

TALLINN UNIVERSITY OF TECHNOLOGY  
School of Information Technologies

Artjom Ljuboženko 210740IADB

# **Open-source System for Localization and Translation Management**

Bachelor's thesis

Supervisor: Mohammad Tariq  
Meeran  
PhD in ICT

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL  
Infotehnoloogia teaduskond

Artjom Ljuboženko 210740IADB

# **Avatud lähtekoodiga süsteem lokaliseerimiseks ja tõlkehalduseks**

Bakalaureusetöö

Juhendaja: Mohammad Tariq  
Meeran  
PhD in ICT

Tallinn 2023

## **Author's declaration of originality**

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Artjom Ljuboženko

14.05.2023

## **Abstract**

Localization and translation services are becoming more important because of the global market reach, better customer experience, brand reputation and compliance. This research work focuses on the need for the development of a secure, customizable, and adaptable open-source localization and translation management system. The goal of the thesis was to develop an open-source system using all the best practices. Identify and prioritize the key requirements for the proposed system, ensuring it caters to the diverse needs of businesses while remaining versatile, efficient, and effective. Investigate the challenges and opportunities associated with the development and implementation of the open-source localization and translation management system, providing insights and recommendations to facilitate its successful adoption by businesses. After investigating and identifying the key requirements a process of system design and development process was started. The development process was divided into two separate phases, the server-side development, and the client-side development. After the system development was completed a process of feature testing was conducted. For key requirements case testing the all the necessary user flow for such system were provided by Swedbank Mobile Team. The flows included all the necessary steps for moderators and translators.

This thesis contributes by studying the need for a localization and translation system and development of a prototype based on the business requirements. The software development process was aligned with modern tech stack and open-source technologies which provide an opportunity for businesses to switch from the existing closed-source, cloud-based solutions to open-source, to an internally run system that will address the limitation of closed-source systems.

This thesis is written in English and is 30 pages long, including 40 chapters, 20 figures and 2 tables.

## **Annotatsioon**

### **Avatud lähtekoodiga süsteem lokaliseerimiseks ja tõlkehalduseks**

Lokaliseerimis- ja tõlketeenused muutuvad üha olulisemaks globaalse turu ulatuse, parema kliendikogemuse, kaubamärgi maine ja vastavuse tõttu. See lõputöö keskendub vajadusele välja töötada turvaline, kohandatav ja kohandatav avatud lähtekoodiga lokaliseerimis- ja tõlkehaldussüsteem. Lõputöö eesmärk oli arendada avatud lähtekoodiga süsteem, kasutades kõiki parimaid praktikaid. Teha kindlaks kavandatava süsteemi põhinõuded ja seada need prioriteediks, tagades, et see vastab ettevõtete erinevatele vajadustele, jäädes samas mitmekülgselt, tõhusaks ja tulemuslikuks. Uurida väljakutseid ja võimalusi, mis on seotud avatud lähtekoodiga lokaliseerimis- ja tõlkehaldussüsteemi arendamise ja rakendamisega, pakkudes teadmisi ja soovitusi, et hõlbustada selle edukat kasutuselevõttu ettevõtetes. Pärast põhinõuete uurimist ja tuvastamist alustati süsteemi projekteerimise ja arendamise protsessiga. Arendusprotsess jagunes kaheks erinevaks etapiks: serveripoolne arendus ja kliendipoolne arendus. Pärast süsteemi arendamise lõpetamist viidi läbi funktsioonide testimise protsess. Võtmenõuete juhtumite testimiseks tagas sellise süsteemi jaoks kogu vajaliku kasutajavoo Swedbanki mobiilimeeskond. Testid sisaldasid kõiki vajalikke samme moderaatorite ja tõlkijate jaoks.

Antud lõputöö aitab kaasa lokaliseerimis- ja tõlkesüsteemi vajaduse uurimisele ning ärinõuete lähtuva prototüübi väljatöötamisele. Tarkvaraarenduse protsess viidi kooskõlla kaasaegse tehnoloogiapaki ja avatud lähtekoodiga tehnoloogiatega, mis annavad ettevõtetele võimaluse lülituda olemasolevatelt suletud lähtekoodiga pilvepõhistelt lahendustelt avatud lähtekoodiga lahendustele sisemiselt juhitavale süsteemile, mis lahendab suletud lähtekoodiga süsteemid.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 30 leheküljel, 40 peatükki, 20 joonist, 2 tabelit.

## List of abbreviations and terms

CLI	Command Line Interface
CLOSED-SOURCE	Software with inaccessible source code, restricting modifications and distribution
CLOUD-BASED	Services that are running on external servers over the internet
DAO	Data Access Object classes, isolated from the rest of the application
GIT	Git is a distributed version control system that tracks changes in any set of computer files, usually used for coordinating work during software development.
JDK	Java Development Kit, a software package for developing Java applications, containing tools, libraries, and runtime environment
MVC	Model-View-Controller design pattern for building web applications
REST API	Representational State Transfer Application Programming Interface
TMS	Translation Management System

## Table of contents

1 Introduction .....	11
1.1 Problem statement .....	11
1.2 Research questions .....	12
1.3 Research goal.....	13
2 Literature review.....	13
2.1 Similar solutions .....	14
2.1.1 Closed-Source Solution: Crowdin .....	14
2.1.2 Open-Source Solution: Pootle .....	14
2.2 Software development methods.....	15
2.2.1 Hexagonal Architecture .....	15
2.2.2 Clean Code .....	15
2.2.3 Test-Driven Development (TDD) .....	15
2.3 Open-source Apache-2.0 license .....	16
3 Methodology.....	17
3.1 Comparative analysis.....	17
3.2 Requirement gathering and prioritization.....	17
3.3 Development planning.....	18
3.4 System testing.....	18
3.5 User acceptance testing .....	18
4 Software development .....	19
4.1 Server-side Architecture and Design .....	19
4.1.1 Application module .....	20
4.1.2 Git service module.....	21
4.1.3 Database Architecture and Design .....	21
4.2 Client-side Architecture and Design.....	22
4.3 Features Development .....	22
4.3.1 Server-side Features Development.....	25
4.3.2 Client-side Features Development .....	26
4.3.3 Security.....	27

4.4 Testing and validation .....	28
5 Result and analysis .....	29
5.1 Final look of the system.....	29
5.1.1 Login Form.....	29
5.1.2 Home Page.....	30
5.1.3 New Project Creation Page.....	31
5.1.4 Project Page with Settings, Working Groups and Labels.....	32
5.1.5 Label Management Page .....	35
5.1.6 New Working Group Creation with User Creation Page .....	36
5.2 System Evaluation and Impact .....	38
6 Conclusions and future work.....	39
References .....	41
Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis .....	42



## List of figures

Figure 1. Server-side architecture.....	20
Figure 2. Database ERD scheme. ....	22
Figure 3. User stories.....	23
Figure 4. Server-side happy flow .....	24
Figure 5. Client-side happy flow. ....	25
Figure 6. JWT generating and acquiring. ....	27
Figure 7. The login form page.....	30
Figure 8. The home page. ....	30
Figure 9. New project creation name page. ....	31
Figure 10. New project creation configuration page. ....	32
Figure 11. New project creation success page.....	32
Figure 12. The project page.....	33
Figure 13. The project settings. ....	33
Figure 14. The project working groups. ....	34
Figure 15. The project labels. ....	34
Figure 16. The label management page.....	35
Figure 17. The label notes. ....	36
Figure 18. New working group creation with existing user. ....	37
Figure 19. New working group creation with new user. ....	37
Figure 20. Translation Management System final architecture.....	38

## **List of tables**

Table 1. Server-side directory structure. ....	26
Table 2. Client-side directory structure. ....	27

# **1 Introduction**

This thesis addresses the need for a secure, customizable, and adaptable open-source localization and translation management system. Existing solutions may not fulfill specific business needs and can pose data privacy risks. The study explores the benefits of an internally-run system, key requirements, and challenges in development. Through a combination of theoretical investigations, requirement gathering, and system development, the research aims to create a platform that meets modern expectations and ensures a seamless transition for companies.

