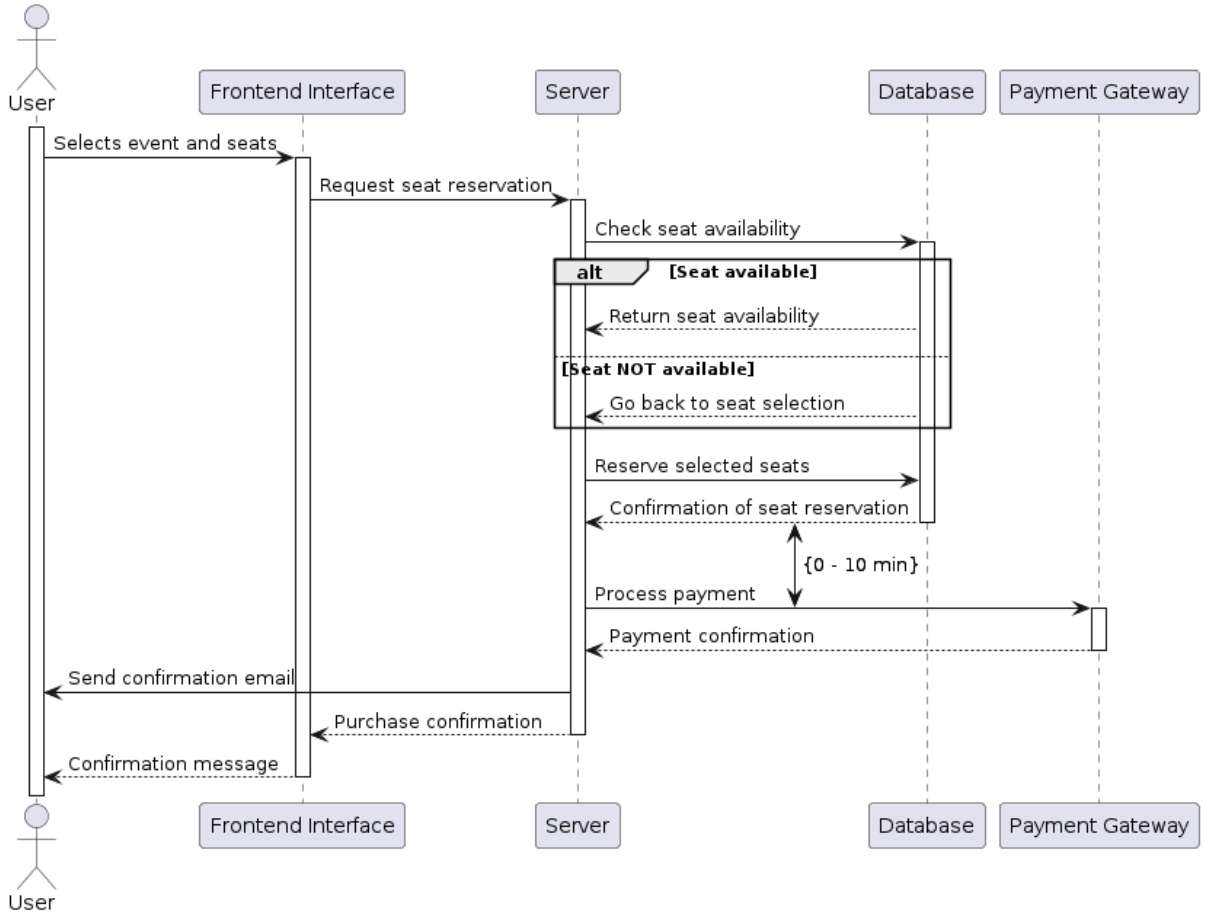
Email Service

- Responsibilities:
    - Implements the Send Email interface required by the Accounting service.
    - Sends emails to clients.

2.3 Process View

In the Process View, we have described the steps of reserving and buying a ticket. We can see the interaction between the components and their order.

