

WORKSHOP 2 — DESIGN ARTIFACTS AND SYSTEM MODELING

JACOBO ALZATE CORREDOR
JEISSON DUVÁN BAREÑO RUIZ
JUAN JERÓNIMO GÓMEZ RUBIANO
JOHN JAIRO PÁEZ ALBINO

Teacher:
Carlos Andrés Sierra Virgüez



UNIVERSIDAD
NACIONAL
DE COLOMBIA

National University of Colombia
Faculty of Engineering
Department of Systems and Industrial Engineering
Bogotá D.C.
2025

1. CRC Cards

- Identify and describe the main classes in your system.
- For each class, specify its responsibilities and collaborators.
- Present CRC cards in a table or diagram format (one per each class).

User	
Authenticate and manage account (register, sign in, profile, payment methods) Manage library, shopping cart, wishlist and orders Write and edit reviews Check achievements and saved files on the cloud	Library, ShoppingCart, Wishlist, Order, Review, Achievement

Videogame	
Show metadata (name, description, genre, price, system requirements, developer) Associate and show reviews, achievements and promotions Validate availability and installation in library Provide data for sales and performance statistics	Review, Promotion, Library, Publisher, Achievement

Library	
Store videogames bought by the user Manage installation, downloads and total gameplay time Verify and validate licenses	User, Videogame, License

License	
Register the relationship between an	User, Videogame

user and a bought videogame Validate authenticity and access validity Allow, revoke or renovate access if applied	
---	--

ShoppingCart	
Store product Load product Delete product Calculate subtotal Calculate total Begin purchase Validate product availability	User, Videogame, Promotion, Order, CartItem

CartItem	
Represent a videogame inside the shopping cart with quantity, price and discount Calculate individual subtotal	ShoppingCart, Videogame, Promotion

Order	
Register a purchase with quantity, unitary price, taxes and state of purchase Associate purchased videogames Apply promotions and calculate final amount	CartItem User Promotion

OrderItem	
Represents a videogame inside the order with its final price Calculate subtotals and applied discounts	Order, Videogame, Promotion

Promotion

Define discount rules (game, category, coupon) Calculate effective prices through promotion validity Apply over videogames, shopping cart and orders	Videogame, ShoppingCart, Order, Publisher
--	---

Review	
Register text, grading, author and date Show average review grade Manage edition and deletion	Videogame User

Wishlist	
Add and remove games Check promotion status Notify price changes if applies	Videogame User Promotion

Achievements	
Define game achievements Register and show unlocked achievements by user Synchronize saved progress on the cloud	User Videogame

Publisher	
Publish and manage published videogames Create and manage promotions Activate and deactivate catalog games	Videogame, Promotion, Administrator

Administrator

Manage users, reviews and published content Approve and reject publishing requests Supervise support and system metrics	User, Publisher, Review, SupportTicket
---	--

SupportTicket	
Register incidents or support requests Allow responses from agents or administrators Change status (open, in progress, closed) Register status	User, Administrator

2. Mockups

- Create mockups or wireframes for your screens.
- You may use digital tools (e.g., Figma, Balsamiq) or hand-drawn sketches.
- Briefly explain the purpose of each screen.

