

DESIGN ARTIFACTS AND SYSTEM MODELING
DAYRO MORENO GOL O ASISTENCIA

Presented by:

Daniel Libardo Diaz Gonzalez - dandiazgo@unal.edu.co
Andres Felipe Leon Sanchez - anleonsa@unal.edu.co
Juan David Loaiza Reyes - juloaizar@unal.edu.co
Juan David Chacon Muñoz - juchaconnm@unal.edu.co
Javier Santiago Giraldo Jiménez - jgiraldoji@unal.edu.co
Heric Francisco Vargas Cabiatiava - hevargasc@unal.edu.co

Instructor:

Liliana Marcela Olarte Mesa
lmolartem@unal.edu.co

Saturday, November 8



Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas y computación e industrial

2025



CONTENTS

1. CRC cards.....	3
2. Mockups.....	7
3. Business Model Processes.....	10
User Management process.....	10
4. Architecture Diagram.....	13
5. Class Diagram.....	13
6. Relational Database Model.....	17
DIAGRAM-NOTIFICATIONS-MicroService.....	18
DIAGRAM-WORKBOARD-MicroService.....	19
DIAGRAM-USERS-MicroService.....	20
7. Delivery Format.....	21



1. CRC cards

Our project is microservices oriented, knowing that we are going to create the CRC card per each microservice that we must create for this project.

Class name	ms-users (Microservice users)
Responsibilities	Create, read, update, and delete user accounts according to the entity-relation diagram. Generate and issue login tokens (JWT) for secure access by all app microservices. Manage user roles (CRUD) and store permissions for each role. Validate access tokens and respond with permission status for requesting microservices, ensuring correct authorization.
Collaborators	Frontend ms-forum ms-notifications ms-meeting ms-workboard ms-chat

Class name	ms-forum (Microservice forum)
Responsibilities	Create and manage discussion posts and themes, allowing comments, upvotes, and



	downvotes.
	Enable navigation by tags, allowing users to assign and use free-form tags for posts.
	Display statistics, such as number of topics and recent activity.
	Validate user sessions by communicating with ms-users.
	Publish events for user interactions to ms-notifications (e.g., when a new comment, upvote, or post occurs).
Collaborators	Frontend
	ms-users
	ms-notifications

Class name	ms-meeting (Microservice meeting)
Responsibilities	The microservice listens for notification events from other services (such as forum, users, chat, and all) and processes the data needed for sending a notification. It formats the notification content using templates, localization, and user preferences. This all is in app. It records notifications sent (for each user) and their delivery status (sent, failed, read), allowing further querying, analytics, or resend attempts.
Collaborators	Frontend ms-forum ms-meeting ms-workboard ms-chat ms-users



Class name	ms-meeting (Microservice meeting)
Responsibilities	<p>Establish and manage real-time video/audio meetings, supporting 1:1 calls and group conferences.</p> <p>Handle signaling and participant connectivity, including negotiation and management of sessions (via WebRTC SDP).</p> <p>Track and report meeting statistics, such as number of participants, call duration, and resource usage.</p>
Collaborators	Frontend ms-users ms-notifications

Class name	ms-workboard (Microservice Workboard)
Responsibilities	<p>Create and manage workboards and boards, including creation, update, deletion, and retrieval of boards.</p> <p>Handle tasks/cards management within boards, supporting CRUD operations on tasks, including assigning users, setting statuses, priorities, and deadlines.</p> <p>Support drag-and-drop and status transitions for tasks to move them across different columns or states (e.g., To Do, In Progress, Done).</p> <p>Provide real-time collaboration features like notifications on task changes, comments, and updates.</p>
Collaborators	Frontend ms-users ms-notifications



Class name	ms-chat (Microservice Chat)
Responsibilities	Manage real-time 1:1 and group chat sessions, enabling users to send and receive messages instantly.
	Store and retrieve chat history to support message persistence across user sessions and devices.
	Handle user presence and status updates, such as online, offline, and typing indicators.
	Ensure message delivery guarantees and synchronization, including queuing messages when users are offline and delivering them upon reconnection.
Collaborators	Frontend
	ms-users
	ms-notifications

Class name	Frontend
Responsibilities	Provide user interface for all microservices, including forum, chat, workboard, meetings, and notifications.
	Manage user session and authentication state, interacting with ms-users to validate login and maintain session.
	Translate user actions into API calls to the appropriate microservices (ms-forum, ms-chat, ms-workboard, ms-meeting, ms-notifications).
	Render dynamic content and update UI in real-time based on responses and events from backend microservices.
Collaborators	ms-chat
	ms-users

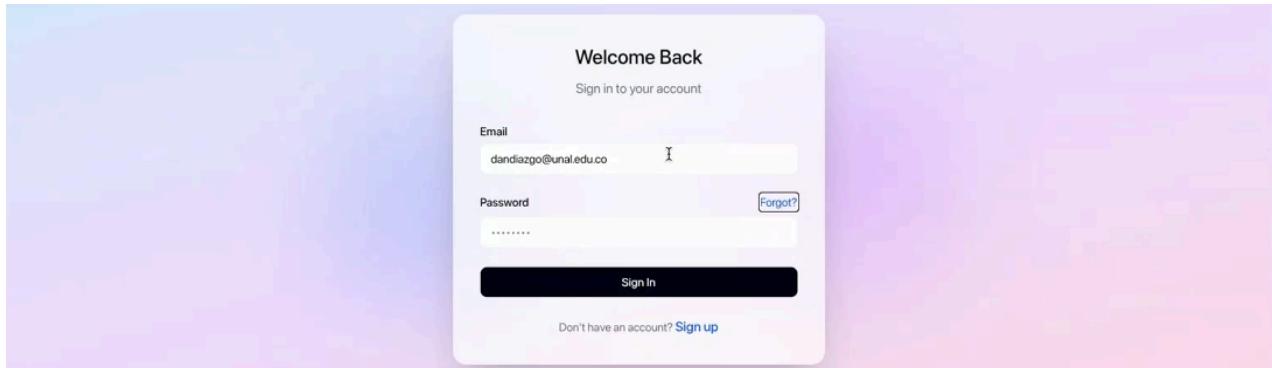


	ms-notifications
	ms-forum
	ms-workboard
	ms-meeting

2. Mockups

This time, for the app mockups we have prepared a demonstrative video that can be accessed here: [Mock-up.mp4](#)

Login:



Feed:



Feed on focus:

Sarah Johnson ~ Trending
Senior Product Designer
2h ago

Just launched our new design system! 🎉 It's been a 6-month journey of collaboration and iteration. Excited to see how it transforms our product experience.

