Universidad Distrital Francisco José de Caldas

Dept. of Computer Engineering

Systems Engineerings

Workshop 2
Software enginnering seminar

Carlos Andres Abella
Daniel Felipe Paez

Leidy Marcela Morales

Supervisor: Carlos Andrés Sierra

Bogotá D.C.

October 19, 2025

Contents

1 Introduction

Event management has become increasingly complex due to the growing demand for conferences, workshops, concerts, and professional networking activities. Traditional event planning methods often involve manual processes such as paper registrations, physical ticket sales, and fragmented communication, which are inefficient and error-prone. Digital transformation offers the opportunity to create platforms that automate registration, simplify payment processes, and improve the overall experience for both organizers and attendees.
This project proposes the design of an Event Management Application that allows organizers to publish and manage events while attendees can easily register, pay, and receive timely updates. By applying software engineering practices, we aim to define a solid business model, user requirements, and initial design artifacts that will guide the development process.

See the workshop website for more instructions:
`https://github.com/Andres0127/Software-Engineering-Seminar-submission-of-papers.git`

1.1 Problem Description

Currently, many event organizers face challenges in managing registrations, tracking attendees, and processing payments efficiently. Attendees, on the other hand, often struggle with limited access to event information, lack of reminders, and insecure or slow payment options.

These difficulties result in the following.

- Lost revenue due to inefficient registration systems.
- Lost revenue due to inefficient registration systems.
- Poor communication between organizers and attendees.
- Limited visibility of events for potential participants.
- Increased operational costs and errors for organizers.

Therefore, a centralized, digital and user-friendly platform is needed that can streamline the end-to-end event management process.

1.2 General Objective

To design and define a digital event management platform that streamlines the organization and participation process by providing organizers with efficient tools for event creation and tracking, and offering attendees a simple, secure, and user-friendly system for registration, payment, and notifications.

1.3 System analysis

An event management system can be divided into several parts, or subsystems, that work together to meet the overall objectives of successful event planning, management, and execution. The main parts are:

- Event Management Subsystem: Includes event creation, editing, publishing, and management. Details such as dates, locations, ticket types, capacity, and logistics are defined here.
- Registration and Sales Subsystem: Handles participant registration, ticket sales, and secure payment processing, ensuring digital ticket issuance and validation.
- Communication and Notifications Subsystem: Manages contact with organizers and attendees through emails, SMS messages, alerts, and real-time updates.
- Monitoring and Reporting Subsystem: Provides dashboards and reports for real-time tracking of sales, capacity, revenue, and post-event feedback for analysis and continuous improvement.
- Integration Subsystem: Allows the platform to connect with external services such as payment gateways, CRM systems, social networks, and other digital tools to expand functionality.

Each subsystem is an autonomous yet interrelated component, forming a complex system that interacts with its environment through inputs (user data, requests, payments) and outputs (tickets, notifications, reports). This division facilitates modular management, scalability, and resilience to changes or failures.

This structure allows addressing critical points in the system, especially during high-demand scenarios such as rapid ticket sales (flash sales). In these cases, a small error in the service fee or an incorrect configuration of event capacity can quickly escalate: a flash sale with poorly set capacity can lead to overbooking, multiple claims, and system saturation, seriously affecting the platform's reputation and financial stability.

Furthermore, the architecture must guarantee payment security, the integrity of event and user data, and operational efficiency at all times. These conditions reflect the chaotic and dynamic nature of the real-world environment, where small failures or incorrect decisions can have significant consequences if not properly controlled through validation, monitoring, and automated recovery mechanisms.

2 Model canvas

The Business Canvas is a fundamental strategic tool that allows us to fully and concisely visualize the nine fundamental pillars of a business. In the context of this event management platform, the Canvas helps us clearly define the value proposition offered to users, identify target customer segments, establish key relationships with these customers, and determine the channels of interaction. It also details the activities and resources essential to operations, the strategic partners that contribute to the ecosystem, and, crucially, the cost structure and revenue streams that support the project's economic viability. This holistic view ensures that the technological architecture is directly aligned with the business objectives and strategy.