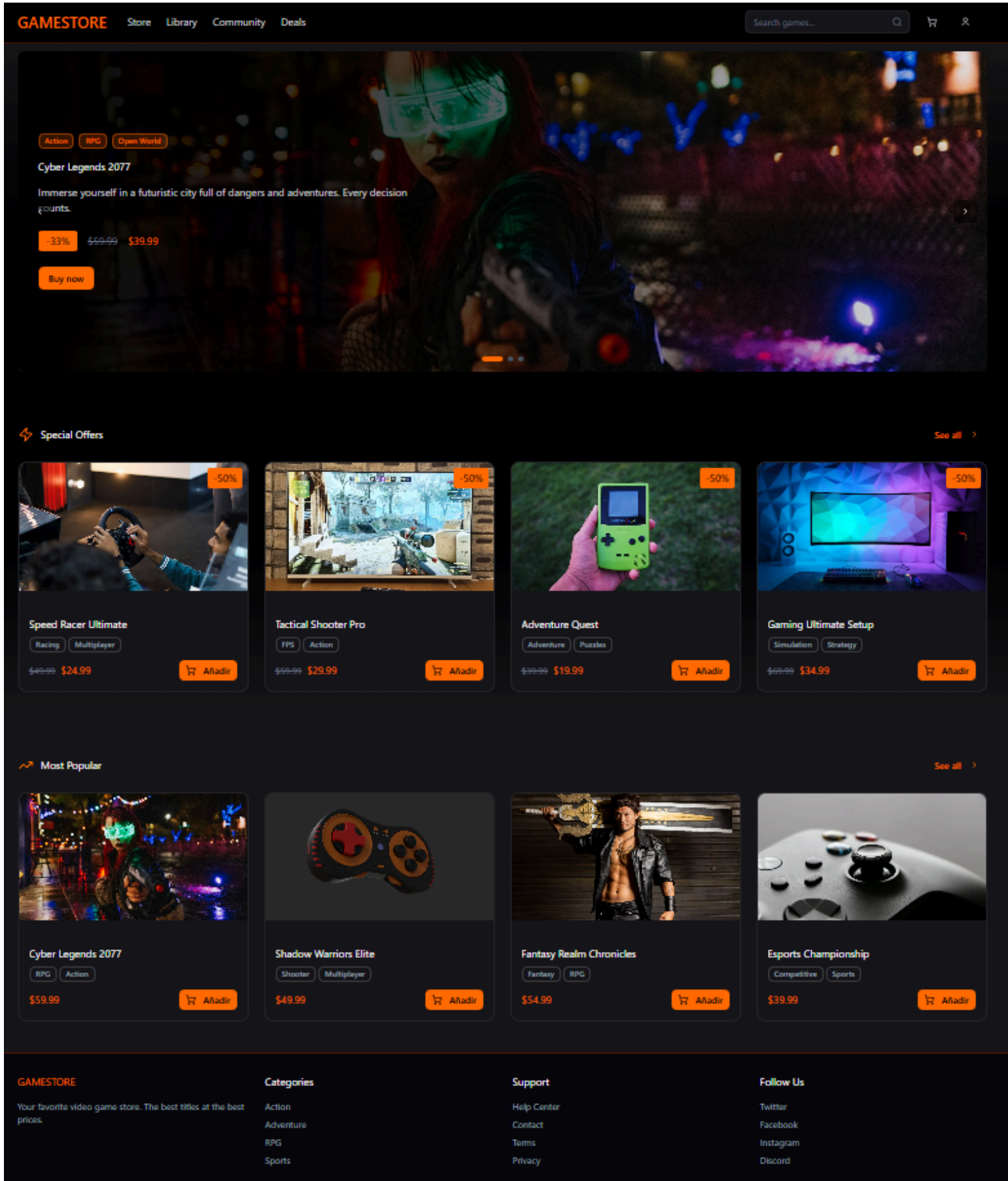


Figure 1. ESTIM main page with games and offers.

This image illustrates the main page of ESTIM, where the store is located. It shows some games available for purchase along with their offers and various interfaces that can be accessed.

