End-Of-Studies Project

In partial fulfillment of the requirements for a

Master's Degree

In

Software Engineering

Presented by

Riadh Laabidi

Notification Center

Defended on September 2023 in front of the committee members:

Mr.	President
Mrs.	Examiner
Mrs. Asma Kerkeni	Thesis Supervisor
Mr. Fahmi Boumaiza	Professional Supervisor

———— **Academic Year: 2022 / 2023** ————

For everyone who has ever been a part of my life.

I dedicate this work.

Abstract

This report outlines the design and implementation of a notification center as a part of the "Convergence" project, which aims to drive digital transformation within the insurance and finance sectors while improving client engagement and optimizing internal processes. The notification center provides a scalable solution for managing and delivering tailored notifications through various channels, facilitating timely and personalized interactions. This project was made during an end-of-studies internship in order to obtain the Master's degree in Software Engineering at the Higher Institute of Informatics and Mathematics of Monastir (ISIMM).

Keywords: Client Engagement, Notification Channels, User Segmentation, Spring Boot, Angular, PostgreSQL, Docker.

Acknowledgment

I would like to express my sincere gratitude and appreciation to my academic supervisor, **Mrs. Asma Kerkeni**, for her invaluable guidance, insightful advice, and unwavering support throughout this internship journey. Her expertise, enthusiasm, and dedication have been a constant source of inspiration and motivation.

I would also like to extend my deepest thanks to my technical supervisor, **Fahmi Boumaiza**, for his technical mentorship, and guidance throughout the project. His knowledge and expertise in software development were of great help in shaping this project and ensuring its success, and I have learned a lot from him.

I am grateful to both my supervisors for providing me with the opportunity to work on this challenging and exciting project, which has enriched my knowledge and skills in software development. Their support and mentorship have been crucial to my academic and professional growth.

Finally, I would like to thank the entire team at **Satoripop** for their cooperation, encouragement, and support throughout this journey. Their insightful feedback and suggestions made my work enjoyable and productive.

Contents

List of Figures

List of Tables

Acronyms

CI/CD Continuous Integration and Continuous Delivery.

CRUD Create, Retrieve, Update, and Delete.

CTR Click-through Rates.

DAO Data Access Object.

HTTPS Hypertext Transfer Protocol Secure.

MVVM Model-View-ViewModel.

UML Unified Modeling Language.

Introduction

In today's rapidly evolving digital landscape, the insurance, banking, and finance sectors are facing unprecedented challenges and opportunities. As customers increasingly expect seamless and personalized experiences, it has become imperative for companies in these industries to undergo a digital transformation. This shift not only enhances customer engagement and satisfaction but also boosts operational efficiency and employee productivity.

Recognizing this need, the "Convergence" project was initiated to provide innovative solutions that facilitate the digital transformation process, enabling organizations to stay competitive, deliver exceptional customer experiences, and achieve sustainable growth in the ever-changing marketplace.

Within the "Convergence" ecosystem, various tools and sub-projects are assembled, each playing a crucial role in this shared mission. Among these modules, we will be focusing on the Notification Center, as it plays a pivotal role in enhancing communication by providing a centralized system for timely and relevant notifications.

This report is structured in several chapters to provide a comprehensive overview of the project's progression. It begins with an introduction to the project context, including a profile of the host organization and an overview of the project itself. The challenges and problems faced in the insurance and finance sectors are also assessed.

The second chapter discusses the initial sprint of the project, which revolves around gathering requirements, defining specifications and setting the groundwork to start the project execution.

Subsequent chapters delve into the specific releases of the project, outlining the specification, design and implementation steps. Each chapter provides a summary of the key findings and progress made during the respective phase.

Chapter I

Project Context

Introduction

This chapter presents the project context, including an overview of the host organization, the project objectives and the problem assessment. Afterwards, we will benchmark some existing solutions, and select a suitable work methodology for a successful project execution. By exploring the context surrounding the project, we aim to provide a deeper understanding of its relevance within the organization and the industry as a whole.

I.1   Host organization profile

Satoripop is a custom software development house that delivers a wide range of end-to-end, reliable software services and solutions for businesses across various industry verticals. Satoripop offers its services in many countries in Europe, Middle East and Africa.

Figure I.1: Satoripop Logo

Satoripop has structured its offering around four key service areas. As a solutions provider, and given that all interactions nowadays pass through IT systems, Satoripop is more than ever at the heart of customers' business.

Services	• **Custom development:** Web applications, Mobile applications, Application modernization • **Design UX/UI:** SEO, Netlinking, Optimization and assistance with content creation, Reporting • **Consulting & Framing:** Design thinking, Customer journey map, Wireframing, Prototypage • **Digital Marketing:** SEO, Netlinking, Optimization and assistance with content creation, Reporting
Phone number	Tunisia: +216 73 210 332
Address	Satoripop MEA, Blvd Hassouna Ayachi, Sousse 4000, Tunisia
Website	`https://www.satoripop.com`

Table I.1: Host Organization Details

I.2 Project overview

A notification center is a system that allows businesses to send and manage notifications to their customers or employees. The value proposition of a notification center is that it can help businesses communicate with their stakeholders more efficiently and effectively, leading to increased engagement and productivity.

In the retail sector as an example, a company could use the notification center to send SMS, email, push or chat alerts about product recalls, special promotions, or other important information to customers. By using a notification center, businesses can improve communication with their customers and ensure that important information is quickly and effectively distributed.

I.3 Problem assessment and challenges

Implementing a notification center system in the retail sector poses certain challenges and requires a comprehensive assessment of the existing scenario. The following issues were identified during the evaluation:

- **Limited Communication Channels:** Many businesses in the retail sector rely heavily on traditional communication methods such as flyers, physical notices, or in-store announcements. These methods often lack efficiency, reach, and real-time delivery, resulting in delayed or ineffective communication.

- **Information Overload:** Businesses need to disseminate various types of notifications, like product recalls or special promotions. However, the challenge is to manage and prioritize notifications to prevent overwhelming customers with exces-

sive information.

- **Personalization and Targeting:** Effective communication requires tailoring notifications to specific customer segments or individuals. The challenge lies in ensuring that customers receive relevant and personalized notifications based on their preferences, previous interactions, or location.

- **Multichannel Delivery:** Customers today expect to receive notifications through various channels such as SMS, email, push notifications, or social media. Managing multiple communication channels and ensuring consistent and synchronized delivery presents a significant challenge.

- **Privacy and Data Security:** With the increasing concerns about privacy and data protection, businesses must handle customer data securely while complying with privacy regulations. Safeguarding customer information and maintaining trust is crucial in implementing a notification center system.

I.4 Competitor benchmarking

In this section, we will evaluate and compare existing notification delivery solutions. By examining the features, functionalities, and performance of these solutions, we aim to identify the strengths and weaknesses of each competitor in order to inform our own development process.

I.4.1 Competitors selection

During our search for similar existing functionalities, we have carefully evaluated numerous solutions and identified the three most closely aligned with our specific needs. We will present an overview of these chosen solutions, highlighting their key characteristics and capabilities.

I.4.1.1 OneSignal

OneSignal is a popular notification service that supports multiple platforms including iOS, Android, and web. OneSignal offers a wide range of features, including segmenting users based on custom attributes, A/B testing, automation, and personalization. OneSignal also provides real-time analytics and delivery reports, allowing tracking and optimization of notification campaigns.

Figure I.2: OneSignal Homepage

I.4.1.2 Pusher

Pusher is a cloud-based notification service that provides real-time messaging for mobile and web applications. It supports push notifications, in-app notifications, and allows integration with other services and platforms.