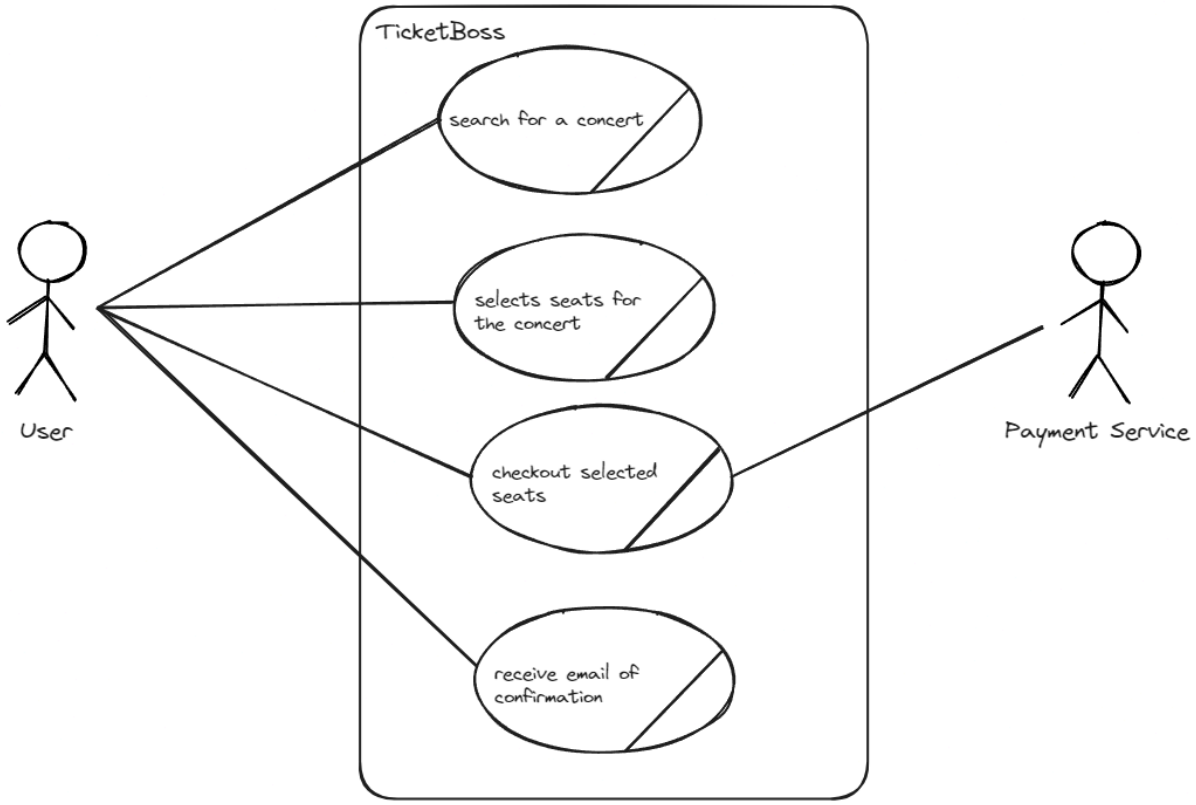
Below is represented the Process Diagram for communication of the different parts (physical) of the system, when the user is selecting and purchasing a ticket.



The participants of the Process Diagram correspond to the physical components of the system. When the user selects a seat for a specific event, the different participants exchange information to allow the registration of the event reservation, payment, and when all is done, send the confirmation and the tickets to the user.

2.4 Use-case View

Use case 1:



Title: Select and Purchase Seats for Entertainment Event

Description: This use case involves a user selecting seats for an entertainment event and completing the purchase process.

Actors:

- User
- Payment Service

Preconditions:

- Entertainment event is displayed with available seats.
- User has selected an entertainment event.

Postconditions:

- Seats are successfully reserved and purchased.
- Confirmation email is sent to the user.
- Selected seats are no longer available for purchase by other users.
- If the purchase process is not completed within ten minutes, the held seats are released.

Path:

Primary Path:

1. User searches and selects an entertainment event from the list of available events.
2. System displays the available seats for the selected event.
3. User selects desired seats, up to a maximum of 10 seats.

4. System holds the selected seats for ten minutes.
5. User confirms the selection and proceeds to checkout.
6. User initiates payment through the Payment Service.
7. User enters credit card information, billing address, and other required details.
8. Payment Service processes the payment and notifies the system of the successful transaction.
9. Confirmation email is sent to the user.