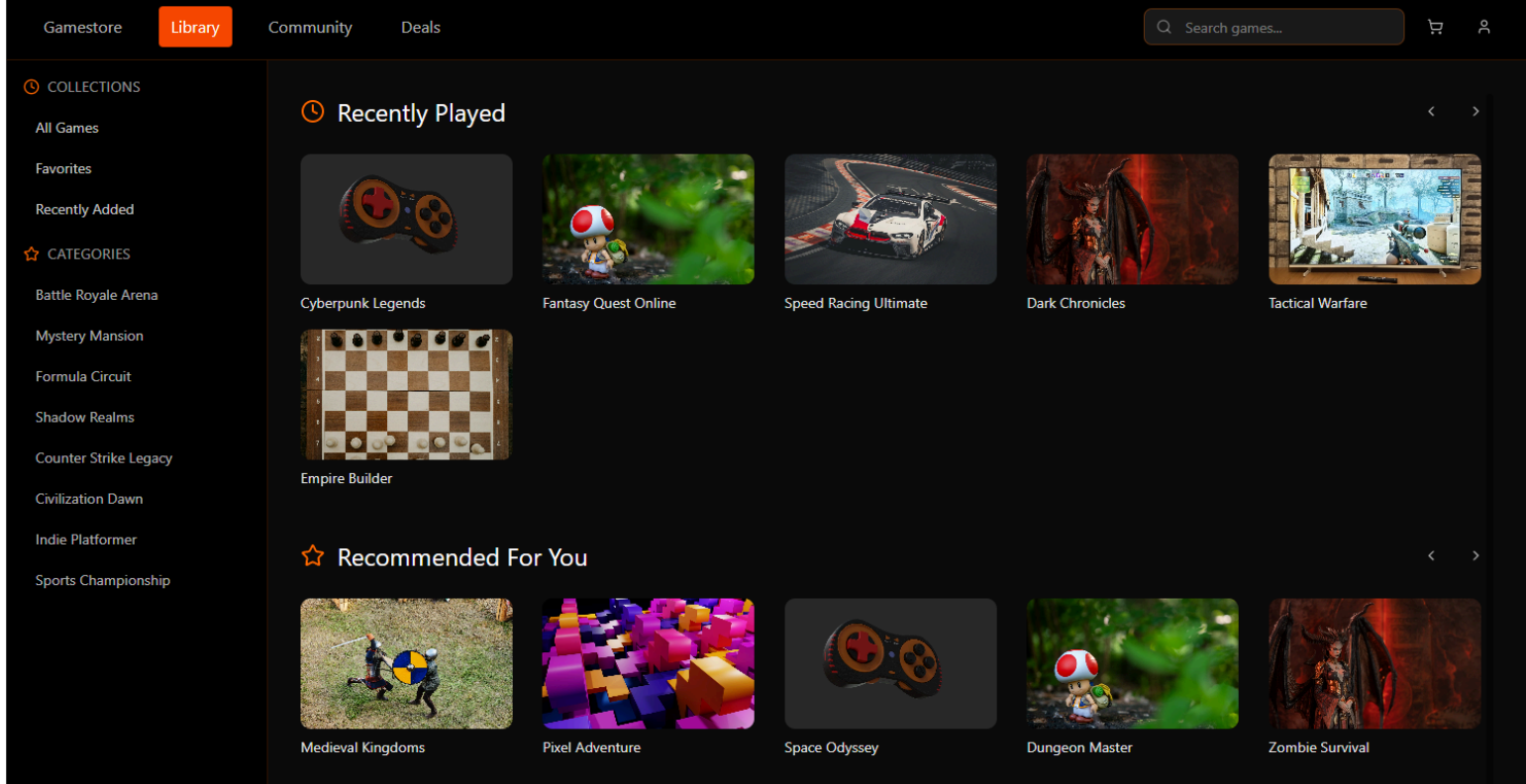


Figure 2. ESTIM library showing purchased games and navigation menu.

This image shows the ESTIM library, where all the games purchased by the user are displayed. On the left side, there is the navigation bar.