Figure I.3: Pusher Homepage

I.4.1.3 Amazon SNS

Amazon SNS (Simple Notification Service) is a fully managed messaging service provided by Amazon Web Services (AWS) that enables you to send messages or notifications to a variety of distributed endpoints or clients.

Figure I.4: Amazon SNS Service Homepage

I.4.2 Comparison

We will conduct a comprehensive analysis of the selected existing solutions. By thoroughly examining their functionalities, strengths, and weaknesses, we aim to gain valuable insights into areas where improvements can be made to enhance user experience, streamline processes, and deliver a superior solution tailored to our specific requirements.

Our evaluation will be based on 6 criterias (Cx) that have been carefully selected as being the most relevant and aligned with the Convergence project requirements:

- **C1: Multi-channel delivery:** The ability to configure multiple channels (Email, SMS, Push, Chat) for notification delivery.

- **C2: Deliverability:** Ensuring that notifications and alerts are successfully delivered to users' devices.

- **C3: User Segmentation:** The ability to select and target specific groups of users based on criterias defined by the business requirements.

- **C4: Customization:** The ability to customize notifications content for targeted users accordingly.

- **C5: Analytics:** The ability to track different metrics during the delivery process in order to inform businesses about the effectiveness of notification campaings and improve user engagement.

- **C6: Cost:** Charge per sent notifications for a larger user base and high notification volume.

7

- **C7: Ease of use:** The ability to easily setup and configure the different steps of the notification delivery process for non technical users.

	OneSignal	Pusher	Amazon SNS
C1 Multi-channel delivery	Provides most of channels (Push, Email, SMS), except for the Chat channel	Provides Push (Android, iOS, Web) notifications only	Does not provide native support for multi-channel notifications
C2 Deliverability	Low CTR and drops over time due to inactive receivers leading to misinformation for senders about the deliverability of their notifications.		
C3 User Segmentation			Does not support targeting specific segments of users based on their behavior or preferences.
C4 Customization	**Limited**: Users can't fully customize the layout of the notification.	**Limited**: Only a subject and a body, no dynamic content.	**Limited**: Only text message, subject and body, no dynamic content.
C5 Analytics			No detailed insights on sent notifications
C6 Cost	**249$/month** (for 50,000 subscribers from push notifications only)	**99$/month** for 50,000 subscribers	Charge per usage (Pay as you go)
C7 Ease of use		Mostly Programmatic, does not provide a fully featured web interface.	Complex and time-consuming particularly for non technical users.

Table I.2: Comparison table

I.5   Proposed solution

The benchmarking process allowed us to identify pain points and challenges commonly faced by businesses when implementing a notification center system.

Based on these findings, we have concluded that developing a custom notification center tailored to our specific needs is the most suitable approach. By addressing the identified pain points and challenges, we can create a robust and user-friendly solution that empowers businesses to communicate more efficiently with their customers and employees.

The proposed notification center will incorporate multi-channel delivery options, advanced user segmentation, and personalized messaging capabilities. It will also prioritize customization and prioritization, enabling businesses to tailor notifications based on audience preferences and deliver essential information effectively.
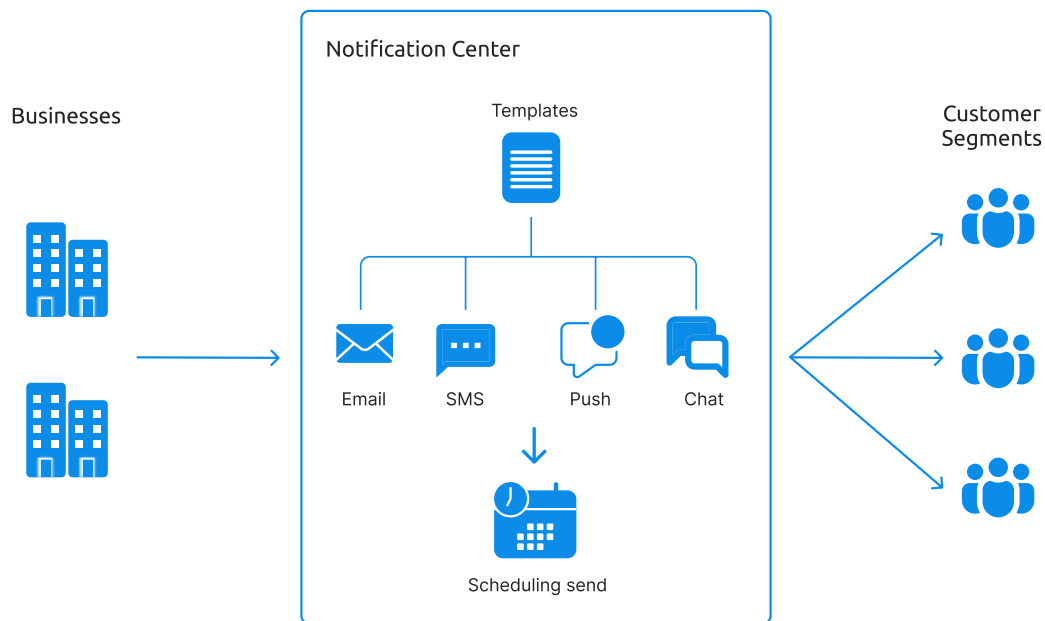
Figure I.5: Proposed solution illustration

I.6   Work methodology

With today's customers and businesses requiring rapid responses and changes, teams at Satoripop are embracing the Agile methodology, a project management approach that involves breaking the project into phases and emphasizes continuous collaboration and improvement, following a cycle of planning, executing, and evaluating.

One of the frameworks that helps practice building the mentioned agile priciples into work and that we will be focusing on is Scrum. we will provide an overview of this framework, highlighting its approach and key principles.

I.6.1 The scrum framework

Scrum is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems. It employs an iterative, incremental approach to optimize predictability and to control risk, while engages groups of people who collectively have all the skills and expertise to do the work and share or acquire such skills as needed.

I.6.2 Members of a scrum team

The fundamental unit of Scrum is a small team of people. The Scrum Team consists of one Scrum Master, one Product Owner, and Developers. Within a Scrum Team, there are no sub-teams or hierarchies. It is a cohesive unit of professionals focused on one objective at a time, the Product Goal. [1]

- **Developers:** People in the Scrum Team that are committed to creating any aspect of a usable Increment each Sprint.

- **Product Owner:** One person that may represent the needs of many stakeholders in the Product Backlog. The Product Owner is accountable for effective Product Backlog management and maximizing the value of the product resulting from the work of the Scrum Team.

- **Scrum Master:** Deeply understands the work being done by the team and can help the team optimize their transparency and delivery flow. As the facilitator-in-chief, he/she schedules the needed resources (both human and logistical) for sprint planning, stand-up, sprint review, and the sprint retrospective.

I.6.3 Scrum events

The Sprint is a container for all other events. Each event in Scrum is a formal opportunity to inspect and adapt Scrum artifacts. These events are specifically designed to enable the transparency required.

- **The Sprint:** Sprints are the heartbeat of Scrum, where ideas are turned into value. They are fixed length events of one month or less to create consistency. A new Sprint starts immediately after the conclusion of the previous Sprint.

- **Sprint Planning** The work to be performed during the current sprint is planned during this meeting by the entire development team. Specific user stories are then added to the sprint from the product backlog. These stories always align with the goal and are also agreed upon by the scrum team to be feasible to implement.

- **Daily Scrum:** A 15-minute event for the Developers of the Scrum Team held every working day of the Sprint to inspect progress toward the Sprint Goal and adapt the Sprint Backlog as necessary.