## **1.1 Problem statement**

Businesses now depend more on localization and translation management because of globalization. However, the issue is that the majority of the localization and translation management options on the market are closed-source, cloud-based platforms. Since these systems require sending data through external systems, they could be risky for businesses that are worried about safeguarding their confidential data.

Additionally, some businesses have particular needs that are not satisfied by pre-made options. They might have to alter the system to suit their particular requirements, such as integrating it with their current IT architecture, adhering to strict rules, or fitting it to particular business procedures. There is a need for more adaptable and customizable solutions because standard solutions might not be appropriate for everyone.

Therefore, there is an increasing need for an open-source localization and translation management solution that enables businesses to run such systems internally. Companies could adapt an open-source solution to their needs, combine it with their current IT infrastructure, and guarantee security and transparency. Additionally, it would offer more options and flexibility, enabling businesses to switch vendors with ease and prevent vendor lock-in.

The review of the literature shows that there is a need for a solution that can address the identified problem. Today, businesses have limited options and in order meet their business requirements they have to create their own internal systems, which can be costly and time-consuming. An open-source solution would give businesses a more flexible, open, and safe alternative to closed-source, cloud-based systems, which would help them to meet their demands and adjust to shifting business requirements.

## **1.2 Research questions**

How can an open-source, internally-run localization and translation management system address the challenges faced by companies when relying on external third-party systems?

This research question aims to explore the potential benefits of implementing an internal, open-source localization and translation management system, as opposed to relying on external systems. The focus will be on understanding how an open-source system can assist companies in resolving issues related to cost, data security, and customization while assessing the performance of an internally managed system in terms of control and adaptability to specific business requirements.

What are the key requirements for such a system and how can they be prioritized based on the needs of companies?

This research question seeks to identify the critical features and functionalities that an open-source, internally-run system should possess and understand how different companies might prioritize these requirements. This analysis aims to inform the development of a versatile, efficient, and effective system that addresses the diverse needs of businesses.

What challenges and opportunities can be encountered while developing an open-source localization and translation management system?

This research question investigates the potential problems and benefits associated with the development of an open-source localization and translation management system. It aims to understand the technical, organizational, and market-related factors that can impact the development process and the subsequent adoption of the system by companies.

### **1.3 Research goal**

Develop an open-source, adaptable localization and translation management system that addresses the limitations of external, closed-source, cloud-based systems, enabling companies to achieve greater control, adaptability, and security.

Identify and prioritize the key requirements for the proposed system, ensuring it caters to the diverse needs of businesses while remaining versatile, efficient, and effective.

Investigate the challenges and opportunities associated with the development and implementation of the open-source localization and translation management system, providing insights and recommendations to facilitate its successful adoption by businesses.

These research goals will be pursued through a combination of theoretical and analytical investigations, requirement gathering from potential users, prioritizing business cases, developing the system using modern technologies, and conducting user case tests to ensure the fulfillment of business requirements. The findings will contribute to the understanding of the current limitations of existing solutions and guide the development of a new platform that meets contemporary expectations and offers a seamless transition for companies.

## **2 Literature review**

The literature review will investigate the present condition of localization and translation management systems, scrutinizing both proprietary and open-source options in the market. Furthermore, it will delve into pertinent software development methodologies and techniques associated with building such systems. This thorough comprehension will offer insights into the difficulties companies encounter when handling these systems and the constraints of current solutions. By examining the pros and cons of different systems and development strategies, this review will lay the groundwork for creating a more robust and flexible open-source alternative.

## **2.1 Similar solutions**

This section will discuss existing localization and translation management solutions, evaluating their features, strengths, and weaknesses. By examining closed-source platforms such as Crowdin and open-source alternatives like Pootle, this analysis will identify gaps in the market and the need for an open-source, customizable, and up-to-date system that addresses data privacy concerns and offers seamless integration with modern technologies. [1] [2]

### **2.1.1 Closed-Source Solution: Crowdin**

Crowdin is a well-known closed-source, cloud-based solution that helps businesses handle their localization and translation needs. Companies can handle their localization projects effectively thanks to the platform it provides for cooperation between developers, translators, and project managers. Crowdin is a good option for businesses looking for a complete solution because it supports integration with a variety of platforms, file formats, and version control systems. [1]

For businesses that place a high priority on data protection and security, Crowdin's closed-source design might raise some concerns. The use of a third-party system increases the danger of data breaches and malicious attacks. A closed-source solution's intrinsic lack of customization and adaptability may also not be compatible with the requirements of some organizations.

### **2.1.2 Open-Source Solution: Pootle**

Pootle is an open-source system for localization and translation management. Pootle offers users the freedom to customize and adjust the system to meet their unique needs because it is an open-source solution. This flexibility can lead to a more tailored approach for businesses with unique localization needs. Despite it being open-source, Pootle has some restrictions. It was created using older technologies, which could make using it less convenient and outdated than using more recent solutions. For businesses looking for a comprehensive and future-proof solution, its limited support for integration with other platforms and the possible difficulties in maintaining and updating the system can be a problem. [2]

While existing solutions such as Crowdin and Pootle offer certain benefits for localization and translation management, they also have their drawbacks. The closed-source, cloud-based nature of Crowdin presents risks in terms of data privacy and customization, whereas Pootle's outdated technology and limited integrations make it less appealing for modern-day businesses. This thesis aims to address these gaps by developing an open-source, customizable, and up-to-date solution that can be run internally within companies, ensuring a secure and adaptable approach to localization and translation management.

## **2.2 Software development methods**

This section will explore best practices in software architecture and code writing that are pertinent to the creation of localization and translation management systems. A thorough understanding of these methodologies will provide a solid foundation for the development of the proposed open-source solution.

### **2.2.1 Hexagonal Architecture**

Hexagonal architecture, also known as the Ports and Adapters pattern, emphasizes the separation of an application's core logic from its external dependencies. This approach allows for easy interchangeability of components and promotes better testability, maintainability, and adaptability. Implementing hexagonal architecture in the localization and translation management system will ensure a flexible and robust design that can accommodate various business requirements. [3]

### **2.2.2 Clean Code**

Writing clean, readable, and well-documented code is essential for the long-term maintainability and success of any software project. Adopting clean coding principles ensures that the code is easy to understand, modify, and troubleshoot, which is crucial for the proposed system as it aims to be customizable and adaptable to the specific needs of businesses. [4]

### **2.2.3 Test-Driven Development (TDD)**

TDD is a software development approach that emphasizes writing tests before writing the actual code. This method ensures that code is correct, reliable, and efficient from the outset, minimizing the need for extensive debugging and refactoring later in the

development process. TDD could contribute to the robustness and reliability of the proposed system. [5]

In conclusion, adopting these best practices in software architecture and code writing will enhance the development process of the open-source localization and translation management system. These practices will ensure that the system is flexible, efficient, and reliable, catering to the diverse needs of businesses in a rapidly changing global landscape.