#Design #UI/UX #Product

222

Comments (3)

Alex Thompson 1h ago
This is amazing! Congratulations on the launch!

Jessica Lee 45m ago
Would love to see a case study on this. What were the biggest challenges?

David Kumar 30m ago
Great work! We're working on something similar. Any tips on component

Search bar:

Search for posts, people, or topics...

~ Trending

- React best practices
- Design systems
- Remote work tips
- Product management

Recent

- Figma plugins
- TypeScript tutorials
- UX research methods



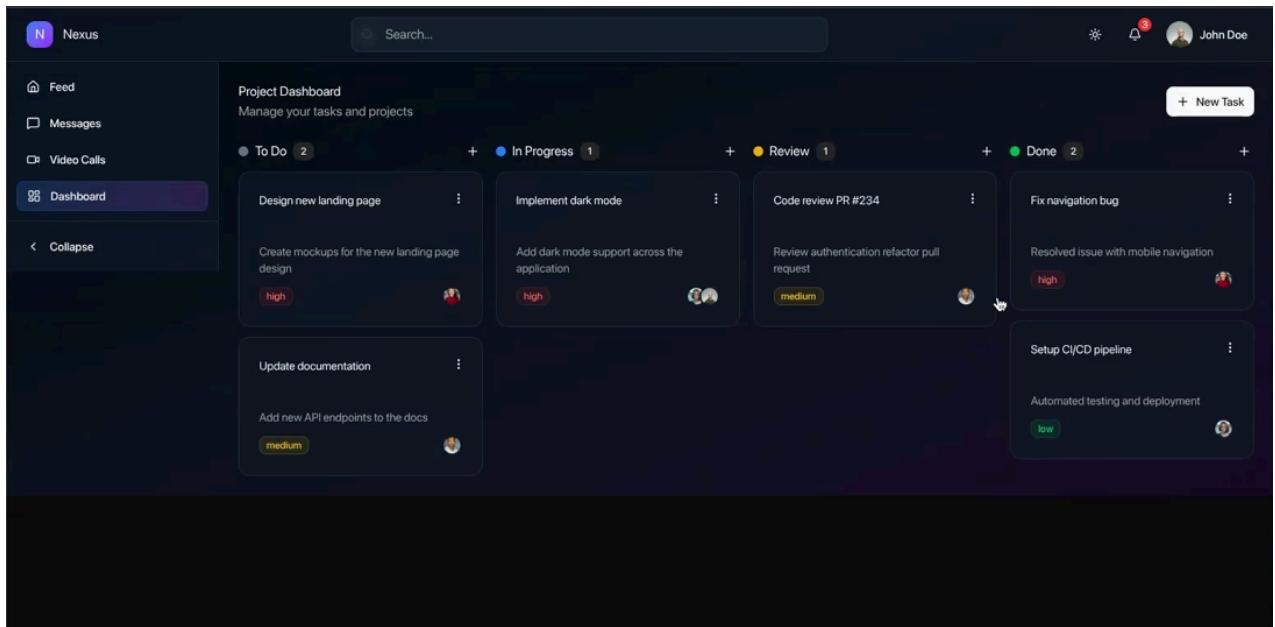
Messages (group chat and direct):

The screenshot shows the Nexus messaging interface. On the left, a sidebar includes options for Feed, Messages (selected), Video Calls, and Dashboard. The main area displays a group chat channel named '# general' with 24 members. A message from Sarah Johnson at 10:30 AM reads: "Hey team! Just wanted to remind everyone about our meeting at 3pm today. We'll be discussing the new feature roadmap." Michael Chen responded at 10:35 AM: "Sounds good! I'll prepare the tech requirements document beforehand." Emily Rodriguez responded at 10:42 AM: "Perfect timing! I've just finished the user research analysis. Will share in the meeting." John Doe responded at 11:15 AM: "I'll join from the client site. Looking forward to it!" Below the group chat, there are sections for Direct Messages, showing conversations with Sarah Johnson, Michael Chen, and Emily Rodriguez.

Video calls:

The screenshot shows the Nexus video calls interface. The sidebar has options for Feed, Messages, Video Calls (selected), and Dashboard. The main area features a 'Video Calls' section with a button to 'Start a call or join a scheduled meeting'. It includes a 'Quick Actions' panel with three options: 'Start Instant Meeting' (begin a video call right now), 'Schedule Meeting' (plan a call for later), and 'Join with Code' (enter a meeting code). To the right is a 'Recent Contacts' panel listing Sarah Johnson (Online), Michael Chen (Online), and Emily Rodriguez (Offline), each with a 'Join' button. Below this is an 'Upcoming Meetings' section showing two scheduled meetings: 'Team Standup' at 10:00 AM with 8 participants and 'Client Presentation'.