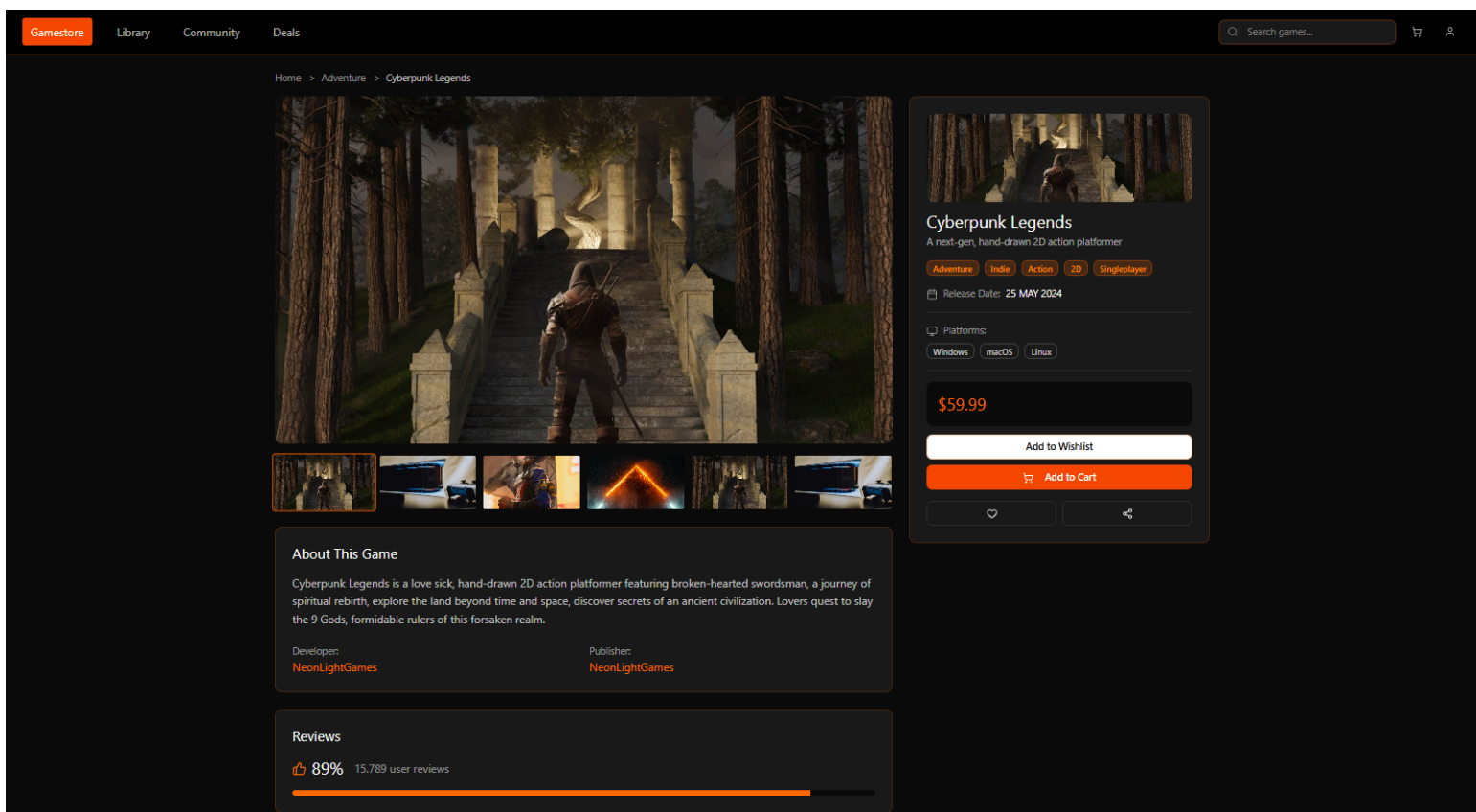


Figure 3. Game page with details, reviews, and purchase options.

This image shows a game's page, where you can see images of the game, a brief description of it, the percentage of positive reviews, the game's price, and a button to add it to the wishlist.

3. Business Model Processes

- Document at least one core business process using an activity diagram or BPMN.
- Describe the process and its role in your application.