### **2.3 Open-source Apache-2.0 license**

The Apache License 2.0 (Apache-2.0) is a widely utilized open-source software license, renowned for its flexibility, permissiveness, and extensive adoption across various sectors.

Several crucial attributes contribute to the Apache-2.0 license in the context of this project. One such attribute is the patent grant, which protects developers and users by granting a license to employ, modify, and distribute any patented innovations integrated into the software. The license also provides copyright protection, ensuring that the original creator retains ownership of their work. Simultaneously, sublicensing provisions facilitate software distribution under different licenses, fostering adaptability and compatibility with diverse licensing models.

In the context of an open-source thesis software project, the Apache-2.0 license offers numerous significant benefits. Its permissive nature fosters collaboration, permitting the redistribution and modification of software with minimal restrictions. Additionally, the license embraces contributions from various sources, such as corporations and individual developers, expanding the scope of knowledge and expertise that can be employed to advance the project.

The Apache-2.0 license is particularly suitable for use by multiple organizations due to its non-reciprocal nature. In contrast to some other open-source licenses, the Apache-2.0 license does not mandate that derivative works be distributed under an identical license, allowing companies to incorporate open-source software into their proprietary products without disclosing their source code. This characteristic makes the Apache-2.0 license an attractive option for projects aiming to collaborate with a diverse array of organizations,



as it provides a flexible and accommodating legal framework that can readily adapt to various commercial environments.

In conclusion, the Apache License 2.0 (Apache-2.0) emerges as a highly advantageous choice for open-source software projects, particularly within the context of this thesis. Its permissive and adaptable nature encourages collaboration, innovation, and the dissemination of knowledge, as it accommodates contributions from a diverse range of sources. The patent grant and copyright protection features further strengthen the license by safeguarding both the creators and the users of the software. The non-reciprocal nature of the Apache-2.0 license, which permits integration into proprietary products without necessitating source code disclosure, renders it exceptionally suitable for partnerships with multiple organizations. Overall, the Apache-2.0 license fosters an inclusive and dynamic ecosystem that aligns with the goals of this thesis project, promoting advancement and the exchange of ideas across various sectors. [6] [7]

## **3 Methodology**

To address the research questions and achieve the research goals, the methodology will consist of a combination of theoretical investigations, requirement gathering, system development, and testing and validation methods. The following sections will describe each of these components in more detail.

### **3.1 Comparative analysis**

A comparative analysis of existing localization and translation management solutions, specifically focusing on Crowdin and Pootle, will be performed to understand their strengths, weaknesses, and the gaps they leave in the market. This analysis will inform the design and development of the proposed open-source, internally-run system.

### **3.2 Requirement gathering and prioritization**

Requirements gathering will be conducted with potential users, including project managers, translators, and developers. Based on the feedback collection process that

included meetings with managers and translators, the key requirements of the proposed system will be prioritized. Discussion with developers will also be conducted to better understand how this system can be integrated into real projects. This information will be used to ensure that the system caters to the diverse needs of businesses while remaining versatile, efficient, and effective.

### **3.3 Development planning**

Through the development process various artifacts will be created, such as diagrams, user stories, and business cases, to plan the work and outline the system's structure, components, and features. These artifacts will provide a clear roadmap for the development process, ensuring that the final product meets the needs of businesses. GitHub was selected as the code repository for managing the project's codebase, further emphasizing the importance of adaptability and collaboration in the project's design.

### **3.4 System testing**

Various tests will be written for each functional component of the system to verify that the individual components work as expected. This approach will help ensure the robustness and reliability of the proposed system.

### **3.5 User acceptance testing**

User acceptance testing will be performed with a selection of potential users to validate that the system meets their needs and expectations. This feedback will be used to make any necessary adjustments to the system to ensure that it is user-friendly and effective in addressing the localization and translation management challenges faced by businesses.

In conclusion, the revised methodology outlined in this section will help guide the development of an open-source, customizable, and adaptable localization and translation management system that addresses the limitations of existing solutions such as Crowdin and Pootle. By combining theoretical investigations, requirement gathering, development planning, and testing and validation methods, this research will contribute to the understanding of the current challenges faced by businesses and provide a platform that meets modern expectations while ensuring a seamless transition for companies.

## **4 Software development**

The construction of the localization and translation management system necessitates an all-encompassing strategy, tackling multiple components such as server-side and client-side framework and design, feature creation, and testing and verification. This chapter explores the various stages of the software development process, concentrating on the choice of technologies, design patterns, and methodologies employed to establish a versatile, modular, and scalable system.

To attain these objectives, the software development process was segmented into several discrete stages: server-side framework and design, client-side framework and design, feature creation, and testing and verification. The subsequent sections will present a comprehensive description of the methodologies, instruments, and technologies utilized in each stage, underlining the significance of flexibility, cooperation, and a user-oriented approach to guarantee the system's effectiveness in addressing the challenges encountered by enterprises in the localization and translation management sphere.

### **4.1 Server-side Architecture and Design**

The server-side architecture and design for the localization and translation management system were influenced by the comparative analysis of existing solutions, requirement gathering, and prioritization. The Hexagonal architecture, also known as the Ports and Adapters pattern, was selected for its ability to separate concerns between the core application and its surrounding infrastructure. This separation enables the development of a flexible, modular, and scalable system. The technology stack employed for the server-side included Java 17 as the programming language and Spring Boot with Gradle for backend development. The server-side was divided into two main components: the application module, which houses the core business logic, and the git-service module, which serves as a plugin for automating integration with git repositories. The final server-side architecture and design is shown in Figure 1.

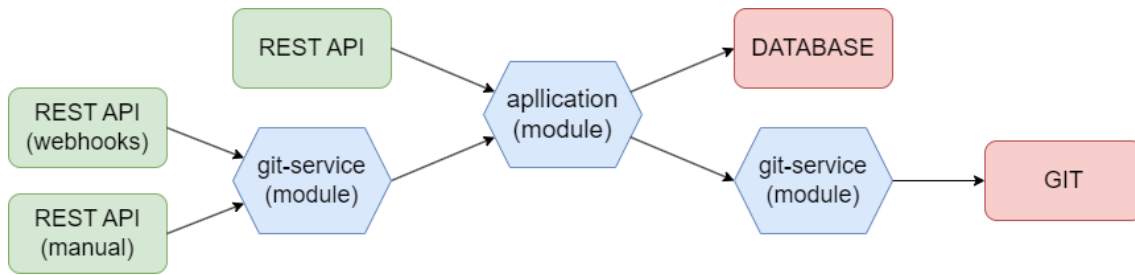


Figure 1. Server-side architecture.

#### 4.1.1 Application module

The application module acts as the central hub for essential business logic within the system, encompassing user management, project management, working group management, and label management.

User management involves creating, updating, and removing user accounts. This functionality ensures efficient administration of user accounts, facilitating smooth user onboarding and related operations.

Project management includes processes like project creation and configuration. During project creation, users provide essential details such as the associated Git repository, branch name, and necessary files. These details enable seamless integration with version control systems and ensure accurate project synchronization.

Working group management focuses on creating and organizing worker groups assigned to specific projects. Workers are assigned specific roles, such as moderator, translator, or both, with their language proficiency specified for accurate label translations. Project owners or moderators have the authority to add new workers, promoting collaboration and task delegation. Existing users can be assigned as workers or new users can be created as needed.

Label management encompasses various aspects, including translation, review, and approval. Users specify the target language and provide contextual information for accurate translation. During the review process, translators and moderators can leave notes, request feedback, or suggest improvements. Moderators play a crucial role in approving labels to ensure quality and adherence to criteria.