- **Sprint Review:** The purpose of the Sprint Review is to inspect the outcome of the Sprint and determine future adaptations. The Scrum Team presents the results of their work to key stakeholders and progress toward the Product Goal is discussed.

- **Sprint Retrospective:** The purpose of the Sprint Retrospective is to plan ways to increase quality and effectiveness. The Scrum Team identifies the most helpful changes to improve its effectiveness and these changes are addressed as soon as possible.

I.6.4   Scrum artifacts

Scrum's artifacts represent work or value. They are designed to maximize transparency of key information. Thus, everyone inspecting them has the same basis for adaptation.

- **Product Backlog:** The primary list of work that needs to get done. It's a dynamic list of features, enhancements, and fixes that acts as the input for the sprint backlog. This list is constantly revisited, re-prioritized and maintained by the Product Owner in case items may no longer be relevant or problems may get solved in other ways.

- **Sprint Backlog:** The list of items, user stories, or bug fixes, selected by the development team for implementation in the current sprint cycle. A sprint backlog may be flexible and can evolve during a sprint without compromising the fundamental sprint goal.

- **Increment:** A concrete stepping stone toward the Product Goal. Each Increment is additive to all prior Increments and thoroughly verified, ensuring that all Increments work together. In order to provide value, the Increment must be usable.

Summary

To summarize our findings in this chapter, we introduced the host organization, then outlined the problem assessment and identified its related challenges. Next we analyzed competitors to get a grasp of the major pain points that we are going to tackle, and finally we presented the adopted agile approach in our team to foster flexibility and collaboration.

The next chapter will introduce the Sprint 0, where we kickstart the project, establish goals, and set the roadmap for upcoming sprints.

Chapter II

Sprint 0: Requirements gathering and Specification

Introduction

In this chapter, we place significant focus on gathering requirements and transforming them into well-defined and documented specifications. This includes creating use cases and user stories, which provide valuable insights into system interactions and user workflows. Furthermore, we establish a prioritized product backlog, ensuring efficient resource allocation and timely delivery of the most critical and valuable features.

II.1   Requirements gathering

By conducting a comprehensive analysis of the requirements, we aim to bridge the gap between the stakeholders' vision and the actual implementation of the product or system. This analysis helps us define clear and concise specifications that serve as the foundation for the design, development, and testing phases of the project.

II.1.1   Identifying end-users

Identifying end-users is a crucial step in requirements analysis as it helps determine the needs, expectations, and constraints of the target audience. In our application, we were able to identify three types of actors:

- **Administrator**: Is responsible for overseeing user accounts, configuring and maintaining the resources needed by agents or employees to ensure the smooth operation of the system inside the organization.

- **Agent**: Is an employee inside the organization in charge of executing actions on the notification center: creating and scheduling notification campaigns including the creation of notification content, client base segmentation.

- **Client**: Is a customer who is going to be targeted by notifications from the business or organization they belong to. A customer should be able to receive notifications and set their preferences for receiving notifications from that business.

II.1.2 Functional requirements

Functional requirements define the specific actions, tasks, and behaviors that the product must be able to perform in order to meet the needs of its end-users. These requirements form the foundation of the system's functionality.

The functional requirements we captured for each actor in are outlined below.

Authentication and Profile settings

- **Sign in:** A registered user should be able to access the system by providing valid credentials.

- **Edit profile settings:** A logged in user should be able to edit his profile settings.

- **View statistics:** A logged in user should be able to view notification activity metrics on his dashboard.

Administrator requirements

- **Manage agents:** An administrator should be able to add new agents, update, delete, desactivate accounts and reset passwords for existing agents.

- **Manage channels:** An administrator should be able to create new notification channels, update, delete, configure service providers for existing channels.

- **Manage topics:** An administrator should be able to create new notification topics, update, delete and configure topic's priority for existing ones.

Agent requirements

- **Manage templates:** An agent should be able to create new notification templates, update and delete existing ones.

- **Manage triggers:** An agent should be able to create new notification triggers, con-

figure the target audience the scheduling, update, change status and delete existing triggers.

- **Manage audiences:** An agent should be able create new segments of users based on a criteria, update and delete existing ones.

- **View logs:** An agent should be able to view logs of sent notifications and their statuses.

Client requirements

- **Manage notification preference:** A client should be able to edit his notification preferences, channels and frequency of receiving notifications.

- **View notification history:** A client should be able to checkout a history of his received notifications (for in-app notifications).

II.1.3   Non-Functional requirements

When designing a notification system, various technical requirements need to be considered to ensure its effectiveness, reliability, and scalability. Here are the key requirements that should be addressed:

- **Security:** The system shall enforce secure communication protocols, such as HTTPS, to protect sensitive data during transmissionn, also data and preferences stored in the system shall be securely encrypted to prevent unauthorized access or data breaches.

- **Real-time:** The system shall deliver notifications in real-time or near real-time to ensure timely communication, messages and notifications should be delivered with minimal delay for high priority topics.

- **Scalability:** The system should be designed to handle a high volume of concurrent users and notifications without compromising performance and the system architecture should be scalable, allowing for horizontal scaling by adding more servers or utilizing cloud-based infrastructure as the user base grows.

- **Customizability:** The system should provide flexibility and customizability to meet the specific branding and user experience requirements of different organizations. Also the system should allow customization of user preferences to provide a personalized experience.

II.2 Scrum implementation overview

This section provides an overview of the Scrum implementation in our project It includes a global use case diagram, showcasing the system's high-level functionalities, followed by the presentation of the product backlog and the planning of the sprints.

II.2.1 Global use case diagram

Using UML use case diagrams to model the requirements allows for a visual representation of the interactions between actors and the system, providing a clear and concise way to specify the functionalities and behaviors expected from the product.

The figure II.1 illustrates the global use case diagram we modeled for our notification system:
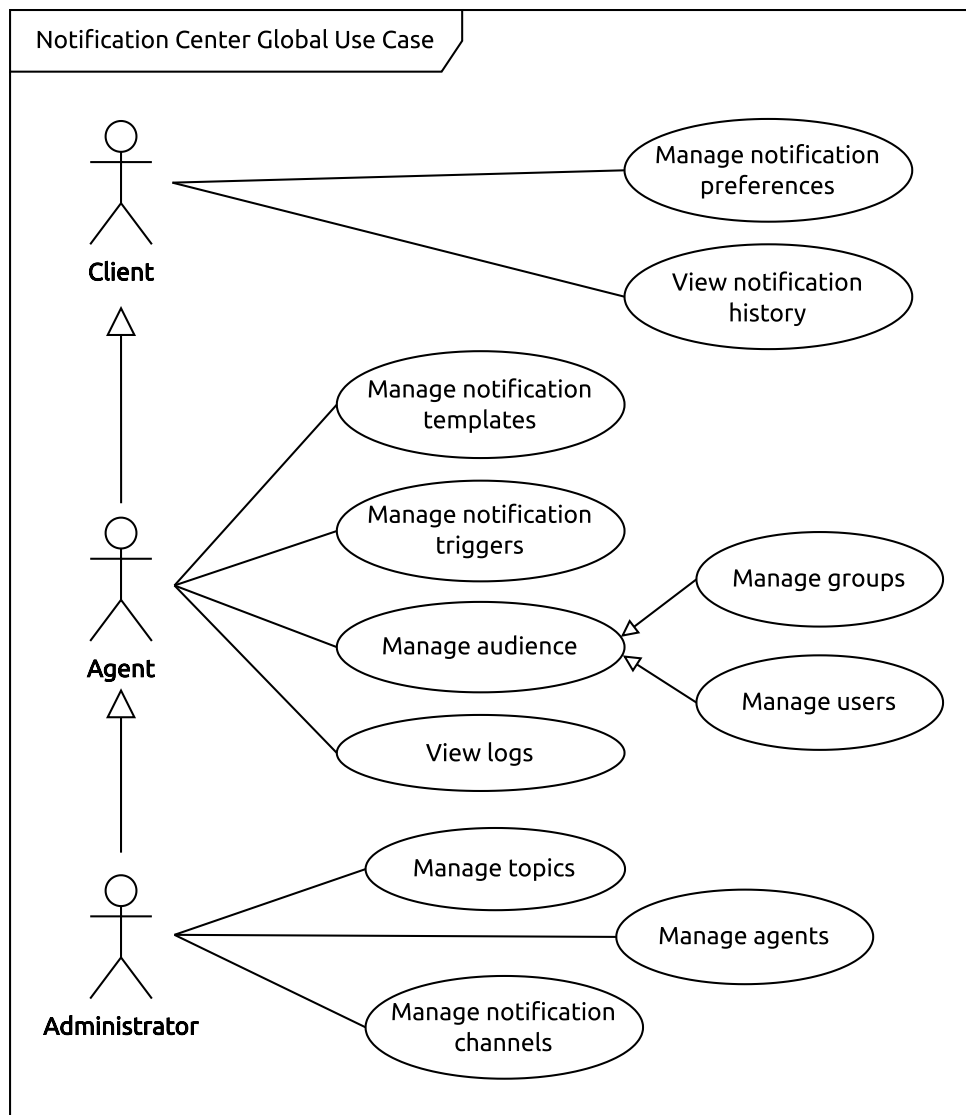