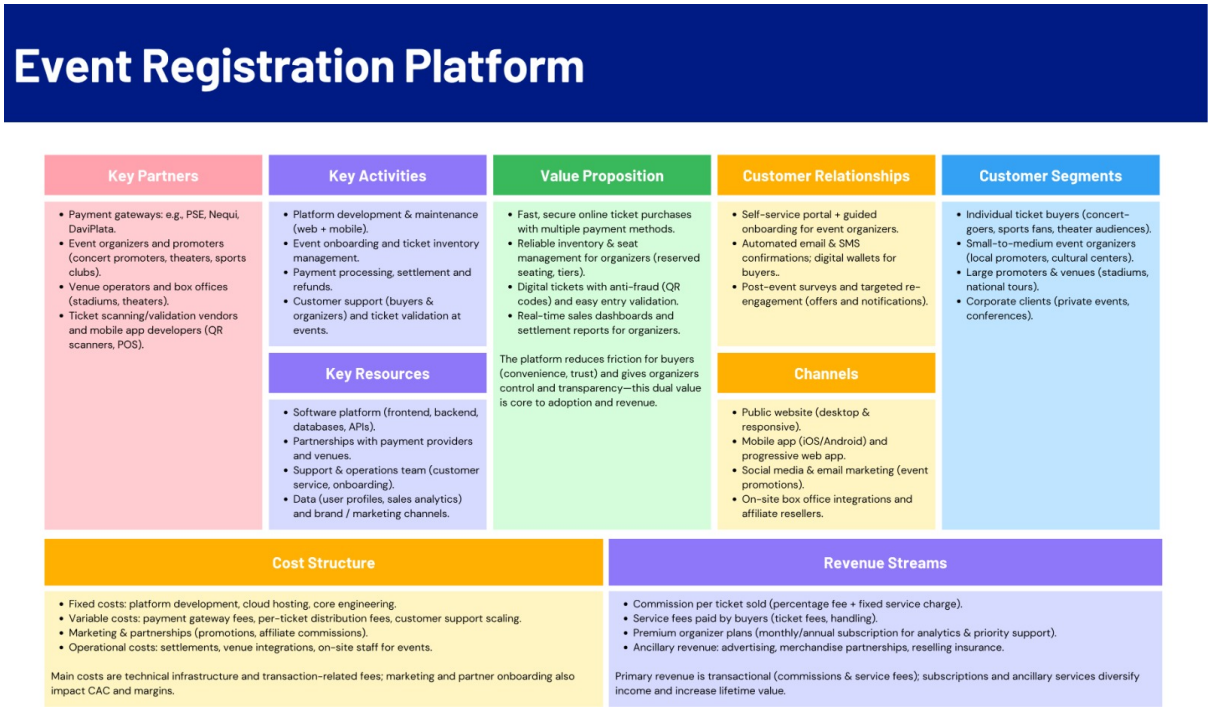

Figure 1: Business model canvas

3 User stories

User Stories are an agile tool that allows us to understand and document system functionalities from the perspective of the different actors who will interact with it. Each story is formulated in a simple and concise manner, following the format "As [role], I want [action], so that [benefit]." This allows us to capture the specific needs of buyers, sellers, administrators, and other roles, ensuring that development focuses on delivering real value to each type of user. Additionally, each story includes detailed acceptance

criteria that define when a feature is considered complete and correct, facilitating its validation. This user-centered approach is vital to building an intuitive platform that truly solves its users' problems and desires.

3.1 Role: Event Organizer

Title: Create and Publish a New Event	**Priority:** High	**Estimate:** Large

User Story:
As an **Event Organizer**, I want to **create and publish a new event with detailed information, seating types, and prices**, so that **I can make it available for ticket buyers on the platform**.

Acceptance Criteria:
Given I am logged in as an "Event Organizer",
When I navigate to the "Create Event" page, fill in all mandatory fields, define at least two ticket tiers, and click the "Publish" button,
Then the event is assigned a unique URL, becomes publicly visible, and its tickets are available for purchase.

Title: View Real-Time Sales Dashboard	**Priority:** High	**Estimate:** Medium

User Story:
As an **Event Organizer**, I want to **view a real-time sales dashboard for my event**, so that **I can track ticket sales and revenue**.

Acceptance Criteria:
Given I am logged in as an "Event Organizer" and am viewing one of my active events,
When I click on the "Dashboard" or "Reports" tab,
Then the system displays the total number of tickets sold per tier and the total gross revenue, updated in near real-time.

3.2 Role: Ticket Buyer

Title: Purchase Event Ticket Securely	**Priority:** High	**Estimate:** Large

User Story:
As a **Ticket Buyer**, I want to **select tickets for an event and complete the purchase securely**, so that **I can receive a valid digital ticket to attend**.

Acceptance Criteria:
Given I am viewing the page for an event with available tickets,
When I select my tickets, proceed to checkout, and confirm the mock payment,
Then I am redirected to a confirmation page and receive an email with the digital tickets as a PDF with a unique QR code.

Title: Browse and Discover Events	**Priority:** High	**Estimate:** Medium

User Story:
As a **Ticket Buyer**, I want to **search for events by name, date, or category**, so that **I can easily find events I'm interested in**.

Acceptance Criteria:
Given I am on the homepage,
When I use the search bar or apply a category filter,
Then the search results page displays only the events that match my criteria.

3.3 Role: Platform Admin

Title: Suspend User Account	**Priority:** Medium	**Estimate:** Medium

User Story:
As a **Platform Admin**, I want to **suspend an Event Organizer's account**, so that **I can prevent fraudulent or harmful activity on the platform**.