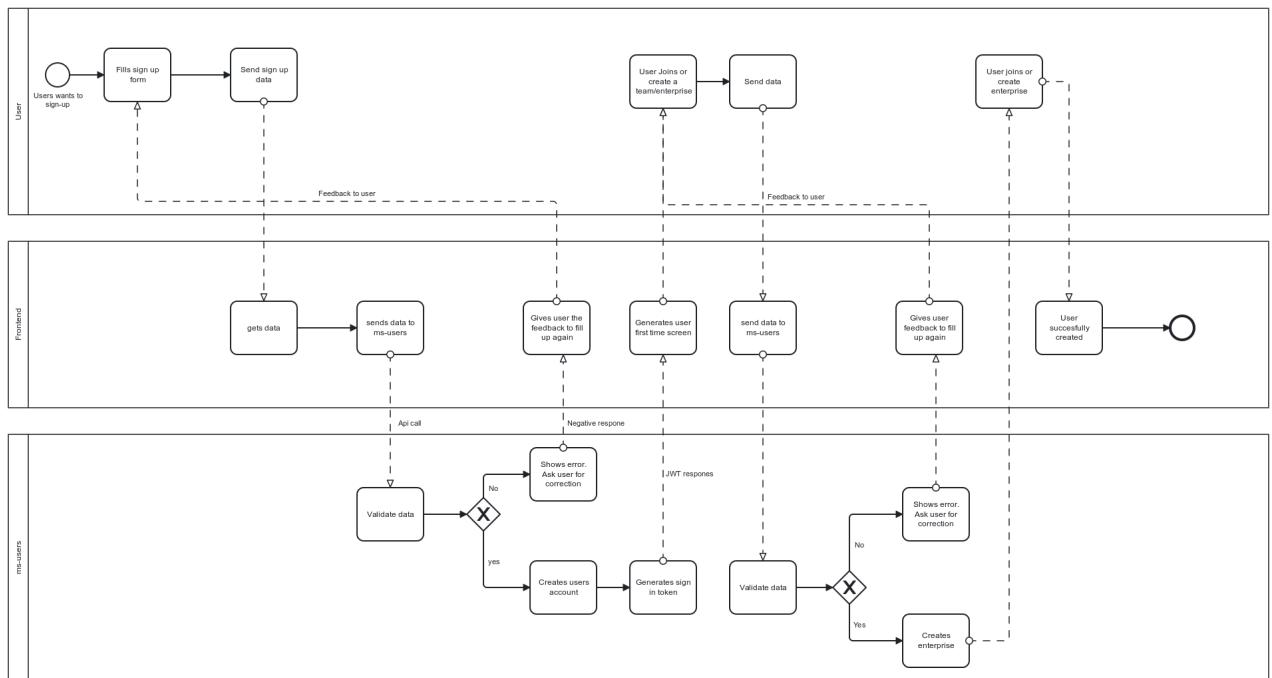
WorkBoards:



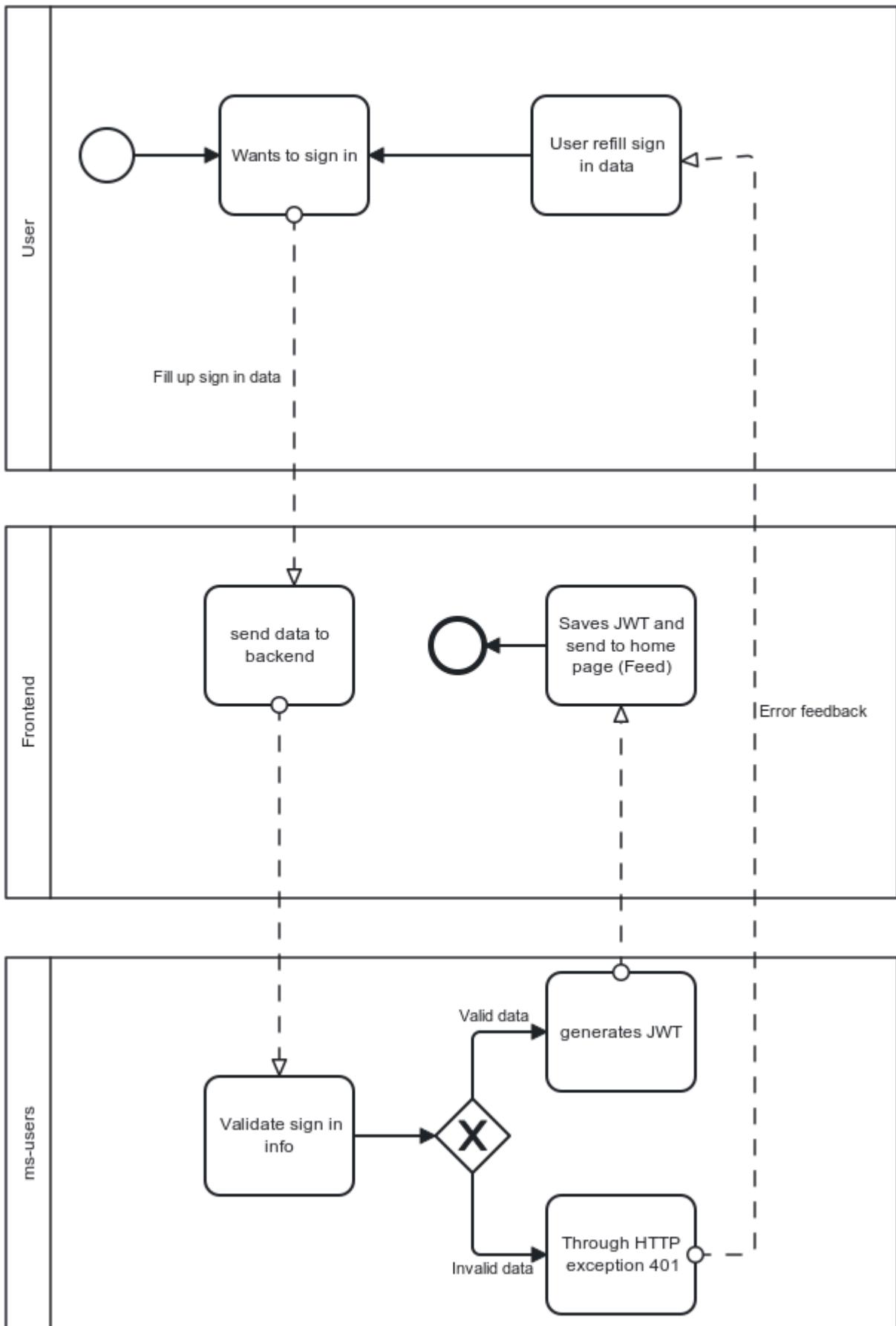
3. Business Model Processes

User Management process

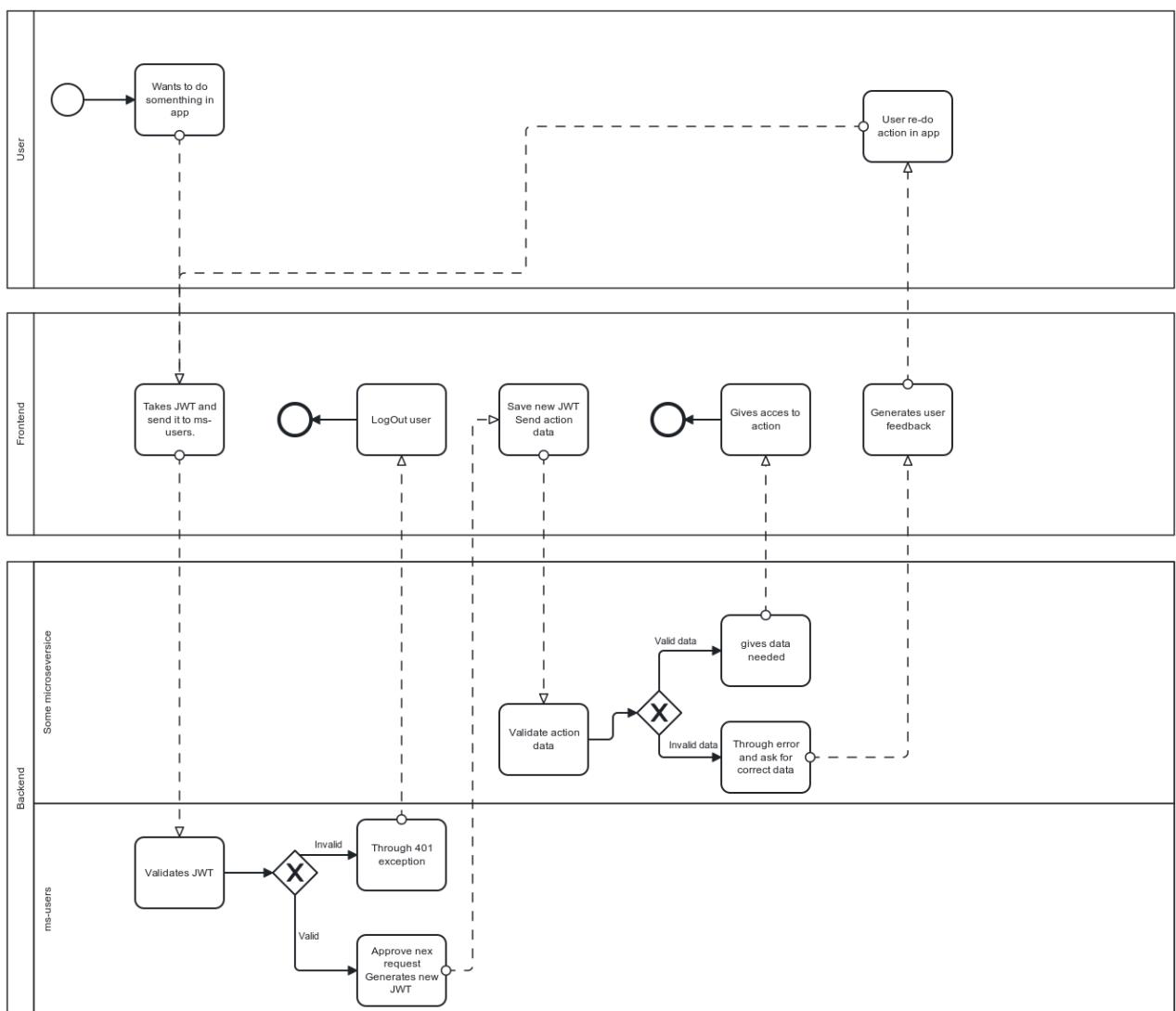
1. Register:



2. Login

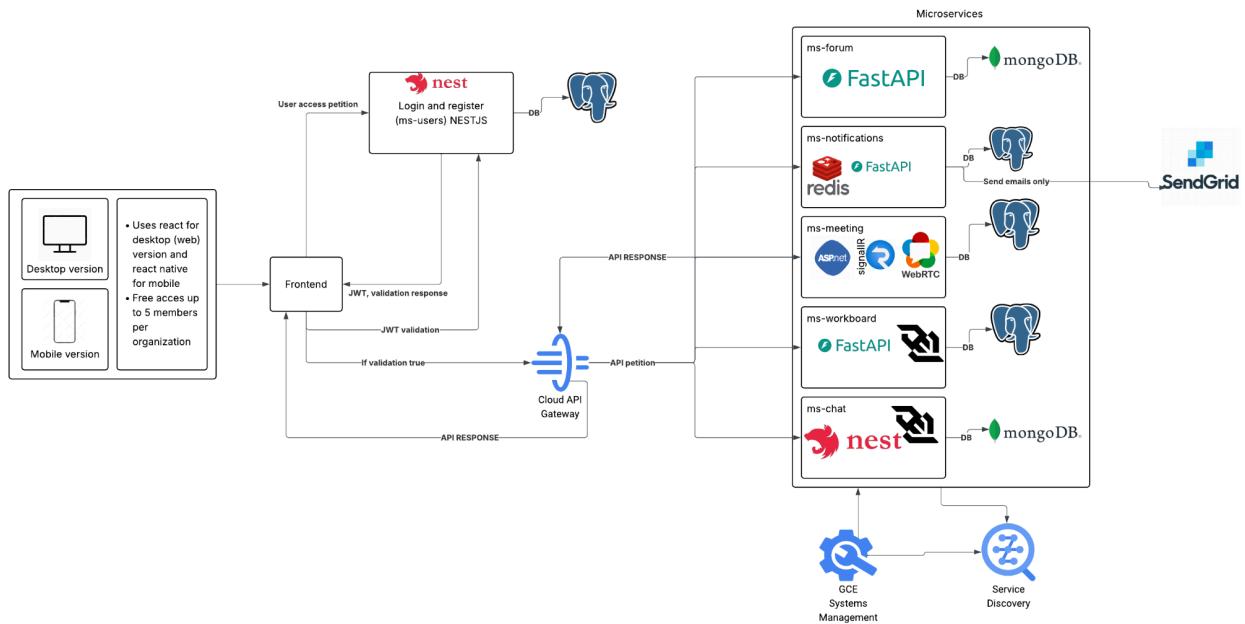


3. JWT validation



4. Architecture Diagram

Full diagram could be seen on: [Architecture diagram.png](#)