Figure II.1: Notification Center global use case diagram

II.2.2   Product backlog

The Product Backlog serves as a dynamic and living artifact that captures and organizes the ever-evolving list of features, functionalities, and enha ncements desired for a software product.

- **Epic:** Is a large body of work that can be broken down into a number of smaller stories.

- **User story:** Is an informal, general explanation of a software feature written from the perspective of the end user or customer, in the form of:
  "As a [persona], I [want to], [so that]."

Epic	User story
Authentication	As a new user, I want to be able to create an account so that I can use the notification center.
	As a registered user, I want to be able to log into my account securely using my email and password.
Agents management	As an administrator, I want to be able to create agents so that I can add them to the notification center.
	As an administrator, I want to be able to list agents so that I can view all registered agents.
	As an administrator, I want to be able to edit agents so that I can modify their information.
	As an administrator, I want to be able to delete agents so that I can get rid of no longer needed agents.
Users management	As an administrator/agent, I want to be able to create users so that I can send them notifications.
	As an administrator/agent, I want to be able to list users so that I can view all created users.
	As an administrator/agent, I want to be able to edit users so that I can modify their information.
	As an administrator/agent, I want to be able to delete users so that I can get rid of no longer needed users.
Audience management	As an administrator/agent, I want to be able to create an audience so that I can target specific individuals based on a criteria.
	As an administrator/agent, I want to be able to list audiences so that I can view all created segments.
	As an administrator/agent, I want to be able to edit an audience so that I can change selection criteria and segment configurations.

Epic	User story
	As an administrator/agent, I want to be able to delete a group of users so that I can get rid of no longer needed groups..
Notification channels management	As an administrator/agent, I want to be able to create notification channels so that I can send notifications through these channels.
	As an administrator/agent, I want to be able to list notification channels so that I can view all created channels.
	As an administrator, I want to be able to edit notification channels so that I can modify or update their configurations.
	As an administrator/agent, I want to be able to delete notification channels so that I can get rid of no longer used channels.
Notification templates management	As an agent, I want to be able to add notification templates so that I can send notifications based on that template.
	As an agent, I want to be able to list notification templates so that I can view all created templates.
	As an agent, I want to be able to edit notification templates so that I can keep them up to date.
	As an agent, I want to be able to delete templates so that I can get rid of no longer used templates.
Notification preferences management	As a user, I want to be able to set my notification preferences, so that I can receive notifications from the channels I want.
Notification triggers management	As an agent, I want to be able to create triggers for notifications so that I can schedule notifications to be sent automatically.
	As an agent, I want to be able to list triggers for notifications so that I can view all created triggers.
	As an agent, I want to be able to edit notifications triggers so that I can modify or update its configurations.
	As an agent, I want to be able to delete notification triggers so that I can get rid of outdated and no longer used triggers.
Notification history	As a user, I want to be able to list notification history so that I can review my received notifications whenever I want.
	As an administrator/agent, I want to be able to list sent notification logs so that I can review all sent notifications.
Dashboard	As an administrator/agent I want to be able to view metrics on my dashboard so that I can get an overview on important statistics related to notification activities.

Table II.1: Product backlog

18

II.2.3 Releases and sprints planning

A release refers to the deployment of a specific version or update of a software product. It involves meticulous planning and coordination to ensure successful delivery.

In the following table, we present the detailed planning of the releases, outlining the sprints included in each release, and the epics included in each sprint.

Release	Sprints	Epics
Release 1	Sprint 1	Authentication
		Agents management
		Users management
	Sprint 2	Audiences management
		Notification channels management
Release 2	Sprint 3	Notification templates management
		Notification preferences management
	Sprint 4	Notification triggers management
		Notification history
		Dashboard

Table II.2: Releases and sprints planning

II.3 Development infrastructure

In this section, we will discuss the development infrastructure established for the successful execution of the project. The primary focus will be on the software architecture. Additionally, we will elaborate on the database choice and the curated set of tools and frameworks that were employed to support and streamline the development workflow.

II.3.1 Software architecture

To implement our notification center, we will base our work on a client-server architecture. In this approach, the client, representing the user interface, interacts with the server, which houses the application's core logic and data. This separation of concerns allows for centralized management of data and business logic while catering to various clients.

One of the extensions to the mentioned architecture, and the one we will adopt is the **three-tier monolithic architecture**, where all the application modules and components are tightly integrated into a single codebase. This decision was based on several

factors such as the project's scope, timeline, and the team's familiarity with monolithic architectures. Furthermore, for the envisioned requirements, a monolithic design offered simplicity and ease of development and deployment to quickly get the application started.

The figure II.2 shows the separation of the three tiers in the monolithic architecture :
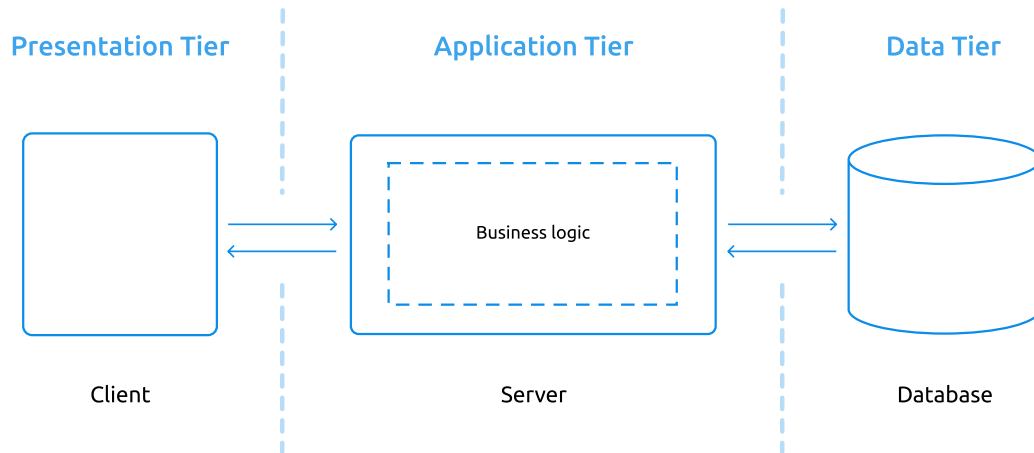

Figure II.2: Monolithic three-tier architecture

Presentation tier: The presentation tier is the user interface and communication layer of the application. Its main purpose is to display information to and collect information from the user. This top-level tier can run on a web browser, as desktop application, or a graphical user interface.