By integrating these functionalities within the application module, the system offers a comprehensive and streamlined approach to core business cases. It facilitates efficient

user management, project administration, working group collaboration, and label handling. This centralized structure promotes effective communication, simplifies complex processes, and enhances the overall localization and translation management experience for businesses.

#### **4.1.2 Git service module**

The git service module acts as plugin that is responsible for automated integration with git through executing the bash commands.

Git service has the endpoint that is been used by git repository webhooks. When git repository receives a new change then it automatically sends a POST request to the git service endpoint to notify what files have been changed. If it is found that the label file has been changed then the git service runs a next stack of commands to fetch only the new version of label files, after that it reads of lines and sends the data to the application module to process the labels and if needed create new language variations for the new labels.

The git service module also has the second endpoint that is needed for triggering the push of the labels to the project git repository. Git service module gets all the latest labels from the application module and then adds them all to the labels files. When all the label files are prepared then git service creates a new commit for the pre-defined branch and sends the push to the git.

#### **4.1.3 Database Architecture and Design**

In order to support the server-side architecture and design of the localization and translation management system, a comprehensive database was required. The process of designing the database began with a thorough understanding of the needs and entities that the server-side components would encompass. To visualize and create the database structure, Vertabelo, a powerful data modeling tool, was used to develop an Entity-Relationship Diagram (ERD) scheme. This ERD scheme ensured that all requirements and relationships between entities were well-represented and organized to facilitate efficient server-side operations. Furthermore, PostgreSQL was chosen as the database management system for the project, due to its robustness, scalability, and open-source nature, which aligns well with the goals of the localization and translation management system. [8] The final database ERD scheme is shown in Figure 2.

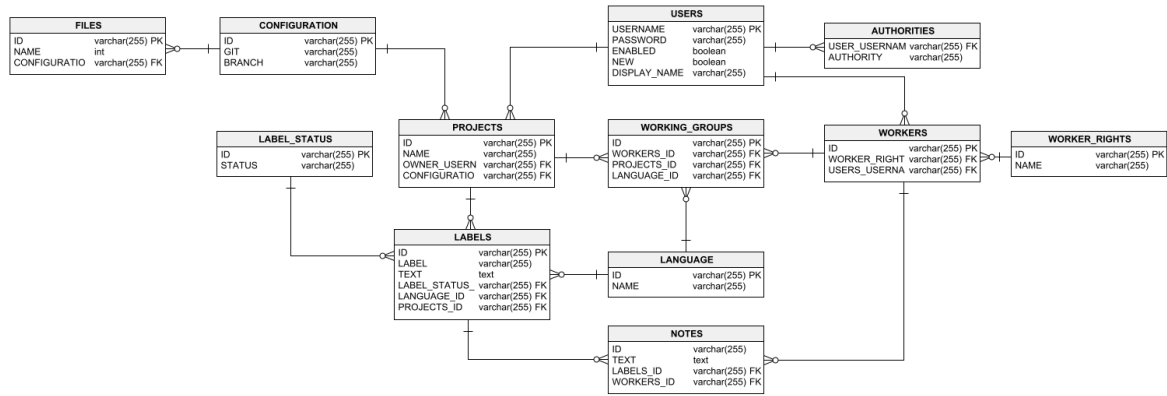


Figure 2. Database ERD scheme.

## 4.2 Client-side Architecture and Design

For the client-side architecture and design, React with TypeScript was chosen as the technology stack, and a lightweight prototype was developed to demonstrate the functionality of the server-side components. This approach ensures adaptability and future-proofing of the system, allowing for the independent development and replacement of components as needed. [9]

## 4.3 Features Development

Developing a comprehensive set of features for the localization and translation management system is crucial to address the diverse needs of businesses and bridge the gaps left by existing solutions like Crowdin and Pootle. The features development process involved user stories creation, ensuring that the system caters to the needs of project managers, translators, and developers. The user stories are shown in Figure 3.

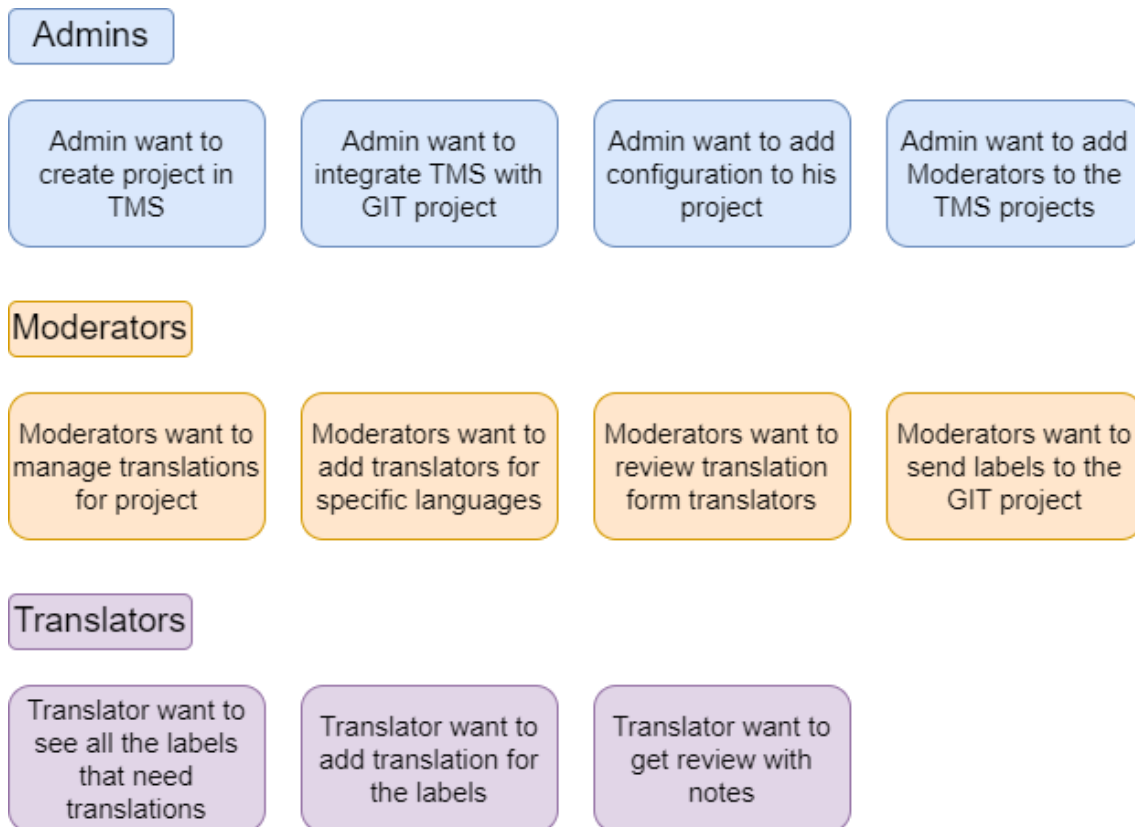


Figure 3. User stories

For these user stories were created happy flow for server and client sides. Server-side is responsible for acquiring webhook request and checking if it contains label file changes or updates if yes then server-side should pull such files from git and process them. If files contain new labels, then for these labels will be created translation variants for all added languages and after that client side would get new labels that need to be translated. The final server-side happy flow is shown in Figure 4.