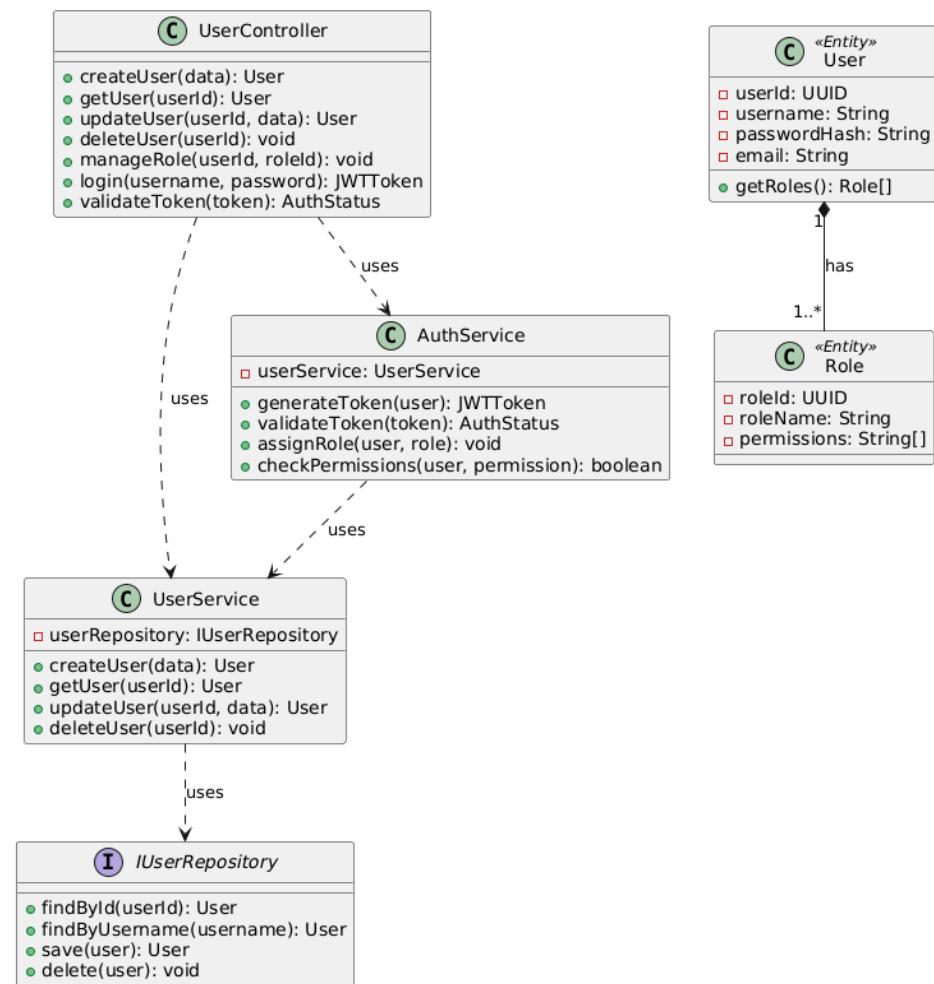
5. Class Diagram

In a microservices-based system, the overall architecture is decomposed into multiple independent services, each responsible for a specific domain or functionality. As a result, class diagrams are not created for the entire system as a whole, but rather one class diagram is developed for each microservice.

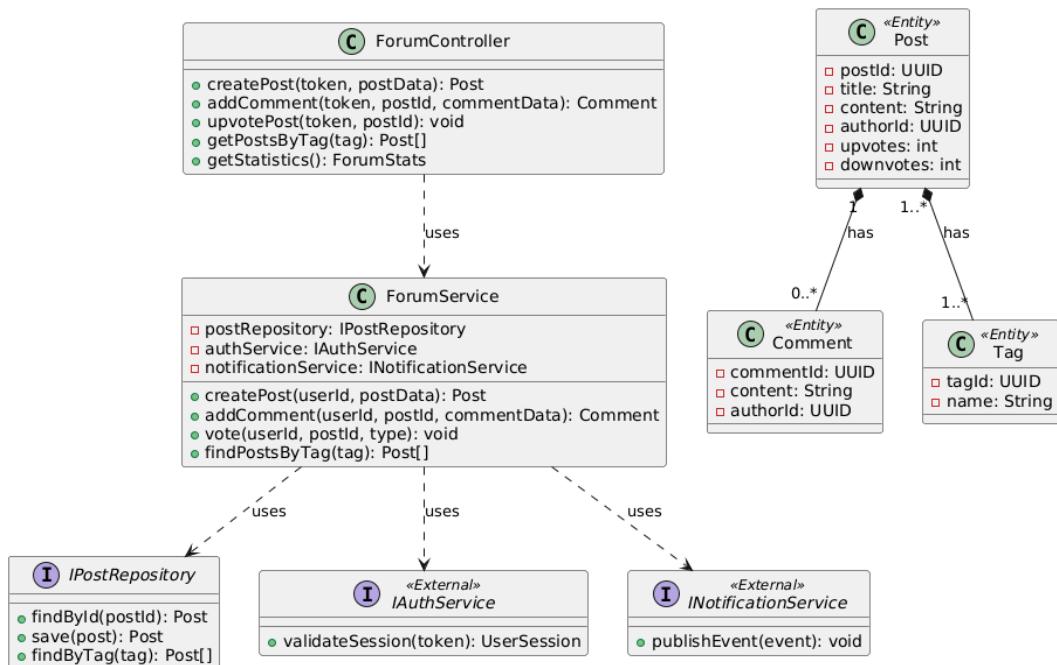
Each diagram focuses on the internal structure of a single microservice, detailing its main classes, attributes, methods, and relationships. This modular approach enhances clarity, maintainability, and scalability, allowing developers to understand and evolve each service independently. It also aligns with the principles of encapsulation and bounded context, which are fundamental to microservices design.