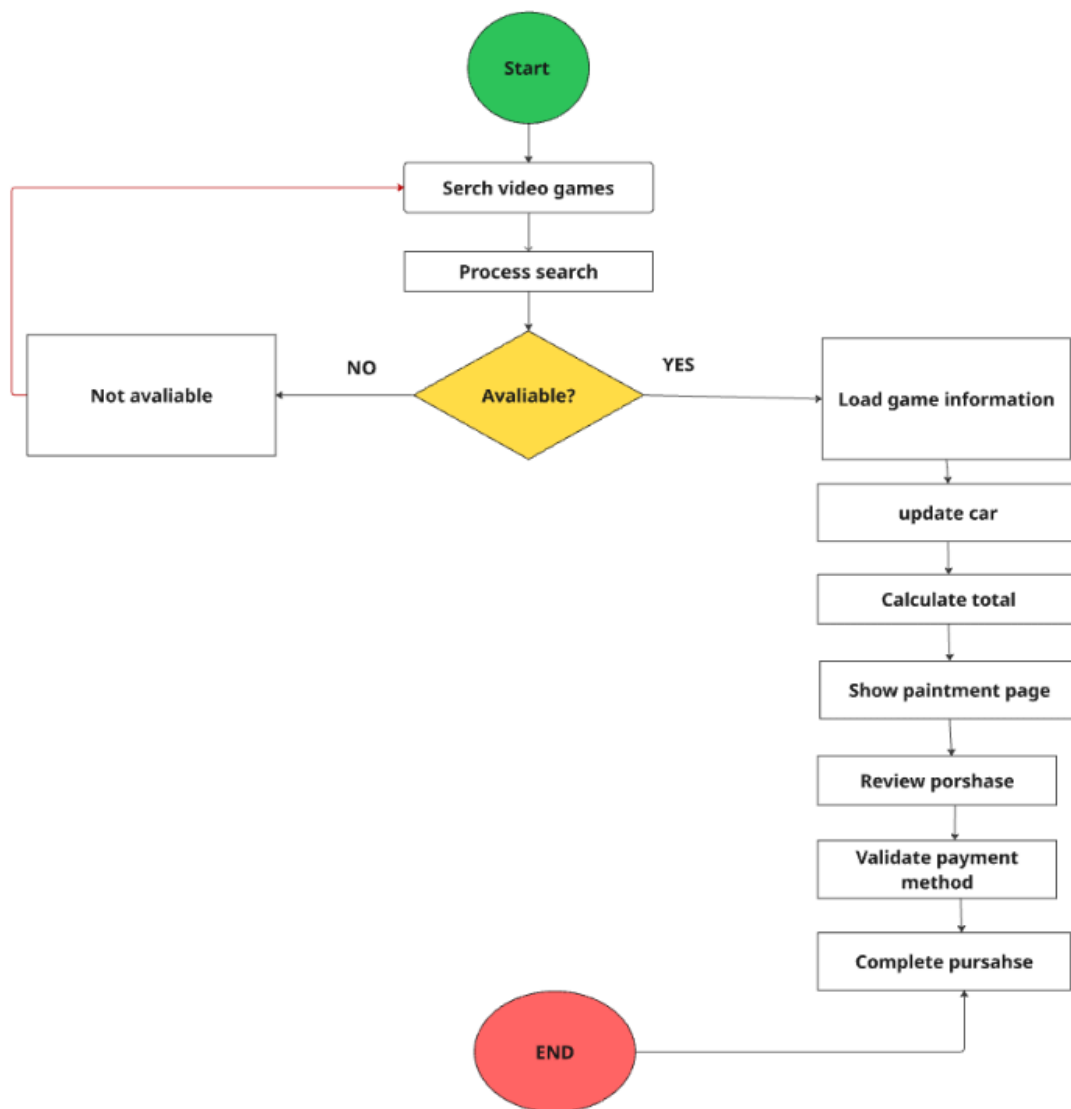


Figure 4. BPMN diagram of the video game purchase process.

This diagram shows the process a user follows to purchase a video game, from searching and selecting to payment and completion of the transaction.

4. Architecture Diagram

- Draw a software architecture diagram showing system components and their interactions.
- Indicate technologies, layers, and communication flows.

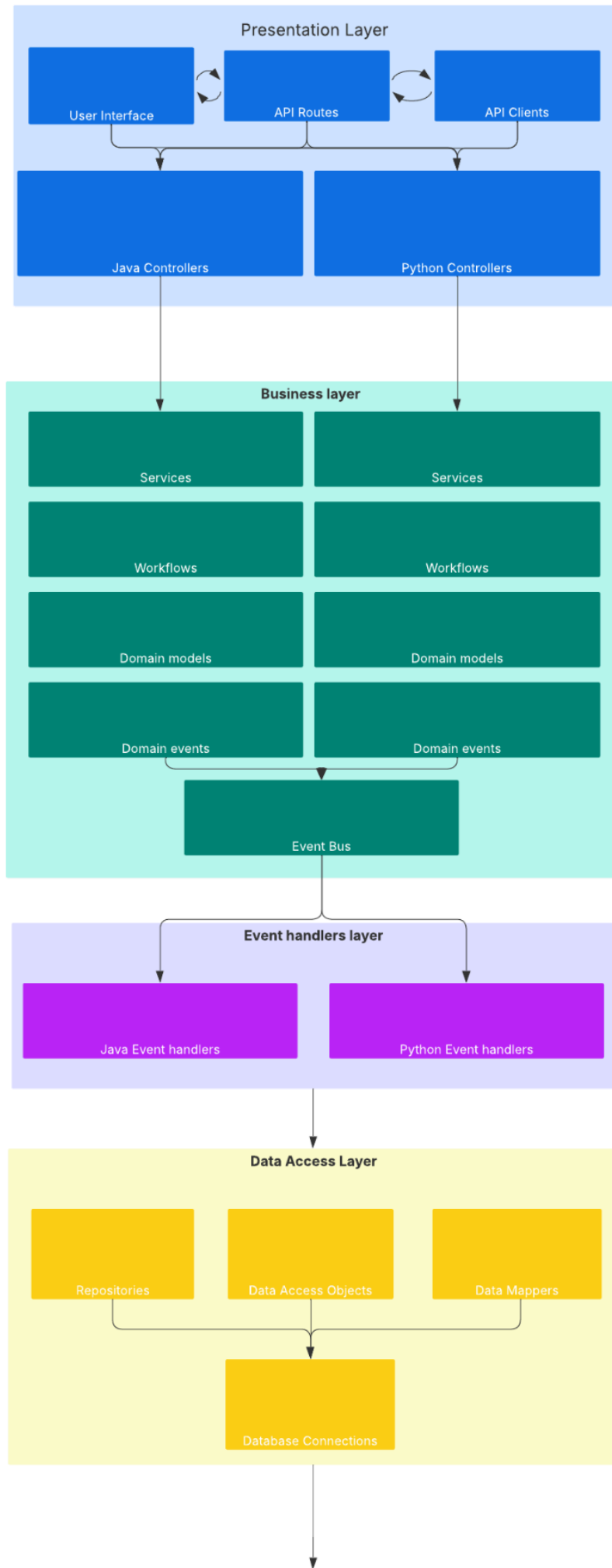
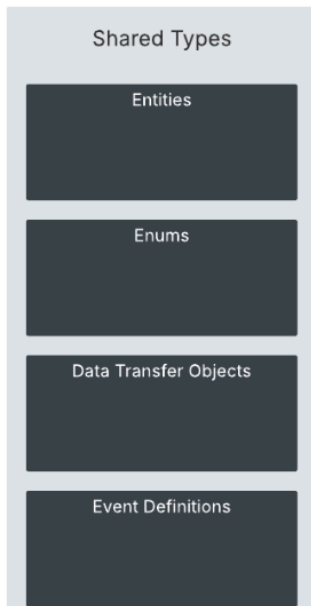