Acceptance Criteria:
Given I am logged in as a "Platform Admin" and am viewing a specific organizer's profile,
When I click the "Suspend Account" button and confirm my action,
Then the organizer's account is marked as suspended, they can no longer log in, and all their published events are unpublished.

4 User Story Mapping

User Story Mapping is a software engineering technique used to **visualize, organize, and prioritize** product functionalities from the user's perspective. Its main goal is to understand how different user roles interact with the system and what steps they follow to achieve their objectives, dividing this journey into *epics (big activities)*, *features (specific actions)*, and *releases (incremental versions)* of the product.

In our project, which consists of an **event ticketing and management platform**, we used this approach to represent the entire system flow—from event creation to ticket purchase and monitoring.

We identified three main **user personas**:

- **Event Organizer:** creates and publishes events, defines ticket prices, and tracks sales.
- **Ticket Buyer:** searches for events, selects, and purchases tickets securely.
- **Platform Admin:** supervises the platform, manages accounts, and prevents fraud.

From these personas, we defined the following main **epics**:

1. **Account Management:** sign up, login, and profile management.
2. **Event Creation & Publishing:** create events, define ticket tiers, and publish them.
3. **Event Discovery:** search and filter available events.
4. **Ticket Selection & Purchase:** select ticket type and quantity, checkout, and payment.
5. **Monitoring & Administration:** sales dashboard and account control.

Each epic was broken down into **features** with detailed functionalities (for example, under "Ticket Purchase" we included *Select Tickets*, *Checkout*, and *Payment*), and every feature included enhancements distributed by **releases** (MVP, improvements, and future versions).

The result was a clear and organized **visual User Story Map** showing how the product evolves, which features are prioritized, and how each user gains value throughout the process. This mapping helped us plan development iterations efficiently and maintain a user-centered vision of the system's evolution.

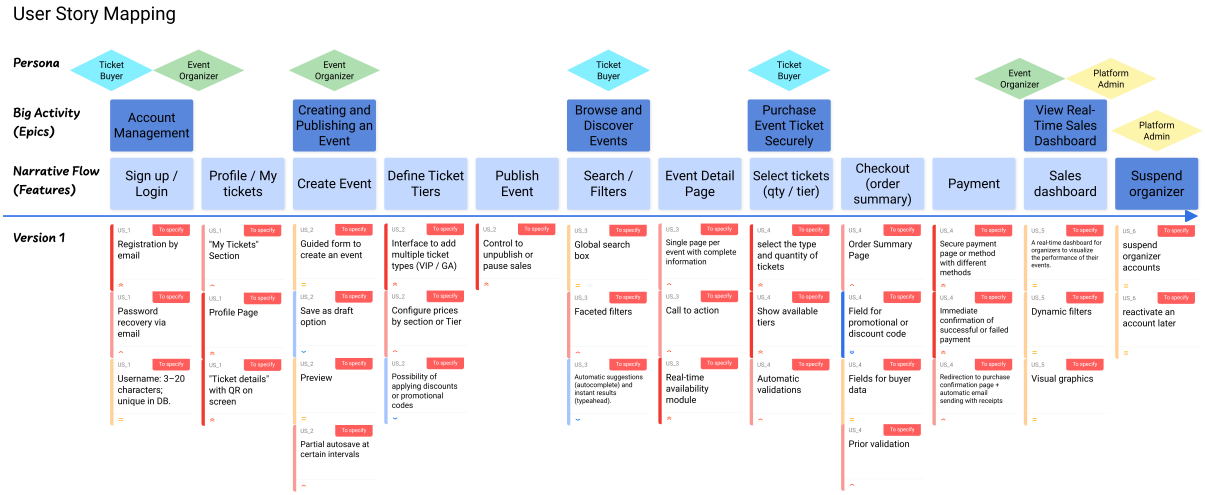

Figure 2: User Story Mapping

5 CRC Cards

In the context of the Event Registration Platform, the CRC (Class–Responsibility–Collaborator) cards present an early object-oriented view of the system's internal design. Each CRC card identifies a core class, outlining its main responsibilities and its collaboration with other entities and services within the platform.

This object-oriented representation helps define clear roles, behaviors, and relationships between components, establishing a direct link between system requirements and future class diagram implementation.

For this project, the key domain classes include both **core entities** and **user roles**:

- **Event:** Represents each published event, maintaining essential details such as title, location, date, capacity, and ticket categories. It is responsible for managing availability, lifecycle status, and links between organizers and attendees.
- **Ticket:** Models each purchased entry for an event, storing attributes such as price, QR code, and seat or tier information. It also manages validation, digital generation (PDF/QR), and payment association.
- **Order:** Groups multiple tickets under a single transaction, controlling totals, payment verification, and purchase confirmation.
- **Payment:** Handles the financial transactions of each order, processing, confirming, or refunding payments through an integrated gateway.
- **Location:** Defines the physical or virtual space where an event takes place, specifying its name, address, and capacity.
- **EventOrganizer, TicketBuyer, and PlatformAdmin:** Represent specialized user roles derived from a common *User* class. Organizers create and manage events; buyers purchase and receive tickets; administrators supervise, monitor, and ensure system compliance.
- **NotificationService:** Supports automated communication across the platform by sending digital tickets, purchase confirmations, and system reports to users.