Class Diagram: ms-users

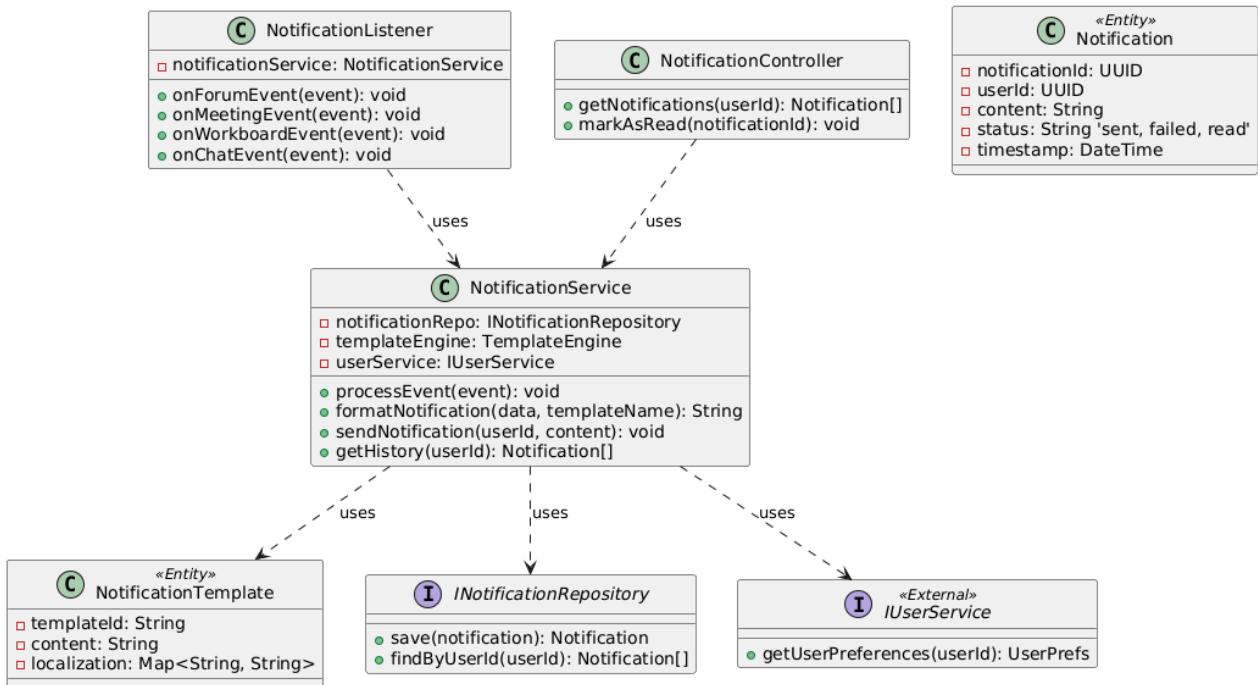


Class Diagram: ms-forum

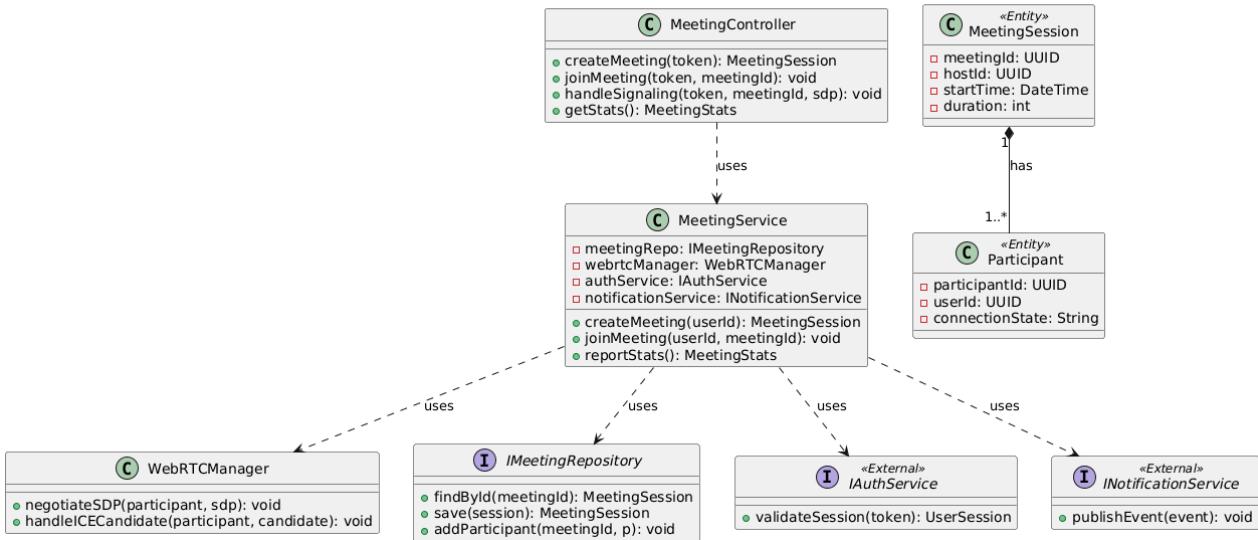




Class Diagram: ms-notifications

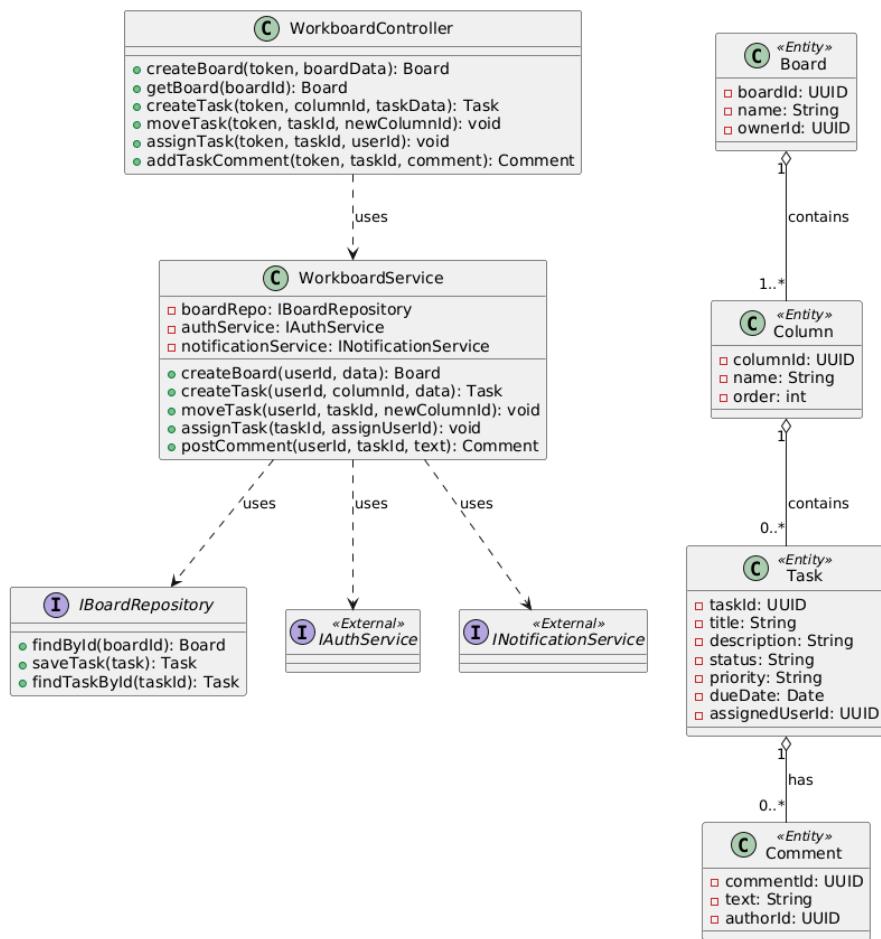