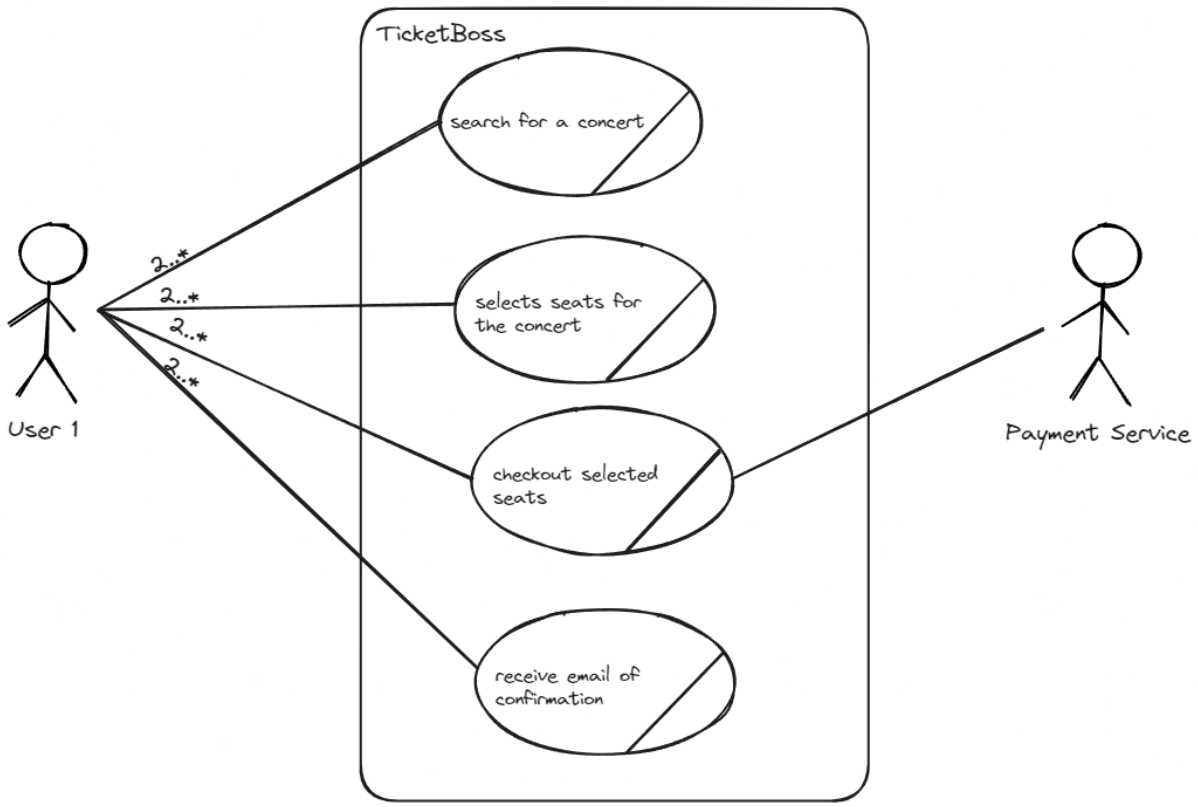
Exception Path:

1. User searches and selects an entertainment event from the list of available events.
2. System displays the available seats for the selected event.
3. User selects desired seats, up to a maximum of 10 seats.
4. System holds the selected seats for ten minutes.
5. User doesn't complete the payment process within 10 minutes
6. System displays an error message informing the user that the reservation of the seats has expired.

Priority and Frequency of Use:

- Priority: High
- Frequency of Use: Frequent

Use case 2:



Title: Multiple Users Select Seats for the Same Event Simultaneously

Description: This use case involves two or more users selecting seats for the same event within ten minutes of each other without encountering conflicts.

Actors:
- User 1
- User 2
- Payment Service

Preconditions:
- Entertainment event is displayed with available seats
- Both User 1 and User 2 have selected an entertainment event.

Postconditions:

- Both User 1 and User 2 successfully reserve and purchase their selected seats.
- Confirmation emails are sent to both users.
- The selected seats are no longer available for purchase by other users.
- If the purchase process for either user is not completed within ten minutes, the held seats for that user are released.

Path:
Primary Path:
1. Two or more users access the system and search for the same entertainment event.
2. System displays the available seats for the selected event to all users.
3. Users independently select desired seats, up to a maximum of 10 seats each.
4. System holds the selected seats for each user.
5. Users confirm their selections and proceed to checkout.
6. Users initiate payment through the Payment Service.
7. Users complete the checkout process, entering necessary details through the Payment Service.
8. Payment Service processes the payment for each user's selected seats.
9. System reserves the selected seats and completes the purchase for each user.
10. Confirmation emails are sent to each user.