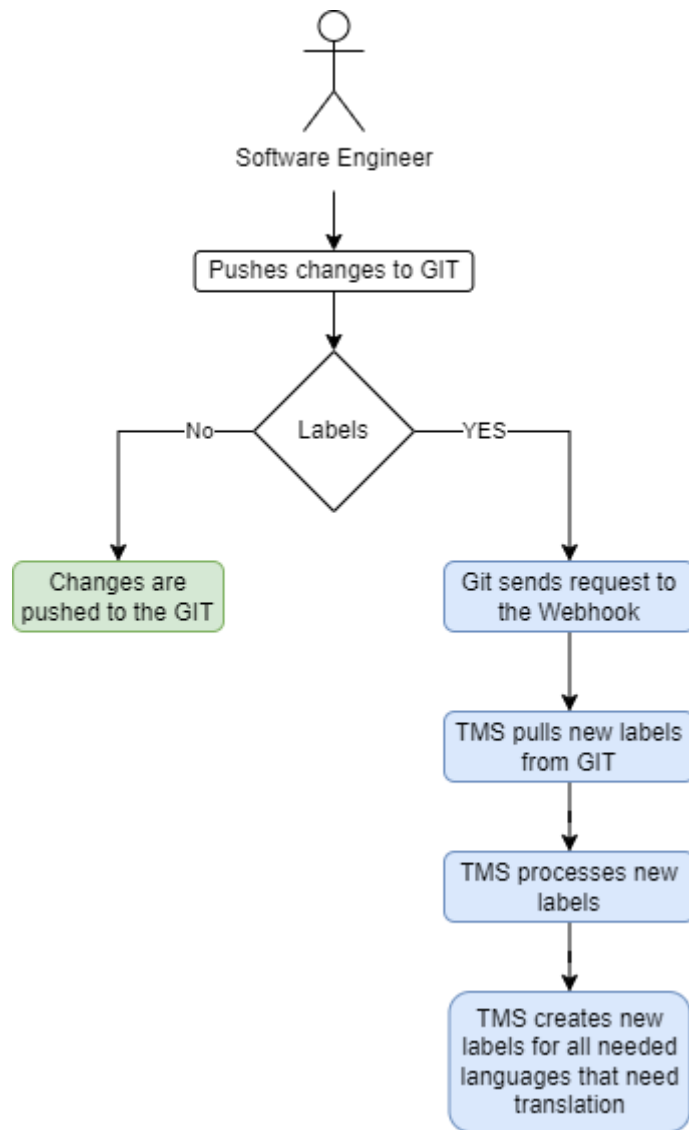


Figure 4. Server-side happy flow

Client-side is responsible for letting moderators to create new users for translators for needed projects. After that translators will be responsible for translating labels. After labels are translated, they will be sent for moderator review. If they pass the review, they will send back to the server where they will be pushed to the project git. The final client-side happy flow is shown in Figure 5.



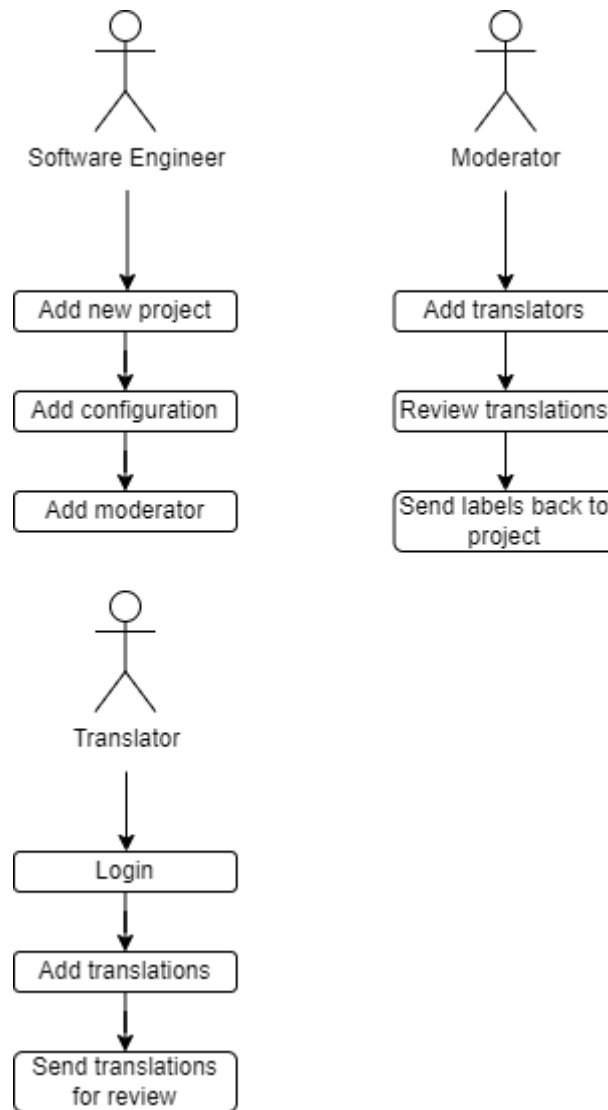


Figure 5. Client-side happy flow.

#### 4.3.1 Server-side Features Development

The server-side features development for the localization and translation management system aimed to create a robust and efficient solution, addressing the key requirements identified during the requirement gathering stage. Utilizing IntelliJ IDEA, Azul Java JDK, and Docker, the server-side components focused on implementing the application and git-service modules.

The application module was responsible for handling the core business logic, and it was further divided into business and endpoint sub-modules. This division allowed for a clear separation of concerns between the domain entities, DAO classes, and Repository classes, as well as the SpringBoot MVC controllers for REST API endpoints. By adopting

this modular structure, the server-side components efficiently catered to the various aspects of the localization and translation management process.

The git-service module, on the other hand, aimed to automate git processes using CLI commands and streamline the management of label data without requiring manual intervention from users. This module also contained business and endpoint sub-modules that servers as endpoint for webhooks that git services are using to notify about updates or changes in labels files or directories, like the application module, ensuring consistency and modularity in the server-side components' design. [10]

The final structure of the server-side for the project is displayed in Table 1.

Table 1. Server-side directory structure

Directory	Purpose
application/business/*/dao/contract	Interfaces for DAO classes.
application/business/*/dao	DAO classes.
application/business/*/entity	All application entities.
application/business/*/repo/contract	Interfaces for Repository classes.
application/business/*/repo	Repository classes.
application/business/*/usecase/contract	Interfaces for usecase classes.
application/business/*/usecase	Usecase classes.
application/entrypoint	Spring MVC controllers.
git-service/business/*/entity	All git-service entities.
git-service/business/*/usecase/contract	Interfaces for usecase classes.
git-service/business/*/usecase	Usecase classes.
git-service/business/*/utils	Utils classes for file processing.
git-service/entrypoint	Spring MVC controllers.
src/*/application	Spring configuration files.

### 4.3.2 Client-side Features Development

The client-side features development focused on providing a user-friendly and accessible interface to interact with the server-side components of the localization and translation management system. Using Webstorm, React, and Node, the development process involved the creation of a simple, lightweight prototype to demonstrate the essential functionalities of the system.

The client-side prototype emphasized user authentication, project creation and configuration, worker management, and label translations. These features were designed to address the needs of project managers, translators, and developers, ensuring that the system caters to the diverse requirements of businesses involved in localization and translation management.

The final structure of the client-side for the project is displayed in Table 2.

Table 2. Client-side directory structure.

Directory	Purpose
src/components	React components.
src/domain	React entities.
src/service	React services.
src/**	App classes to start project.