Figure 5. ESTIM layered architecture

Technologies:

- Frontend: React, SPA consuming REST APIs.
- Java backend: Spring Boot (Controllers, Services, Repositories), JUnit.
- Python backend: FastAPI (Routers, Services, DAOs), pytest.
- Database: PostgreSQL (primary), ORMs: JPA/Hibernate (Java), SQLAlchemy (Python).
- Cache & messaging: Redis (cache + pub/sub/streams for events).
- Contracts: OpenAPI (REST DTOs) + JSON Schema for shared types; AsyncAPI for event definitions.
- Infra: Docker & Docker Compose; CI/CD with GitHub Actions.

Layers:

- **Presentation Layer:** The presentation layer consists of the frontend (UI), the different API Routes, which define the endpoints and HTTP requests, and the controllers, which extract data, format responses, handle errors and transform them into HTTP status codes, and send information to the business layer. The Java controllers handle everything regarding user information and authentication, whereas the Python controllers manage the business logic and UI interactions
- **Business Layer:** Manage logic of the described use cases in Workshop-1 as services, workflows which orchestrate multi-service processes than span through different services, and different domain models and events which encapsulate the different business processes as domain objects, and separate the effects from main business operations. The event bus is tasked with communicating the Java backend with the Python backend.
- **Event handlers Layer:** This layer is tasked with listening for the occurrence of the different system events, and executing the associated business processes with said events
- **Data access Layer:** This layer is tasked with storing information with repositories, managing of Data Access Objects, isolating and executing database queries, as well as mapping data to domain models
- **Database Layer:** Formal databases of the system, and use of Redis for caching hot reads
- **Shared Types:** Using shared types across layers and databases allows us to avoid different serialization errors, data mismatches or runtime exceptions, by using common structures and references that facilitate the communication between modules. We define entities, enums, data transfer objects and event definitions to globalize the communication of the system.

Communication flows:

- **Frontend to Backend:** The UI communicates with both backends through the API REST, using Data Transfer Objects.
- **Cross-backed events:** The services emit domain events to Redis. Java handlers consume events to update user information. Game and service information is updated by Python handlers.
- **Services:** The different services use Repositories and Data Access Objects to execute their respective read-write operations.
- **Data access:** The different repositories are tasked with executing the database queries of the system.

5. Class Diagram

- Create UML class diagrams for the main classes.
- Include attributes, methods, and relationships.