These CRC cards collectively describe how classes collaborate to achieve the platform's main processes—event management, ticket sales, payment processing, and user communication—forming a conceptual foundation for the class diagrams and implementation detailed in the following sections.

5.1 Class: Event

Class: Event	
Responsibility	**Collaborator**
• Maintain event information (title, date, location, description). • Define and manage ticket categories (VIP, General, etc.). • Track total capacity, sold tickets, and availability in real time. • Handle event lifecycle: Draft, Published, Sold Out, Cancelled. • Provide performance data to generate reports. • Notify organizers and buyers about event updates.	• EventOrganizer • Ticket • Location • NotificationService • Report

Table 1: CRC Card — Event Class

5.2 Class: Ticket

Class: Ticket	
Responsibility	**Collaborator**
• Represent a purchased entry to a specific event. • Store ticket data (seat/tier, price, QR code, status). • Generate and deliver digital tickets (PDF/QR). • Validate ticket authenticity at event check-in. • Record and update payment confirmation status. • Notify buyer upon successful purchase or cancellation.	• TicketBuyer • Event • Order • Payment • NotificationService

Table 2: CRC Card — Ticket Class

5.3 Class: User

Class: User (Abstract)	
Responsibility	**Collaborator**
• Authenticate and manage access to the platform. • Maintain personal profile and credentials. • Interact with events, orders, and notifications depending on role. • Receive system updates and account notifications.	• NotificationService • Event • Order

Table 3: CRC Card — Abstract User Class

6 Class diagram

The Class Diagram of the Event Registration Platform represents the structural backbone of the system, showing how the main entities, user roles, and services interact under an object-oriented architecture. This diagram is derived directly from the CRC (Class–Responsibility–Collaborator) analysis, translating identified responsibilities and collaborations into well-defined classes, attributes, methods, and relationships.

The diagram emphasizes the application of core object-oriented principles such as encapsulation, inheritance, and composition. It illustrates both the hierarchy of user roles and the relationships between the main business entities—*Event*, *Ticket*, *Order*, *Payment*, and *Location*—as well as the supporting service classes like *NotificationService*. Each class encapsulates its specific responsibilities, promoting modularity, maintainability, and clear separation of concerns within the system.

In this model:

- **User hierarchy:** The abstract *User* class is extended by specialized roles—*EventOrganizer*, *TicketBuyer*, and *PlatformAdmin*—each defining unique behaviors aligned with their functions in the platform.
- **Core entities:** *Event*, *Ticket*, *Order*, and *Payment* model the essential business processes of event creation, ticket sales, and transaction handling.
- **Associations and compositions:** Relationships between classes are clearly defined, showing ownership and multiplicity—for instance, one *Order* may include multiple *Tickets*, while each *Ticket* is associated with a single *Payment*.
- **Supporting services:** The *NotificationService* facilitates automated communication across user roles, enhancing the platform's interactivity and reliability.

Overall, the class diagram provides a comprehensive blueprint for system implementation. It formalizes the structure of the platform, ensuring that each class and interaction aligns with the functional and non-functional requirements defined in the previous stages of design.