### 4.3.3 Security

In addition to the structural and functional aspects of the localization and translation management system, guaranteeing a secure environment is of paramount importance. To accomplish this, the system incorporated Spring Security, along with JSON Web Tokens (JWT) for authentication and authorization objectives.

JWT generation and acquiring is shown in Figure 6.

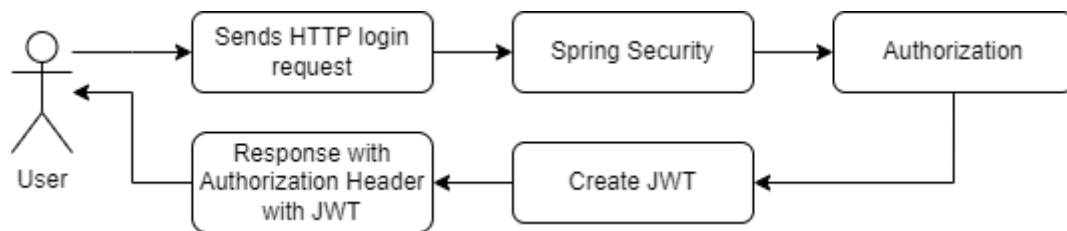


Figure 6. JWT generating and acquiring.

Spring Security was integrated into the server-side elements via filter chains utilized throughout the request receiving process, specifically within the application module, to safeguard all REST API endpoints and server-side operations. The framework enabled the establishment of a robust, stateless authentication method, wherein each user's identity

and access privileges are confirmed through JWT. Consequently, the server-side elements remain secure and impervious to unauthorized access. [11]

JWT functions as a compact and self-sufficient means of securely conveying information between parties in the form of a JSON object. Within the localization and translation management system's context, JWT operates as a bearer token, offering a secure method for authenticating and authorizing users. Upon successful login, the server-side elements generate a JWT, which is subsequently transmitted to the client-side. The client-side, developed using React and TypeScript, stores the JWT and incorporates it as an authorization header in subsequent API requests. This technique ensures that solely authenticated and authorized users can access the system's resources and execute actions in accordance with their assigned permissions.

By implementing Spring Security in conjunction with JWT, the localization and translation management system effectively addresses security concerns, protecting organizational data and system integrity. This security implementation augments the modular and scalable structure of the system, contributing to a comprehensive solution that caters to the varied requirements of businesses in the localization and translation management domain.

#### **4.4 Testing and validation**

Testing and validation of the system were conducted in accordance with the methodology outlined earlier. A combination of unit testing, integration testing, and system testing techniques were employed to ensure the robustness and reliability of the system's functional components. Test cases and scenarios were carefully designed to cover all critical aspects of the system, and any issues or bugs discovered during the testing process were promptly addressed and resolved.

In conclusion, the software development process for the localization and translation management system followed a structured approach, adhering to the methodology and focusing on the key components of architecture, feature development, and testing and validation. The resulting system is a flexible, modular, and scalable solution that addresses the limitations of existing solutions such as Crowdin's closed-source and cloud-based approach and Pootle's old and inconvenient technology stack, meeting the diverse

needs of businesses and ensuring a seamless transition for companies in localization and translation management.

## **5 Result and analysis**

The outcomes illustrate the system's effectiveness in addressing the limitations of existing solutions and meeting the diverse needs of businesses engaged in localization and translation management.

### **5.1 Final look of the system**

The research findings show the efficacy of the developed system in addressing the limitations of existing solutions and meeting the diverse requirements of businesses involved in localization and translation management. The final appearance of the system showcases a user-friendly and intuitive web interface on the front-end.

#### **5.1.1 Login Form**

One of the most important components of the system is the login form page, which acts as the initial access point for unauthorized users. With its simple design, users are required to input their username and password for authentication. This page ensures secure and controlled access to the system, granting entry only to users who are authorized. Please refer to Figure 7 for the presentation of the login form page.

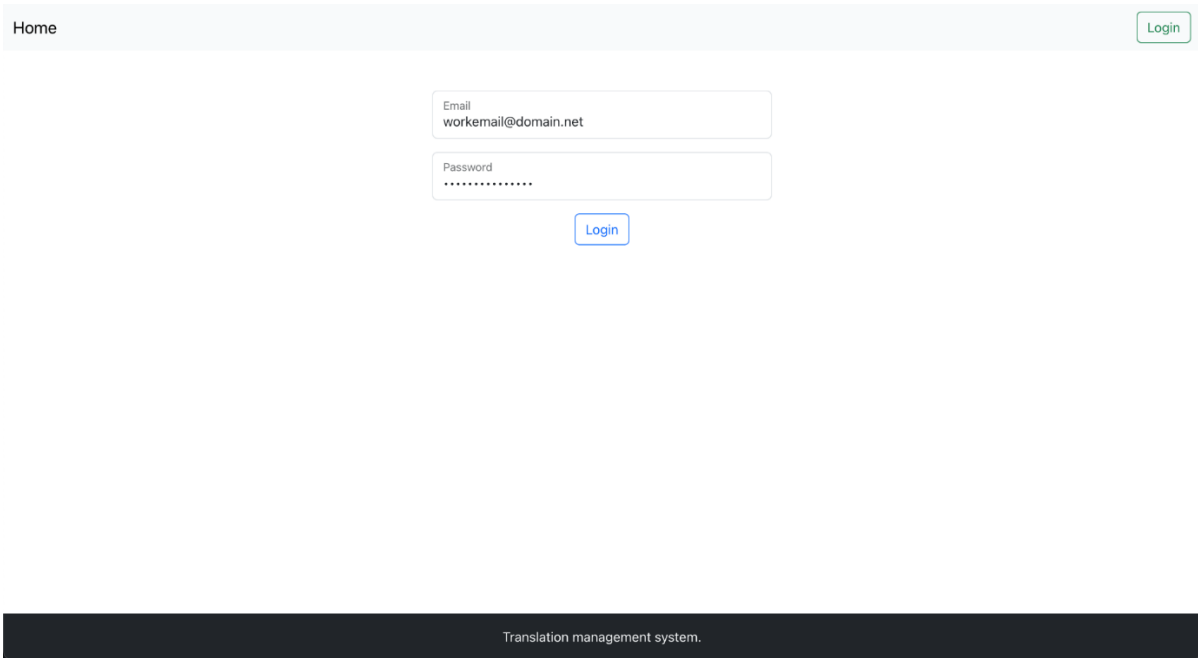


Figure 7. The login form page.

### 5.1.2 Home Page

The home page serves as a centralized hub for projects within the system. It presents users with a comprehensive list of projects they have access to, displaying crucial information such as project name, owner, and associated git repository. This page offers a clear overview of available projects and facilitates efficient project navigation. The design of the home page can be found in Figure 8.