Application tier: In this tier, information collected in the presentation tier is processed, sometimes against other information in the data tier, using business logic, a specific set of business rules. The application tier can also add, delete or modify data in the data tier.

Data tier: The data tier, sometimes called database tier, data access tier or back-end, is where the information processed by the application is stored and managed. This can be a relational database management system or a NoSQL Database server.

The user interface, business logic, and the database all reside on different machines and, thus, have different tiers. They are physically separated.

II.3.2 Database

Selecting the appropriate database is a pivotal decision in our system design. After careful consideration, we've chosen to employ a relational database, specifically PostgreSQL. This choice is driven by the following criteria:

- **Structured Data:** We need to store notification channels and templates, user segments, and scheduling details in structured tables, these entities share relationships

that interconnect them.

- **ACID Compliance:** We need to ensure data integrity and consistency, as it's critical for reliable notification delivery and management.

- **Query Flexibility:** We need to leverage SQL queries to effectively retrieve and manipulate Data, enabling comprehensive reporting and analysis.

PostgreSQL is an open source object-relational database system that has a strong reputation for its reliability, flexibility, and support of open technical standards. PostgreSQL comes with many features aimed to help developers build applications and help managing higher data loads. In addition to being free and open source, PostgreSQL is highly extensible.

II.3.3   Server-side technology

This section delves into the chosen technology for the backend part of our solution and the internal architectural pattern.

II.3.3.1   Spring Boot framework

Spring Boot emerged as the optimal backend technology for our project. Spring Boot is built upon the robust Spring framework, makes developing production-grade Spring based web application and microservices faster and easier through three core capabilities:

- **Auto Configuration:** Spring Boot comes with built-in autoconfiguration capabilities, it automatically configures both the underlying Spring Framework and third-party packages based on the provided settings.

- **Opinionated approach:** Spring Boot chooses which packages to install and which default values to use,rather than requiring the developer to make all those decisions on their own and set up everything manually.

- **Standalone applications:** Spring Boot helps developers create standalone applications that run on their own, without relying on an external web server, by embedding a web server such as Tomcat into the application during the initialization process.

II.3.3.2   Spring boot flow architecture

Spring Boot uses a hierarchical architecture in which each layer communicates with the layer immediately below or above it as show in the following figure II.3
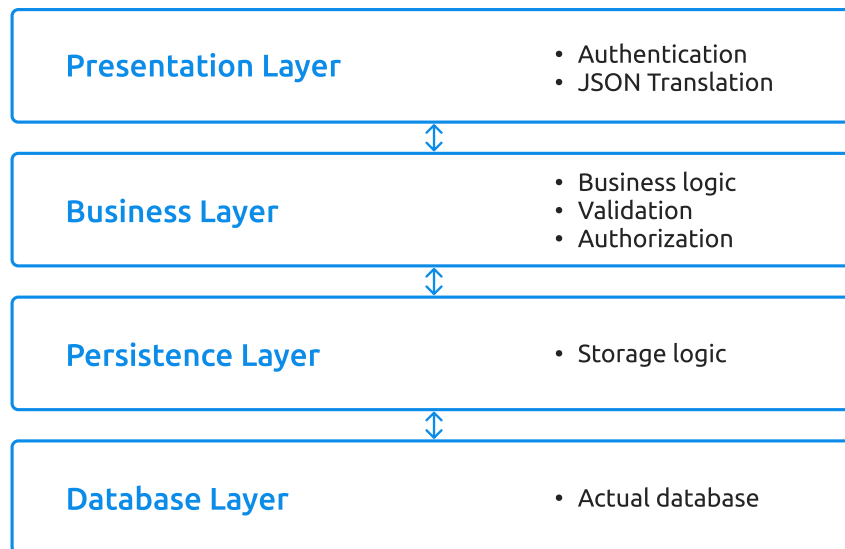
Figure II.3: Spring Boot application layers

Spring Boot integrates almost all of the features and modules of the Spring framework: like Spring MVC, Spring Core, Spring Data & JPA, etc. Spring Boot follows nearly the same architectural pattern as Spring MVC, except for one thing: there is no need for Data Access Object (DAO) and DAO Implementation classes in Spring boot because as we mentioned in figure II.3 its architecture has a data access layer and performs CRUD operations.
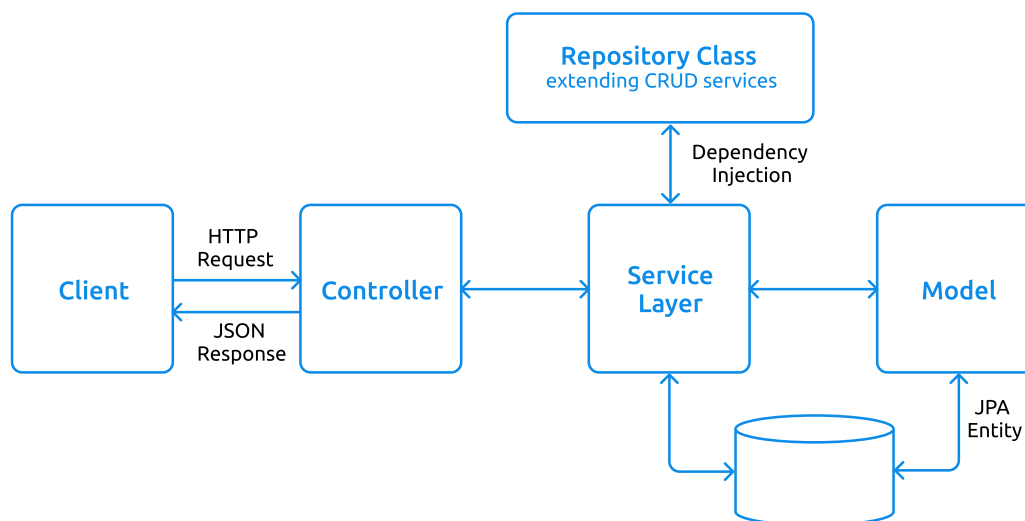

Figure II.4: Spring Boot flow architecture

II.3.4 Client-side technology

This section illustrates the chosen technology for the client side of our solution and the adopted internal architectural pattern.

II.3.4.1 Angular framework

For the user facing part of our solution, we will select the Angular frontend framework. Angular, developed and maintained by Google, is a development platform, built on TypeScript and includes:

- A component-based framework for building scalable web applications.

- A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, etc.