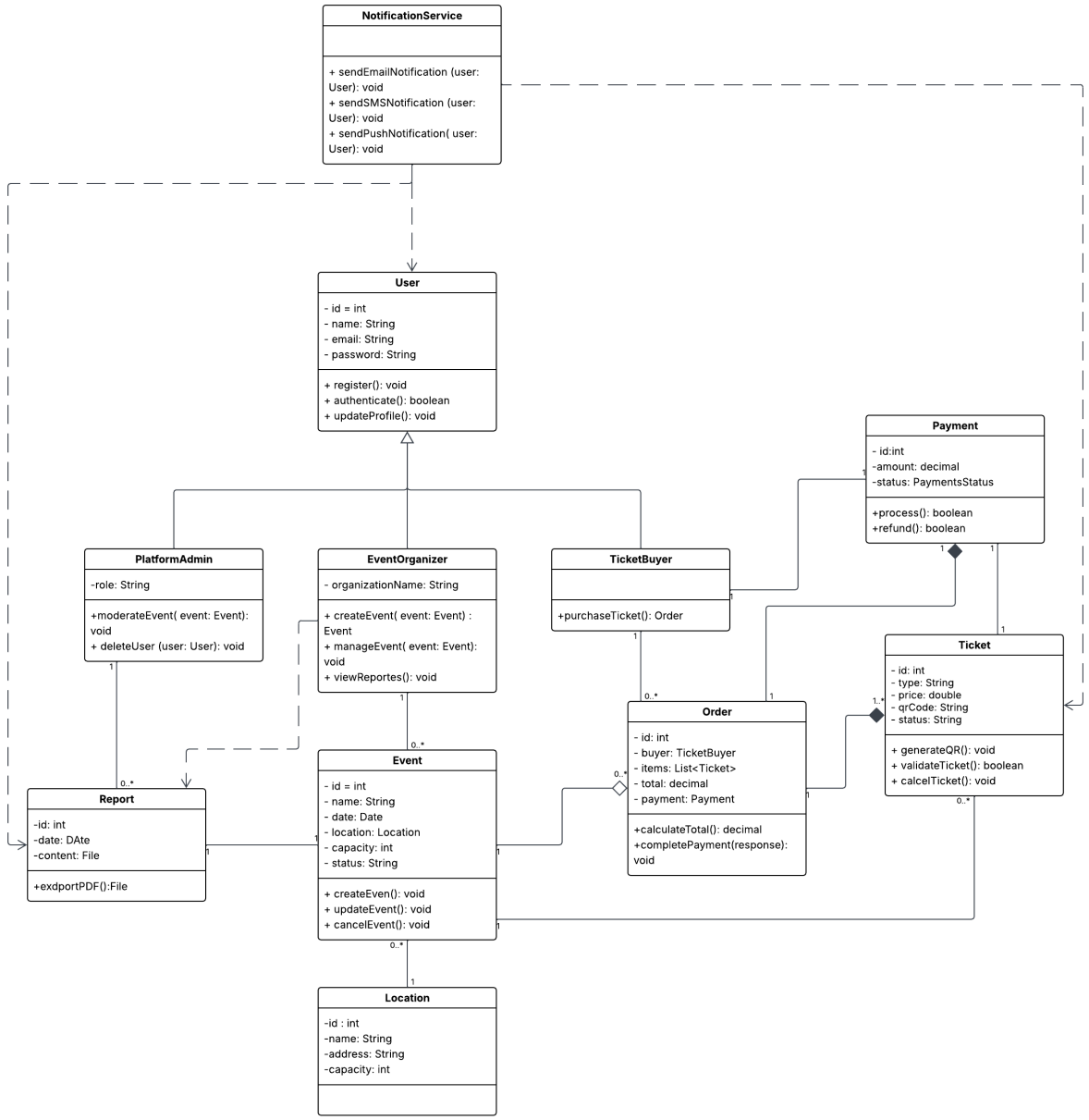

Figure 3: Class diagram

7 Architecture Diagram

The event management system's software architecture is designed with a modular, service-oriented approach, ensuring scalability, ease of maintenance, and integration with external platforms. The system allows users to create, manage, and attend events, in addition to offering ticketing, payment processing, and notification functionality.

The architecture follows a three-tier model that separates the presentation layer, the application layer, and the data layer.

The presentation layer represents the user interface, accessible through a web or mobile application. It allows users to browse events, register, and purchase tickets via RESTful APIs.

The application layer includes the core business logic and service modules, such as PaymentService, NotificationService, and ReportService. Each service encapsulates specific functionality and communicates internally via standardized API calls.

The data layer manages persistent storage using a relational database system, ensuring data consistency for entities such as users, events, and tickets.

Additionally, the system integrates external APIs for payments, notifications, and analytics through the IntegrationAPI component. This modular structure not only improves maintainability but also facilitates cloud deployment and container orchestration using Docker technologies.