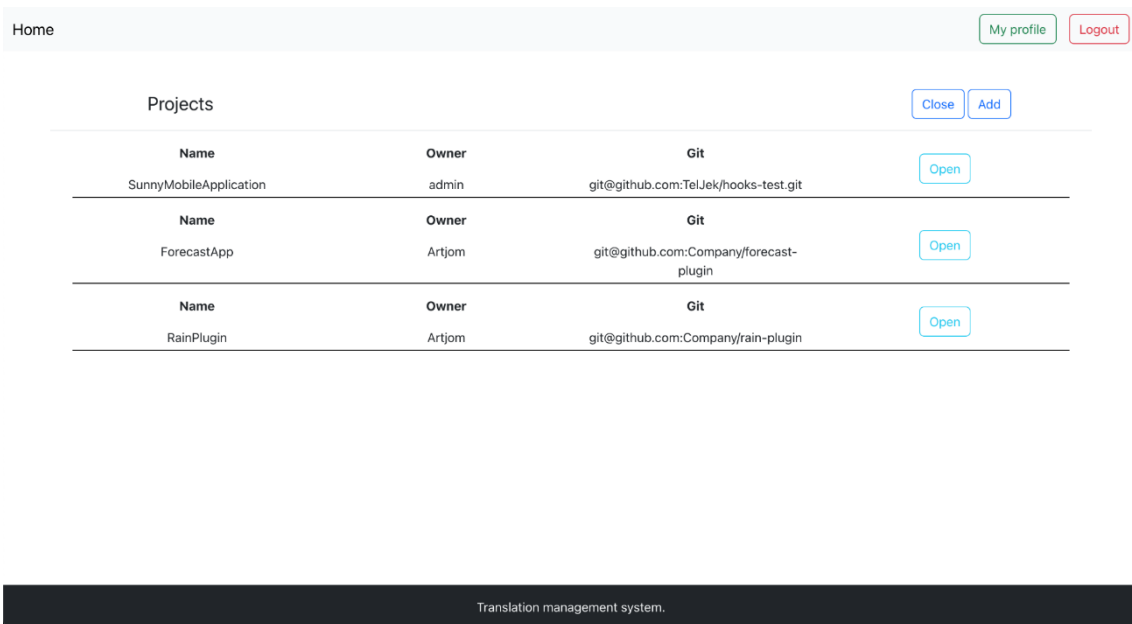


Figure 8. The home page.

### 5.1.3 New Project Creation Page

For administrators, the new project creation page provides a dedicated space to add projects and configure their settings. With its user-friendly design, the page prompts users to input essential details such as project name, git repository information, branch name, and file names that will serve as the primary label files for the project. This page simplifies the process of creating new projects and ensures accurate project configuration. Screenshots of the new project creation page are available in Figure 9, Figure 10, and Figure 11.

In Figure 9 is shown the starting point of the project creation, with entering the project name.

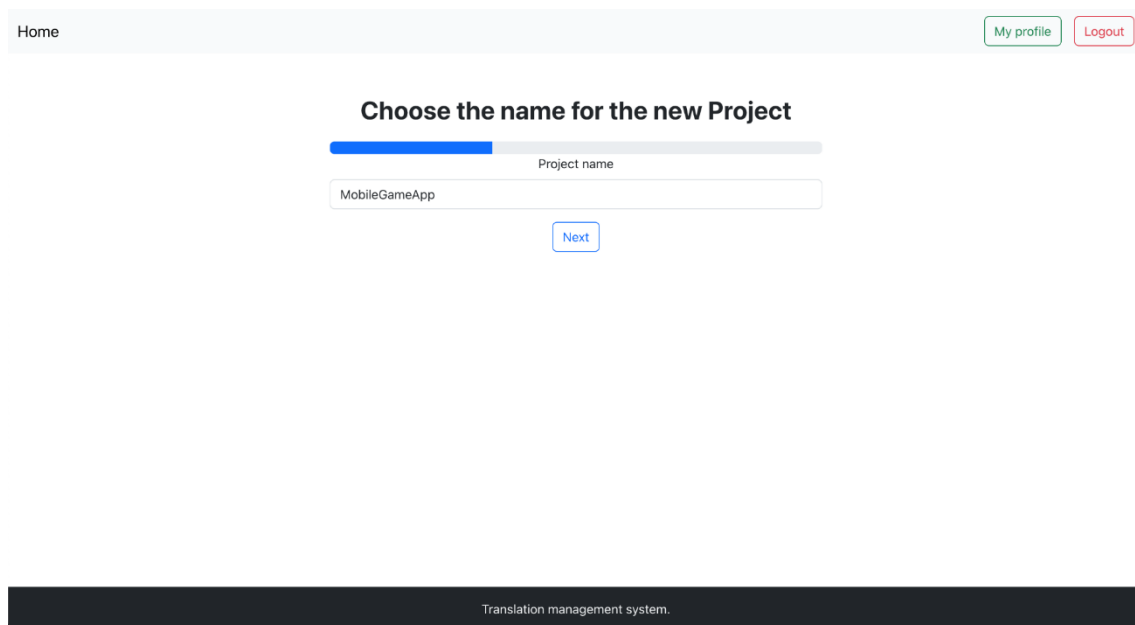


Figure 9. New project creation name page.

In Figure 10 is shown the second step of the project creation process. In the second step user should enter the configuration properties for the project. The configuration properties are the git repository URL, the git branch for the translations and the file name where all the labels are stored.

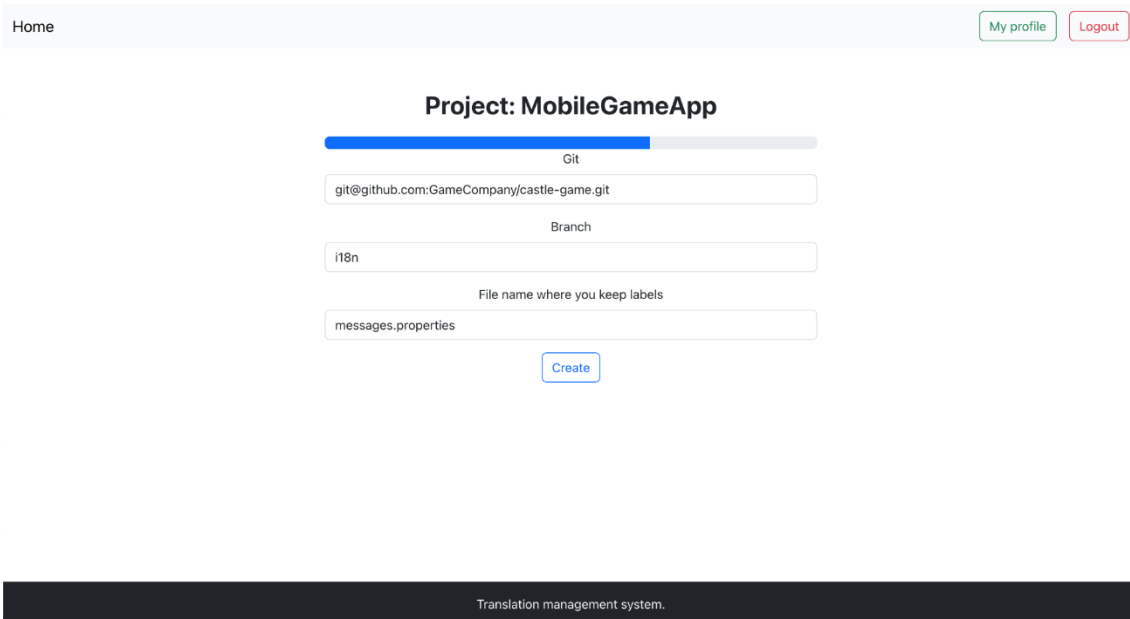


Figure 10. New project creation configuration page.

In Figure 11 is shown the final step of the project creation where user can see that all went successfully, and the project is created.