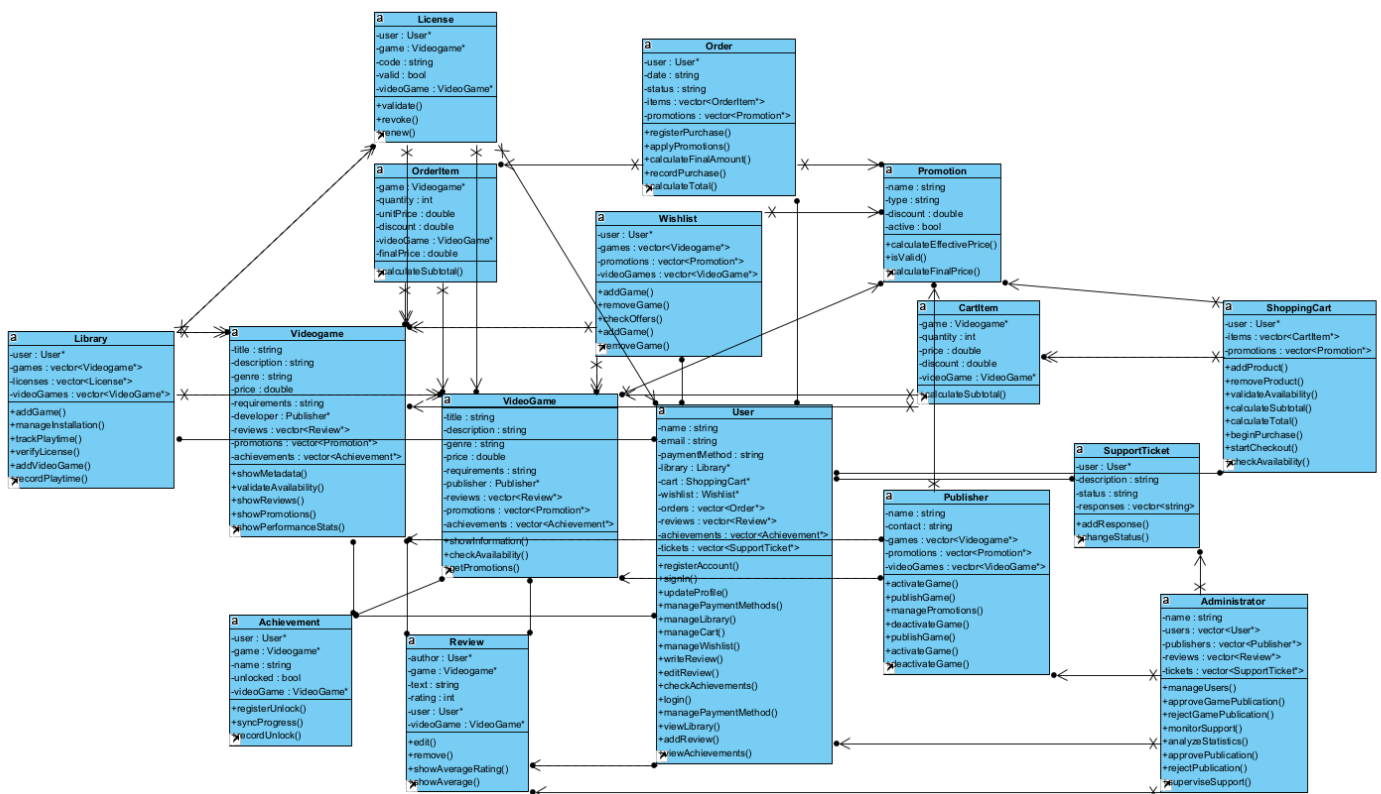


Figure 6. ESTIM class diagram.

This diagram represents the overall structure of the system, showing the main classes and the relationships between them. It illustrates how the different components interact and how dependencies are managed within the system.

The VideoGame class appears twice only to improve the visual clarity of the diagram due to the large number of associations it maintains with other classes.

6. Relational Database Model

- Design the relational database schema for your application.
- Include tables, primary and foreign keys, and relationships.
- Present the model as an ER diagram or table definitions.