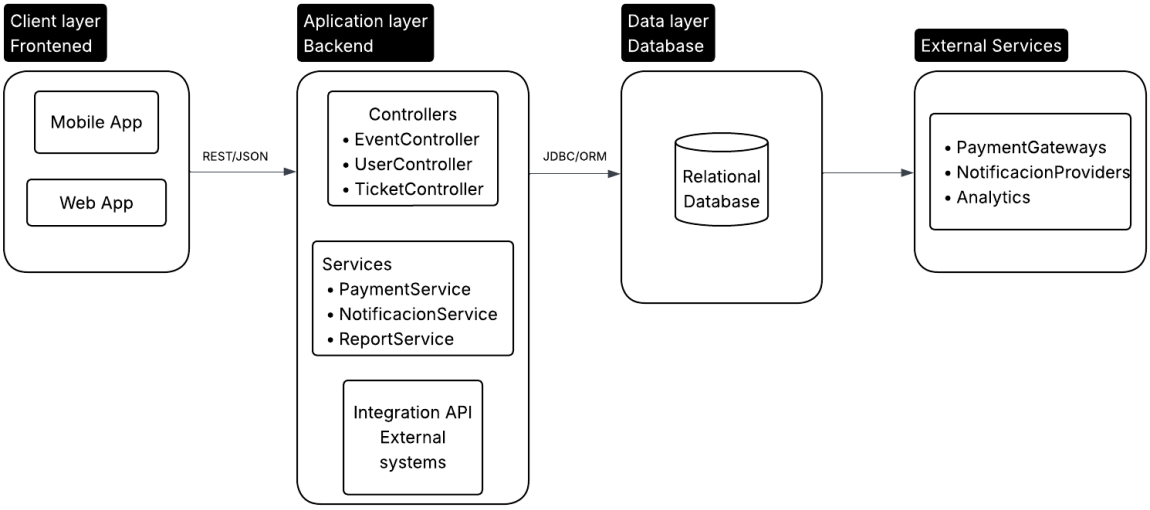

Figure 4: Architecture diagram

The deployment architecture defines how the system's components are physically distributed and executed across the infrastructure. In this project, deployment is implemented using Docker containers, which allow each module of the system to run in an isolated and reproducible environment. This approach simplifies development, testing, and scaling across different platforms such as local servers, cloud instances, or Kubernetes clusters.

Each service — for example, the API server, database, and auxiliary modules — is packaged as a separate Docker image defined by a Dockerfile. This ensures consistent behavior and environment configuration, regardless of where the system is deployed.

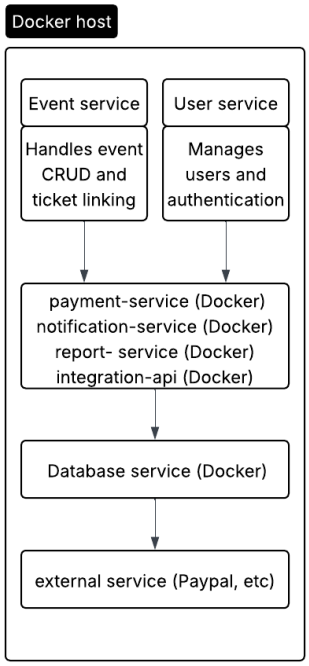

Figure 5: Deployment diagram

8 BPMN Diagram: Ticket Purchase Process

8.1 Overview

The BPMN diagram represents the primary business process of the event ticketing platform: the **Ticket Purchase Process**. This process was modeled from the perspective of the three main human actors involved in the system: the **Ticket Buyer**, the **Event Organizer**, and the **Payment Gateway**. Each lane captures the specific actions performed by these participants without including automated system validations or internal system logic. The focus is placed solely on the visible user interactions and decision points that define the overall flow of the ticket purchase.

8.2 Restrictions and Business Rules

An essential part of this model is the inclusion of **explicit business rules and restrictions** that govern the user experience. Rather than embedding these rules as hidden validations, they were modeled as **text annotations** linked to the relevant steps in the process. This provides transparency and improves communication between business analysts, developers, and stakeholders.

The following restrictions were defined and represented with their corresponding annotations in the diagram:

- **Maximum Tickets per Purchase:** A user may purchase up to six tickets per transaction to prevent reselling or bulk acquisition.
- **Real-Time Availability:** Ticket availability is validated at the time of selection; if tickets are sold out, the user must reselect.
- **Age Restriction:** Events marked as +18 require the buyer to confirm age eligibility before proceeding to payment.
- **Reservation Time Limit:** The user has ten minutes to complete the payment before the tickets are released.
- **Payment Validation:** The success of the transaction depends on the external payment gateway's authorization.

- **Retry Limit:** If a payment fails, the buyer can retry up to three times before the process is automatically canceled.

8.3 Justification of the Design

This modeling choice ensures that the process diagram remains **user-centric**, intuitive, and aligned with real operational behavior. By emphasizing human actions and decisions, the diagram captures how users actually experience the purchase flow in a platform similar to *tuboleta.com*. It also allows stakeholders to easily identify potential friction points, such as time limits or failed transactions, which are critical for usability and business performance.