Exception Path:
1. Two or more users access the system and search for the same entertainment event.
2. System displays the available seats for the selected event to all users.
3. Users independently select desired seats, up to a maximum of 10 seats each.
4. System holds the selected seats for each user.
5. Users confirm their selections and proceed to checkout.
6. One or more users take more than 10 minutes to complete the checkout process.
7. System detects that the checkout process for a user has exceeded the 10-minute limit.
8. System releases the held seats for the user who exceeded the time limit.

Priority and Frequency of Use:

- Priority: High
- Frequency of Use: Frequent

3. Architecture Patterns

In this section, we will describe the patterns present in the TicketBoss Architecture and the characteristics and advantages of the architectures.

3.1 Client-Server Architecture

The system is a classic example of a Client-Server Architecture. The frontend interface (client) interacts with the backend server (server) to perform various operations.

3.2 Event-Driven Architecture

TicketBoss utilizes event-driven architecture for communication between components. Events such as seat reservation, payment completion, and other events, trigger actions within the system. Utilized to ensure loose coupling between the components.

3.3 Shared Repository

The inventory component functions as a centralized repository, housing comprehensive data concerning concert tickets. Multiple components rely on this repository for both retrieval and modification of information.

The SearchEngine component utilizes this repository to present users with a comprehensive view of available tickets. Meanwhile, the SeatReservation component facilitates the seamless transition of tickets between available and reserved statuses. Lastly, the ManageOrder component is responsible for updating ticket statuses to reflect purchases.

4. Quality Attributes

4.1 Availability

- **Meaning:**

    - The system needs to maintain a high level of availability to ensure customers can purchase tickets at any time. Therefore, it should be capable of managing thousands of concurrent connections. If this quality attribute fails, we risk losing customers and, consequently, revenue.
- **Acceptable level:**
    - The system should manage connections in accordance with the scale of events reach/dimension. If the system experiences highly popular events, it must be prepared to handle the increased connections resulting from such events.
- **Importance:**
    - As mentioned above, failing to provide an available service can have a negative impact on customer satisfaction and revenue. As such, this quality attribute is of extreme importance for TicketBoss.

4.2 Security

- **Meaning:**
    - Maintaining a secure system entails ensuring it's impervious to malicious attacks from individuals with ill intent, whether they aim to pilfer data/products or disrupt services. Security, therefore, encompasses the system's ability to withstand such threats.
- **Acceptable level:**
    - The system must effectively handle all data transactions with utmost care, especially since critical information such as credit card details or tickets should never fall into the hands of attackers. Additionally, it's crucial to establish a resilient system against denial of service attacks, as they can severely impact event revenue and customer experience.
- **Importance:**
    - Security is paramount for TicketBoss as it deals with sensitive customer information, including personal details and payment data. A breach in security could lead to severe consequences, including financial losses, damage to reputation, and legal repercussions. Therefore, ensuring robust security measures is essential to maintain trust and confidence among customers and stakeholders.

4.3 Reliability

- **Meaning:**
    - Reliability refers to the system's ability to perform consistently and predictably under normal and abnormal conditions. It involves minimizing downtime, errors, and failures to ensure uninterrupted service delivery.
- **Acceptable level:**

    - The system should be highly reliable. It must incorporate fault-tolerant mechanisms, such as redundancy, failover, and graceful degradation, to mitigate the impact of hardware or software failures.
- **Importance:**
    - Reliability is paramount because we cannot afford costly errors. Picture a scenario where two separate servers inadvertently sell the same ticket to two different customers. Such mistakes not only result in financial losses but also erode the trust clients have in the system. Therefore, ensuring a reliable system is of utmost importance to uphold customer trust and prevent financial setbacks.

5. Conclusion

In summary, TicketBoss has been designed to provide a seamless ticket purchasing experience for users attending entertainment events. Its architecture incorporates various views and patterns to ensure efficient operation, while its focus on quality attributes like availability, security, and reliability underscores its commitment to customer satisfaction and data protection. With a solid architectural foundation in place, TicketBoss is poised to meet the demands of its users while maintaining a high standard of service delivery and system integrity.