- A suite of developer tools to help develop, build, test, and update code. [2]

II.3.4.2 Angular architectural pattern

Angular follows a component-based architectural pattern that aligns closely with the Model-View-ViewModel (MVVM) pattern, a design paradigm that separates concerns in frontend development. While Angular's terminologies don't exactly match MVVM, the concepts map as follows:
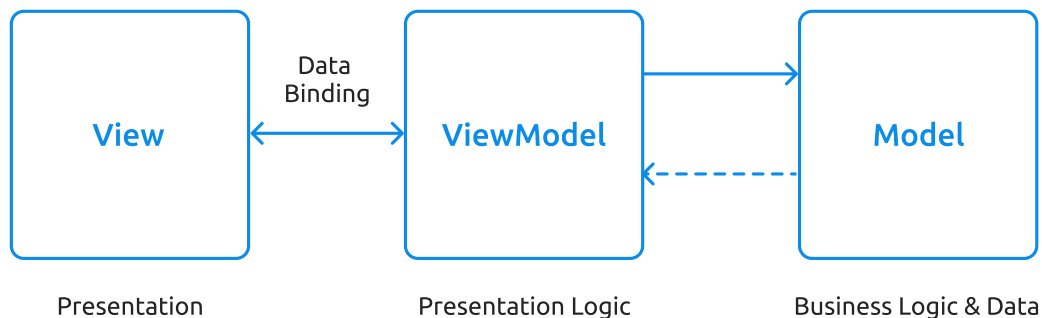
Figure II.5: MVVM architectural pattern

Model: Corresponds to business logic encapsulated within services and components. Services encapsulate data retrieval and manipulation, and act as the bridge between the application and external data sources.

View: Corresponds to the HTML templates in Angular. Templates define the structure and appearance of the user interface. Angular's binding mechanisms (property binding, event binding, and two-way binding) facilitate the synchronization between the model (data) and the view (UI).

ViewModel: Is represented by the component class. The component class contains the presentation logic, handling user interactions, processing data, and managing the application's state. It orchestrates the communication between the model and the view, ensuring that changes in one are reflected appropriately in the other.

II.3.5 JHipster Platform

In our quest to identify effective tools for accelerating the development of an enterprise-level solution while minimizing the burdens of boilerplate code and other repetitive tasks, we conducted a research. Our objective was to pinpoint a solution that not only expedites project initiation but also empowers our team to concentrate on the core application logic.

We have chosen JHipster as the foundation for our project development. JHipster is a development platform to quickly generate, develop, and deploy modern web applications and microservice architectures. [3]

The value proposition that JHipster provides is reflected in:

- The creation of fully configured applications for both the front and the back ends. In addition to the generation of code according to best practices and coding standards, ensuring a high level of code quality and consistency throughout the project.

- JHipster's integrated DevOps and CI/CD capabilities provide a seamless pathway to automate testing, deployment, and delivery processes.

In conjunction with JHipster, we employed various complementary software tools to enhance the development process. (see Appendix A)

Summary

In this chapter we laid the groundwork for the project by comprehensively gathering and analyzing the requirements. By identifying end-users, defining functional and non-functional requirements, and creating a clear specification through use case diagrams, elaborating the product backlog, planning sprints, specifying the application's architecture and the requisite development infrastructure, we have set the stage for the subsequent phases of development.

The insights gained from this chapter will prove invaluable in ensuring the successful implementation of the project and the satisfaction of the end-users. In the following chapter, we will start executing our project providing the practical realization of the initial release.

Chapter III

Release 1: Users, segmentation & channels

Introduction

In this chapter, we focus on the first release of our product, this release aims to create and enhance the user experience, implement userbase segmentation strategies, and manage notification channels. We will delve into the sprints backlog, design considerations, and the implementation process undertaken to bring these features to fruition.

III.1   Sprints backlog

During the Sprint Planning event, we estimated the effort required for each item in the sprint backlog based on the number of working hours.

The priority of backlog items is reflected by their relative order in the table. Items positioned higher in the table indicate a higher priority.

Sprint	Epic	User story	Estimation
1	Authentication	As a new user, I want to be able to create an account so that I can use the notification center.	16
		As a registered user, I want to be able to log into my account securely using my email and password.	16

Sprint	Epic	User story	Estimation
	Agents management	As an administrator, I want to be able to create agents so that I can add them to the notification center.	16
		As an administrator, I want to be able to list agents so that I can view all registered agents.	16
		As an administrator, I want to be able to edit agents so that I can modify their information.	8
		As an administrator, I want to be able to delete agents so that I can get rid of no longer needed agents.	8
	Users management	As an administrator/agent, I want to be able to create users so that I can send them notifications.	16
		As an administrator/agent, I want to be able to list users so that I can view all created users.	16
		As an administrator/agent, I want to be able to edit users so that I can modify their information.	8
		As an administrator/agent, I want to be able to delete users so that I can get rid of no longer needed users.	8
2	Audience management	As an administrator/agent, I want to be able to create an audience so that I can target specific individuals based on a criteria.	24
		As an administrator/agent, I want to be able to list audiences so that I can view all created segments.	16
		As an administrator/agent, I want to be able to edit an audience so that I can change selection criteria and segment configurations.	8
		As an administrator/agent, I want to be able to delete a group of users so that I can get rid of no longer needed groups..	8

Sprint	Epic	User story	Estimation
	Notification channels management	As an administrator/agent, I want to be able to create notification channels so that I can send notifications through these channels.	32
		As an administrator/agent, I want to be able to list notification channels so that I can view all created channels.	16
		As an administrator, I want to be able to edit notification channels so that I can modify or update their configurations.	8
		As an administrator/agent, I want to be able to delete notification channels so that I can get rid of no longer used channels.	8

Table III.1: Backlog of Sprint 1 & 2

III.2   Design

The design phase of software development is a critical step in the software development life cycle. It is the process of transforming the customer requirements into a detailed plan for how the software will be implemented. In this section we are going to focus on the design aspects related to the database and sodtware design.

III.2.1   Database design

The figure III.1 illustrates the entity-relationship diagram for our first release.
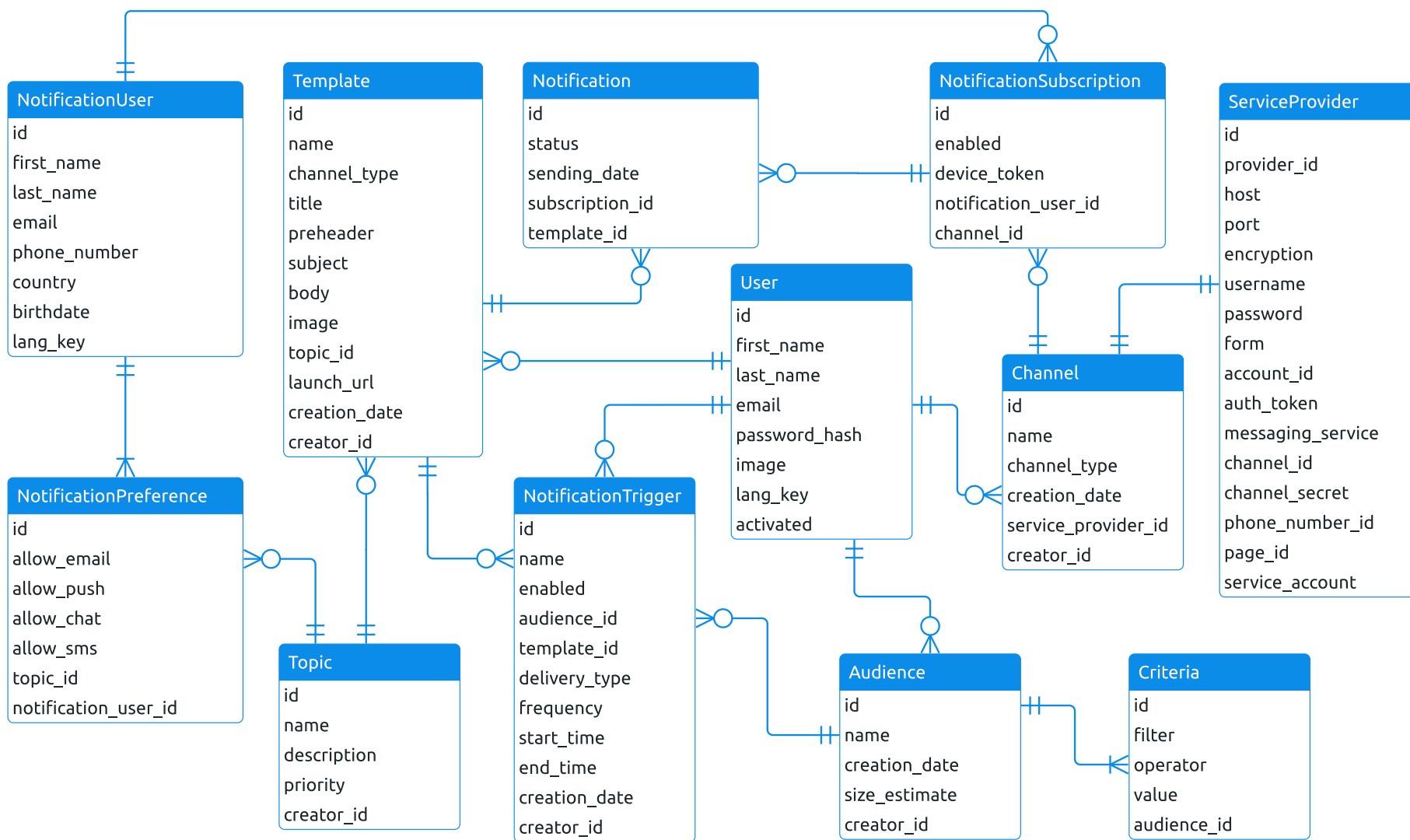
Figure III.1: Notification center entity-relationship diagram