Class Diagram: ms-meeting

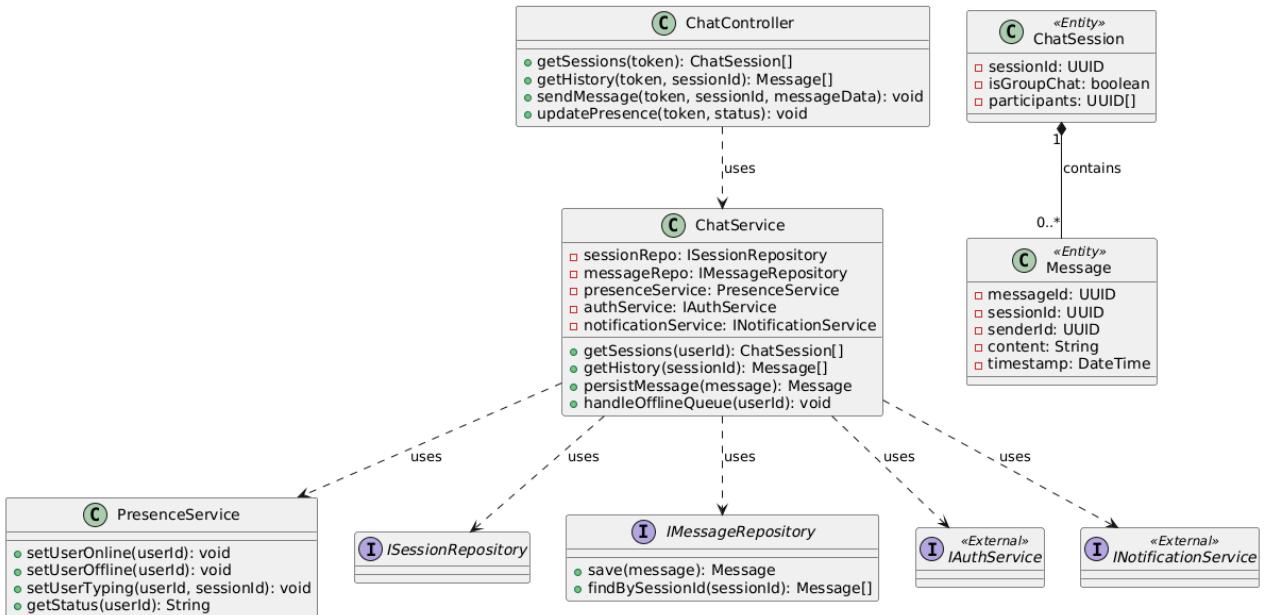




Class Diagram: ms-workboard



Class Diagram: ms-chat



6. Relational Database Model

Full diagram could be seen on: [DB diagram](#)