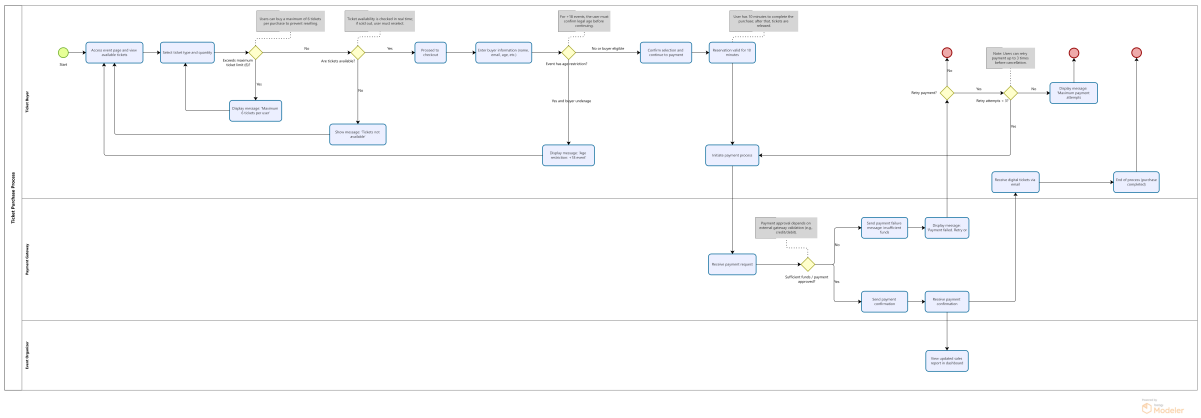

Figure 6: Ticket Purchase Process

9 Web UI Mockups

The mockups represent the visual and interactive design of the Event Management Platform's user interface. They bridge the gap between functional requirements and the final implementation, enabling validation of navigation flows and visual consistency before coding.

The interface was prototyped using the main user roles defined in the user stories: Event Organizer, Ticket Buyer, and Platform Administrator.

Each mockup corresponds to a critical user action: event creation, ticket purchase, payment confirmation, and dashboard display. This visual representation ensures alignment between the system's usability objectives and its architectural and process models (BPMN).

9.1 Design Purpose

The main objective of these mockups is to demonstrate the functional scope and visual consistency of the platform. Every interface follows a coherent design system that ensures:

- A **clear navigation structure**, allowing users to access features easily through a sidebar and top navigation bar.
- **Consistent visual hierarchy**, prioritizing key actions such as creating an event, purchasing tickets, or confirming payments.
- **Seamless user flow**, reflecting the main processes modeled in the user stories and BPMN diagrams.

9.2 Mockups Included

The mockups included in this section are:

1. **Homepage:** Entry point for users to browse and discover upcoming events.

2. **Event Details:** Provides detailed information about an event and allows ticket selection by type or tier.
3. **Secure Checkout:** Displays the payment form and order summary for completing a purchase.
4. **Purchase Confirmation:** Confirms a successful transaction and generates the digital ticket with a unique QR code.
5. **Organizer Dashboard:** Central interface for event organizers to track sales, revenue, and upcoming events.
6. **Create New Event Form:** Enables event organizers to create, configure, and publish new events on the platform.

These mockups serve as a visual complement to the system's functional and process models, providing a realistic representation of how end users will interact with the platform once implemented.