III.2.2   Software design

III.2.2.1   Class diagram

The class diagram provides a high-level overview of the system's structure, relationships, and interactions between different classes. It represents the static structure of the system, including classes, attributes, methods, and associations.

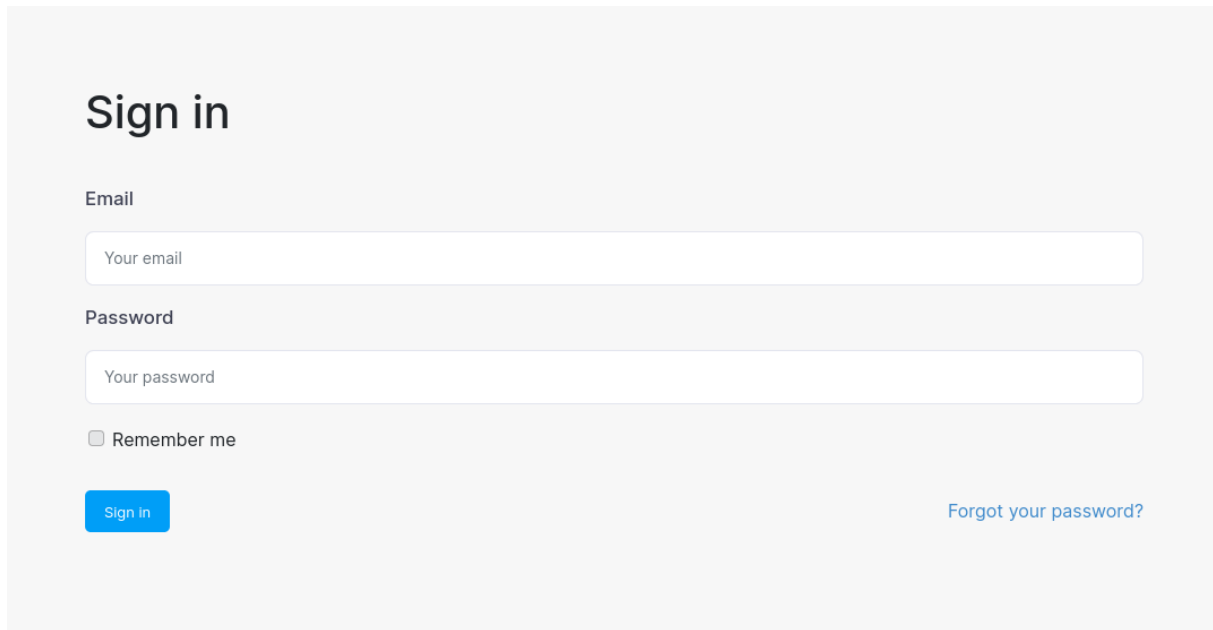The figure III.2 illustrates the class diagram for our first release.
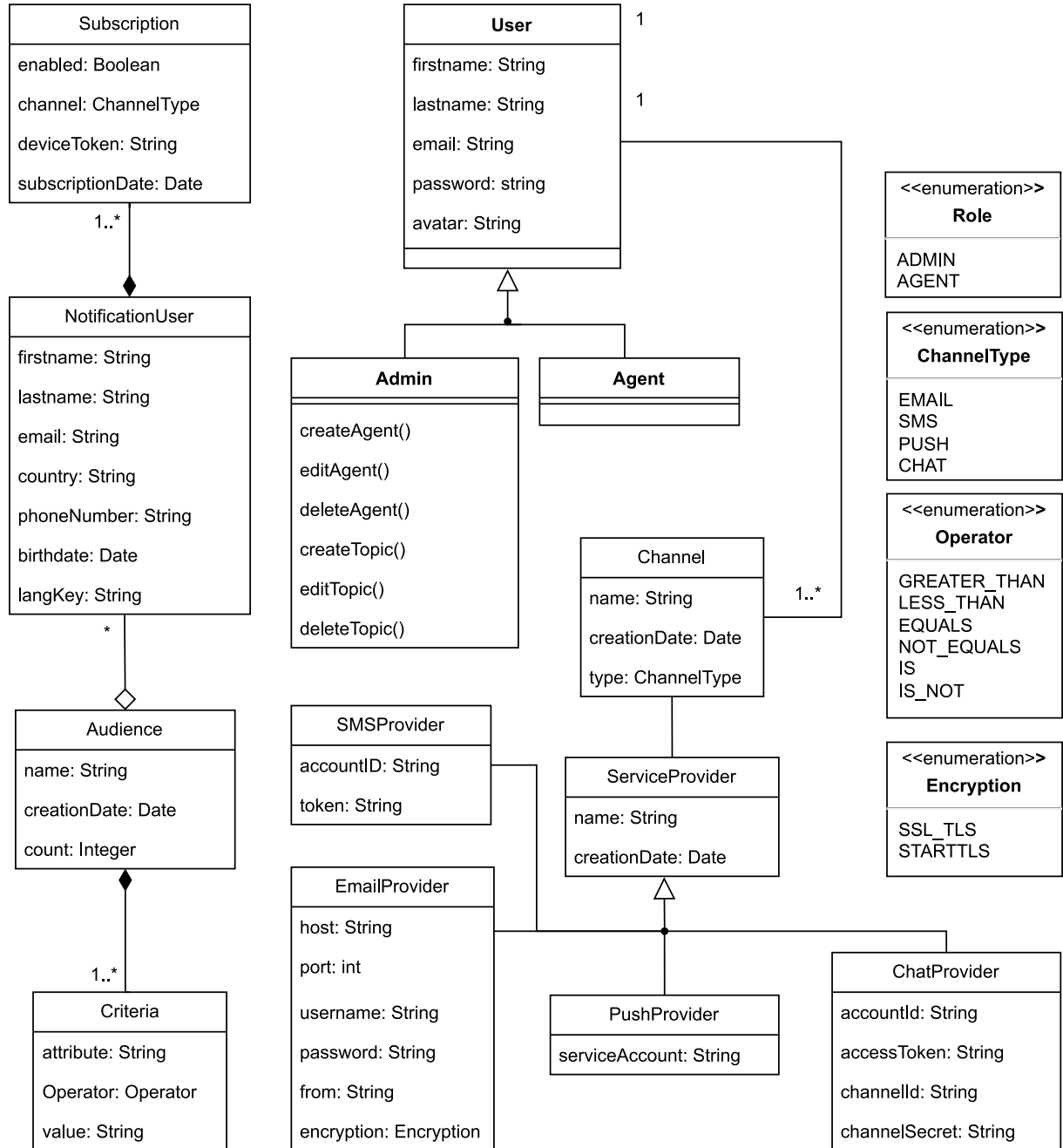
Figure III.2: Notification Center global class diagram

III.3 Implementation

III.3.1 Authentication page

The figure III.3 illustrates the final result of the implementation of the authentication epic.