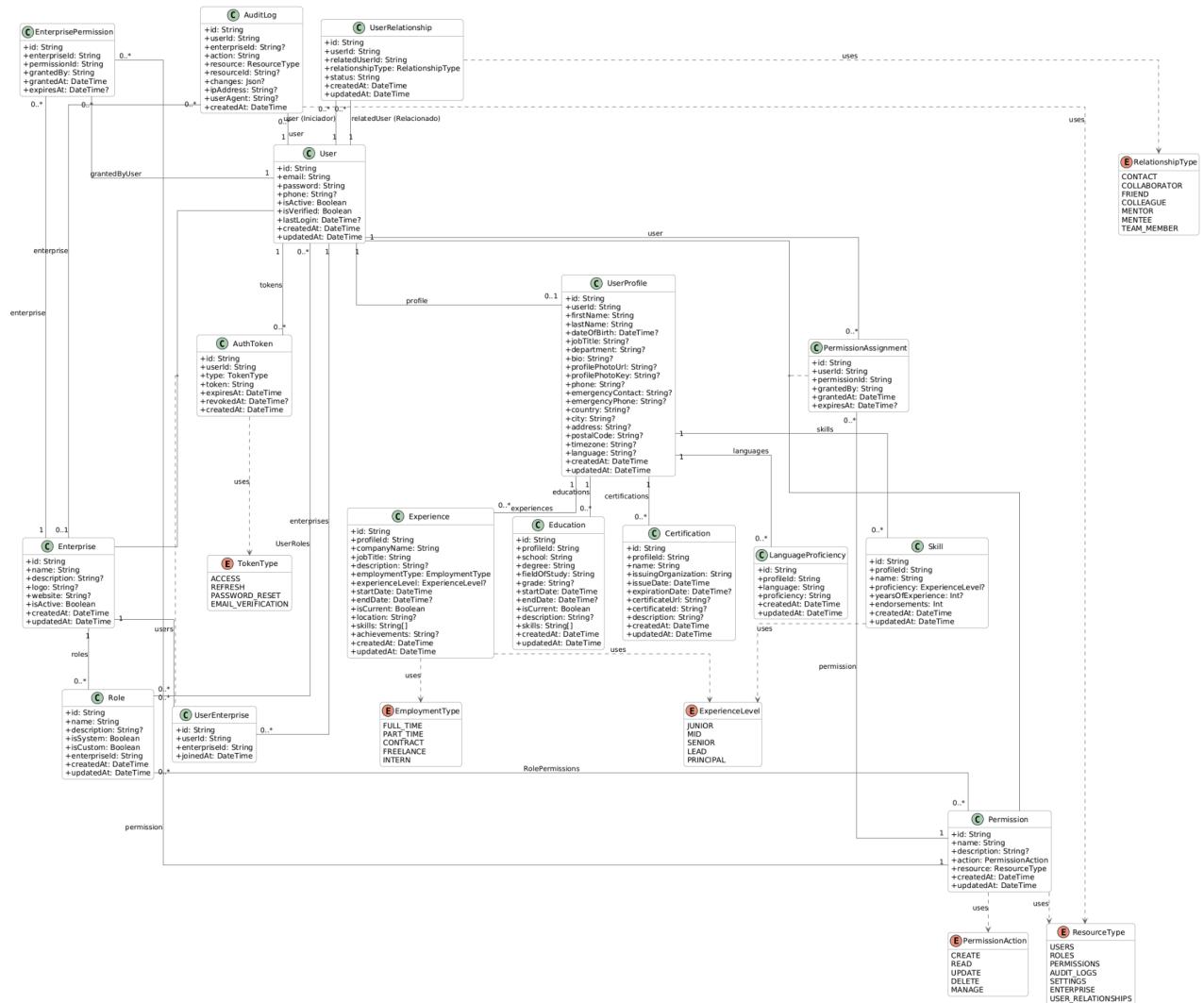




DIAGRAM-NOTIFICATIONS-MicroService

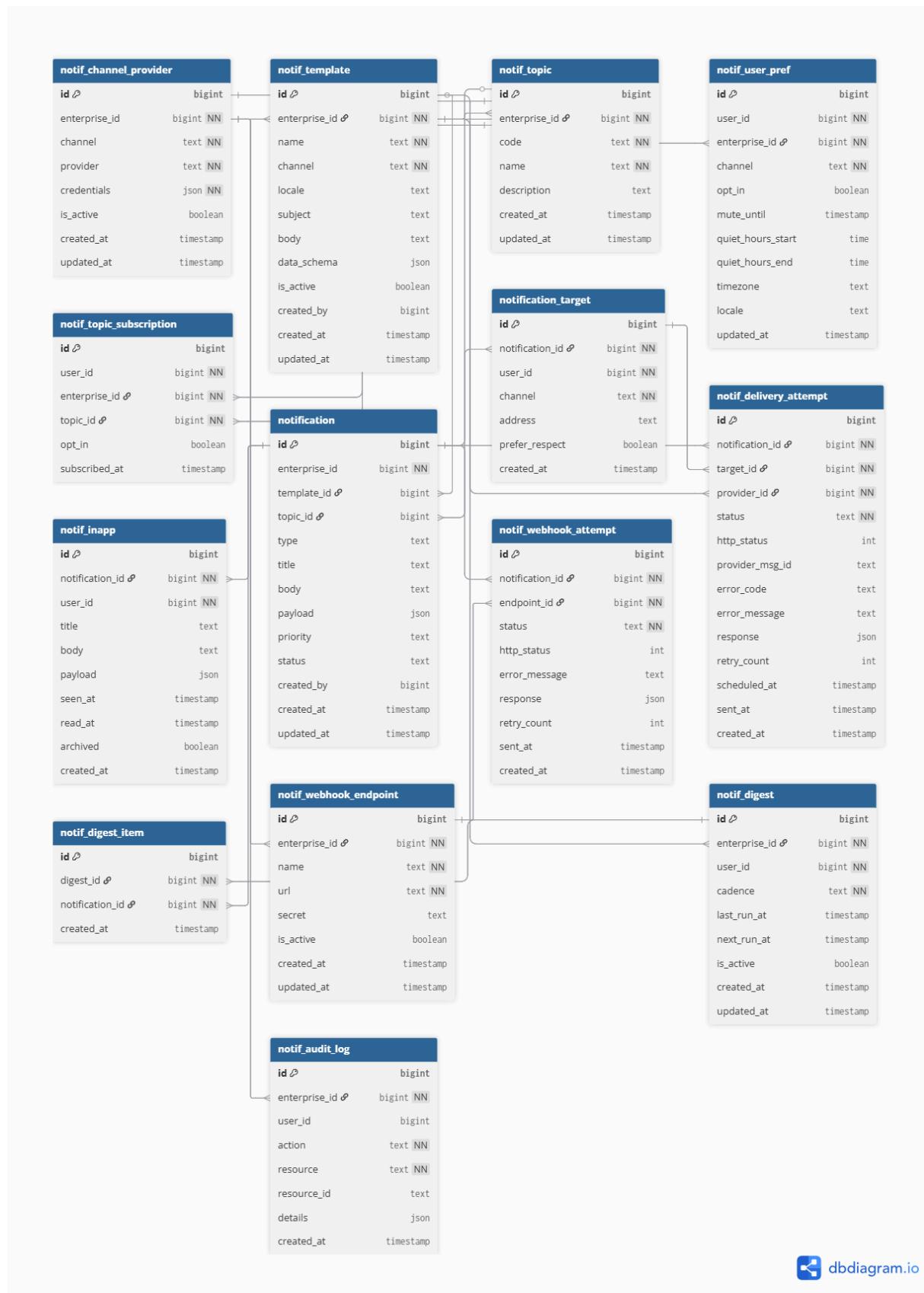




DIAGRAM-WORKBOARD-MicroService



dbdiagram.io



DIAGRAM-USERS-MicroService





7. Delivery Format

Since we are using a microservices-based architecture, all work can be found within the organization <https://github.com/IngSoftII-DayroGol-Asistencia> in the .git repository, where it can be viewed both in PDF and Markdown formats.