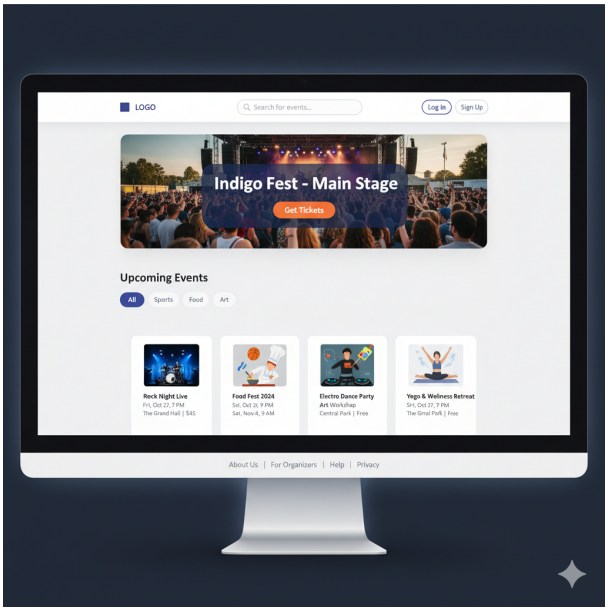

Figure 7: Homepage

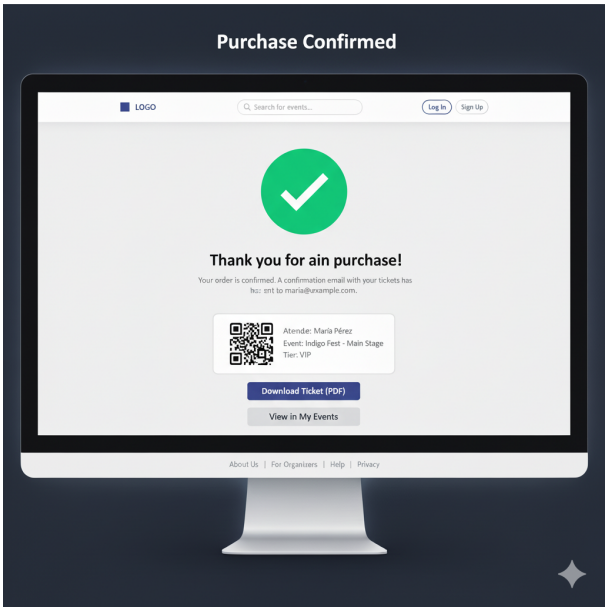

Figure 8: Purchase Confirmed

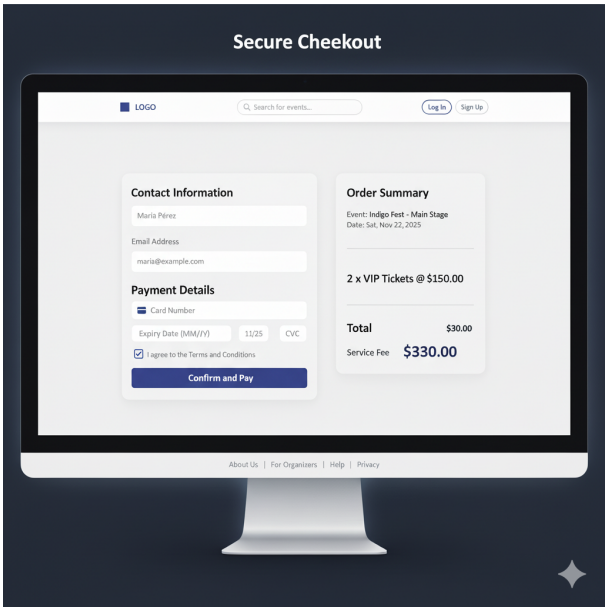

Figure 9: Secure Checkout

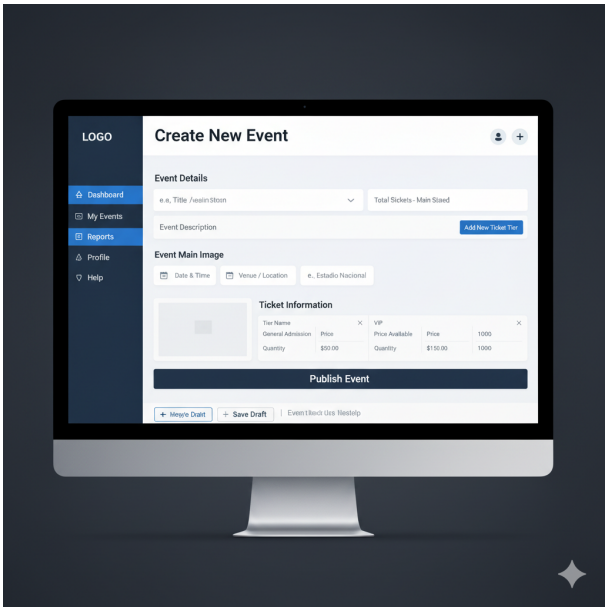

Figure 10: Create New Event Form

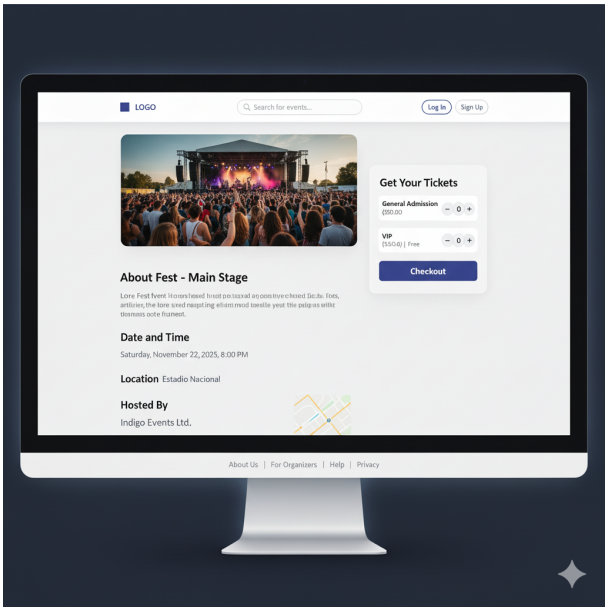

Figure 11: Event Details

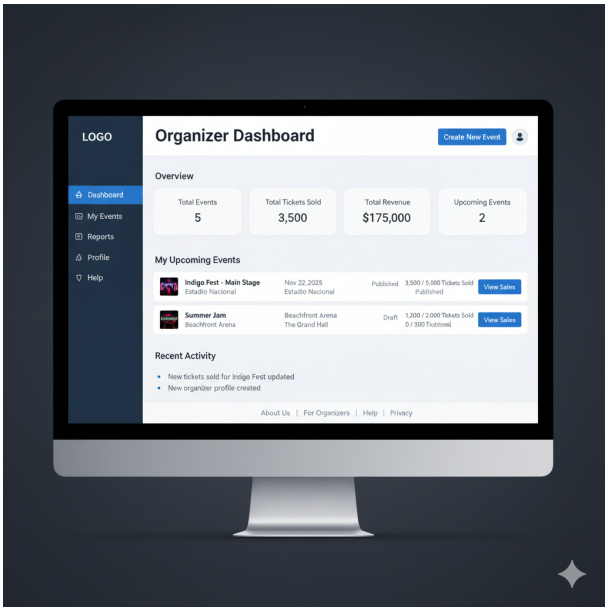

Figure 12: Organizer Dashboard