Figure III.3: Authentication page

III.3.2 Agents management page

The figure III.4 illustrates the final result of the implementation of the agents management epic.
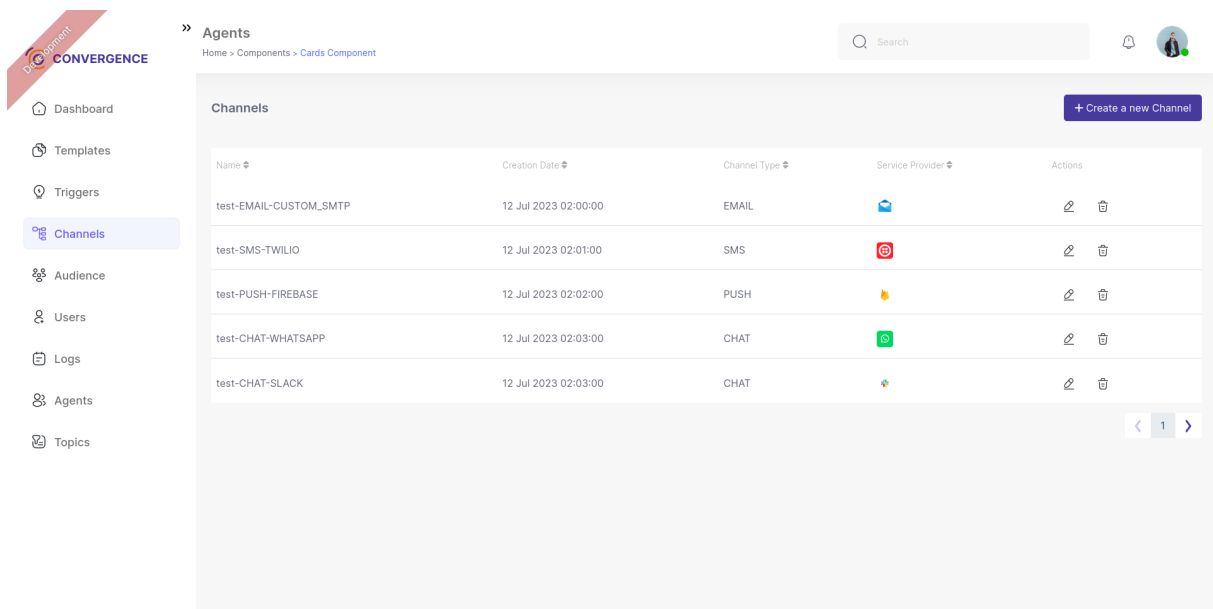
III.3.3 Channels management page

The figure III.5 illustrates the final result of the implementation of the channels management epic.

Summary
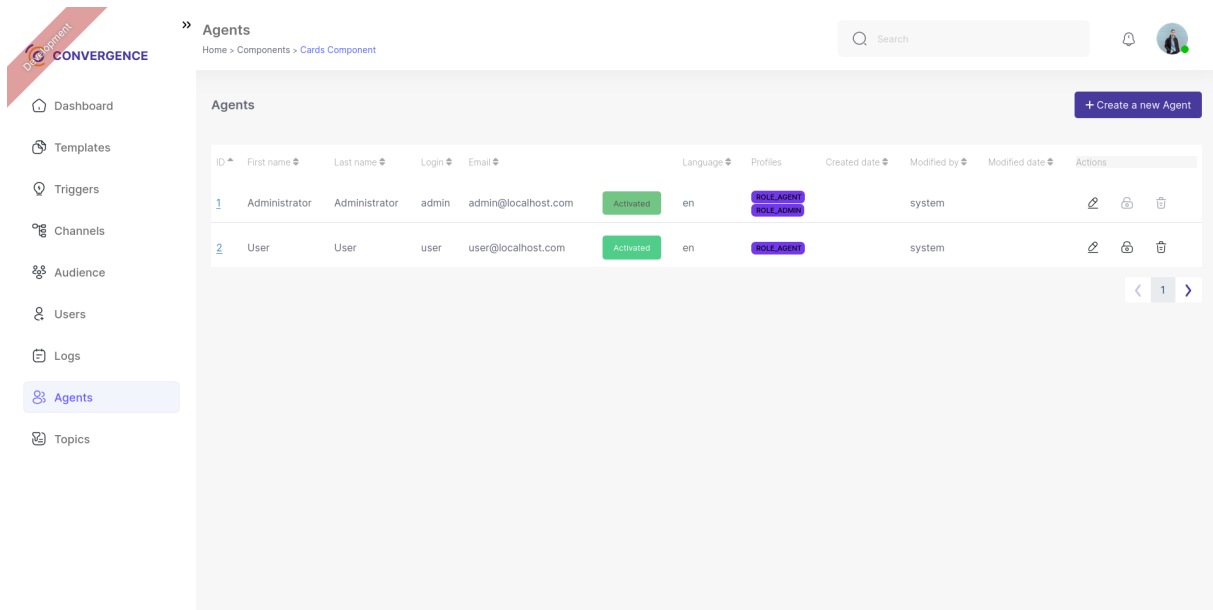
Figure III.4: Agents management page

Figure III.5: Channels management page

Chapter IV

Release 2: Sending notifications & dashboards

Introduction

IV.1   Sprint backlog

IV.2   Specification

IV.3   Design

IV.4   Implementation

Summary

Appendix A

Tools

In this section we present the set of software tools we used to support our development process.

	Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. We used docker to create containers for the database and other components we needed for development and also to fully dockerize the application and all the services that it depends on.
	Bitbucket is a Git based source code repository hosting and collaboration tool owned by Atlassian. We used this service to host the source code for our project.
	Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration to create APIs faster. We used postman to test and maintain our notification center API.
	IntelliJ IDEA is an integrated development environment for developing computer software written in Java, Kotlin, Groovy, and other JVM-based languages.
	Figma is a collaborative web application for user interface and user experience design, with an emphasis on real-time collaboration, utilising a variety of vector graphics editor and prototyping tools. We used Figma to create wireframes, UI designs and illustrtations for our solution.
	draw.io is a cross-platform graph drawing software that can be used to create diagrams such as flowcharts, wireframes, UML diagrams, etc. We used draw.io to create the uml diagrams during the software desgin process.
	Trello is a web-based visual work and tasks management tool. We used trello to keep track of and to collaborate on the development tasks during the execution of the project.

Table A.1: Software tools

Bibliography

[1] Ken Schwaber & Jeff Sutherland. *The Scrum Guide.* Nov. 2020. URL: https://
     scrumguides.org/scrum-guide.html (visited on 07/01/2023).

[2] *What is Angular?* Feb. 2022. URL: https://angular.io/guide/what-is-angular
     (visited on 08/20/2023).

[3] *What is JHipster?* URL: https://www.jhipster.tech/ (visited on 08/22/2023).