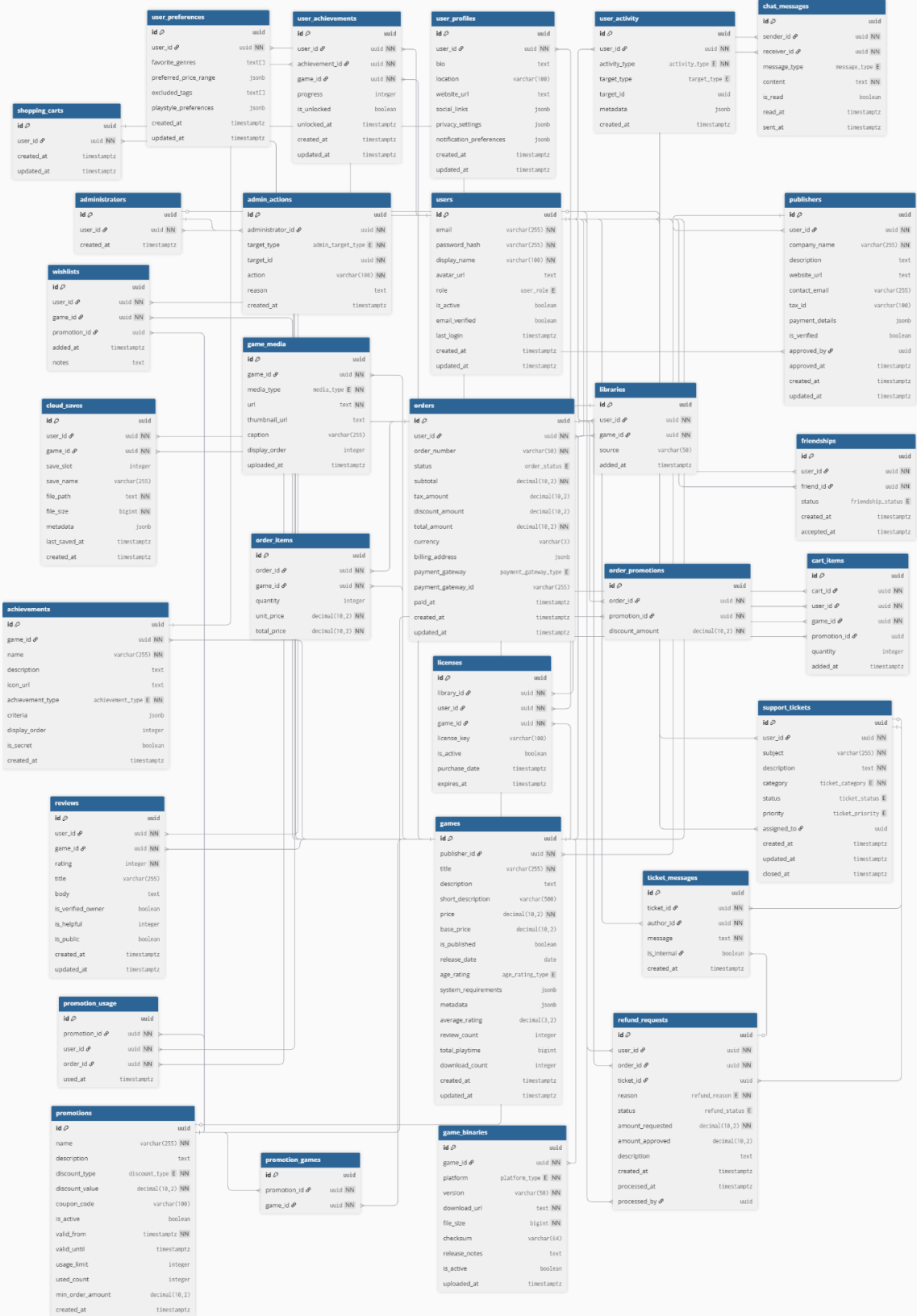


Figure 7. ESTIM database schema

Entity-relationship diagram of the game platform covering users, profiles, preferences, achievements, messaging, orders, licenses, games, media, binaries, reviews, promotions, support tickets, and refunds with their foreign-key links.

REFERENCES

Lucidchart. (s.f.). *How to draw architectural diagrams*. Retrieved from <https://www.lucidchart.com/blog/how-to-draw-architectural-diagrams>

Lucidchart. (s.f.). *What is an Entity Relationship Diagram?*. Retrieved from <https://www.lucidchart.com/pages/er-diagrams>

UML Diagrams. (s.f.). *UML Class Diagrams Overview*. Retrieved from <https://www.uml-diagrams.org/class-diagrams-overview.html>

Visual Paradigm. (s.f.). *What is BPMN?*. Retrieved from <https://www.visual-paradigm.com/guide/bpmn/what-is-bpmn/>