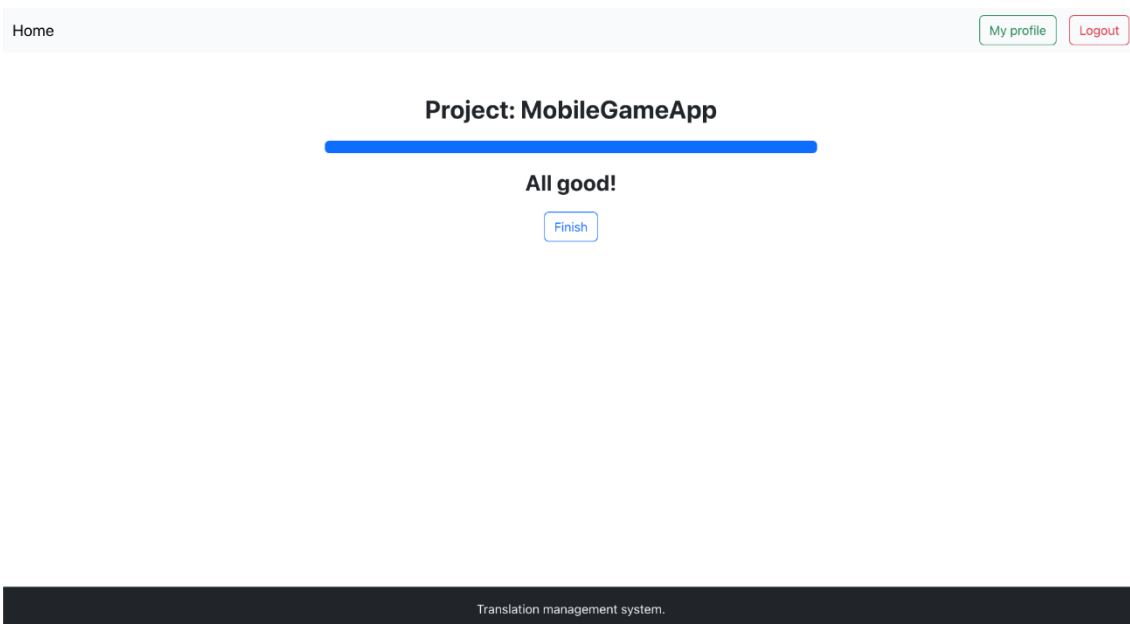


Figure 11. New project creation success page.

#### 5.1.4 Project Page with Settings, Working Groups and Labels

The project page serves as the main page for the system. It features such options as project settings, where project owners can update some configuration data for the project. The working groups management, where users can add more moderators or translators for the project. The labels list for the project where translators will see all the labels that are



assigned to specific language. The project page is shown in Figure 12, Figure 13, Figure 14 and Figure 15.

In Figure 12 is shown the overall view of the project page.



Figure 12. The project page.

In Figure 13 is shown the project settings where user can update the git URL, the branch name. Also, here the moderators of the project can push all the complete labels or fetch labels manually.

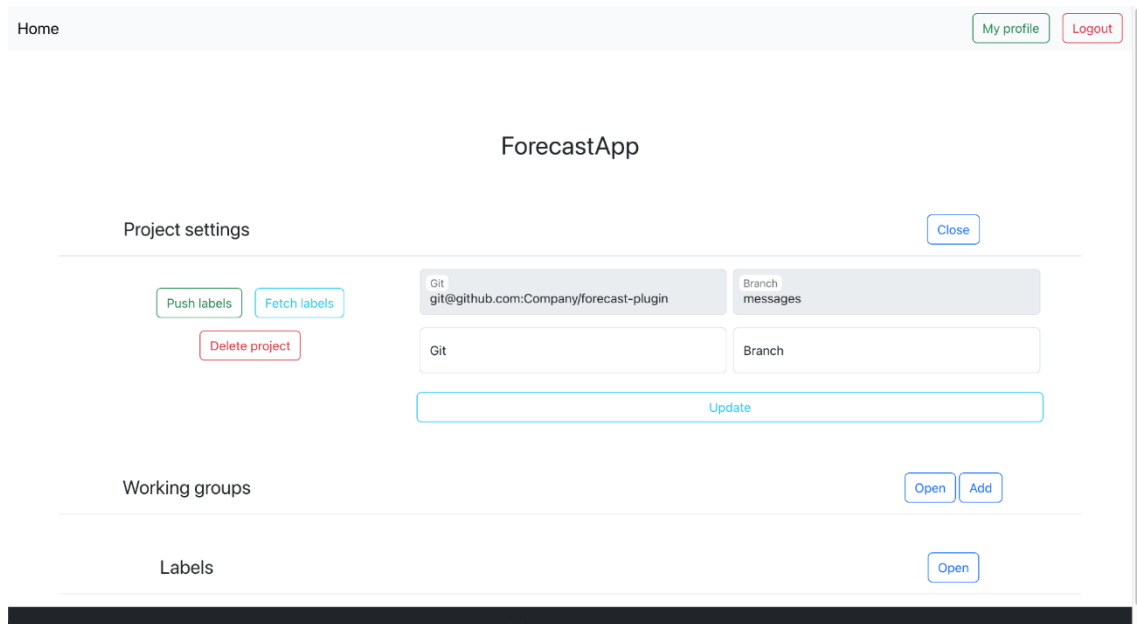


Figure 13. The project settings.

In Figure 14 is shown the working groups of the project. Moderators also can add new working groups or delete the existing ones.

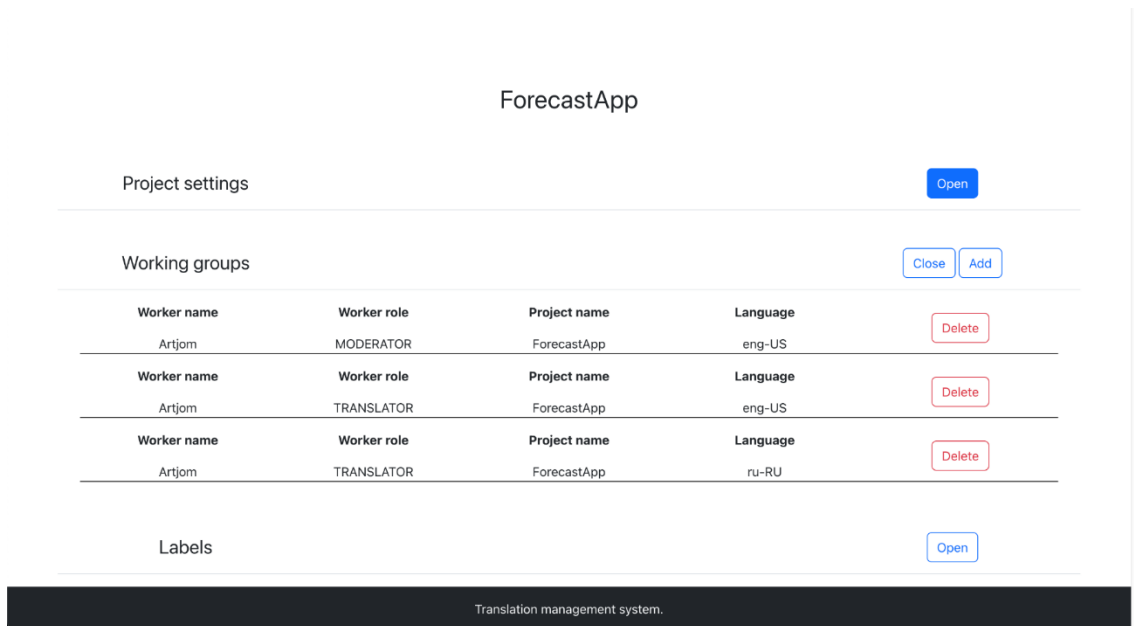


Figure 14. The project working groups.

In Figure 15 is shown the labels of the project. Here translators can open the label and proceed with the label translation.

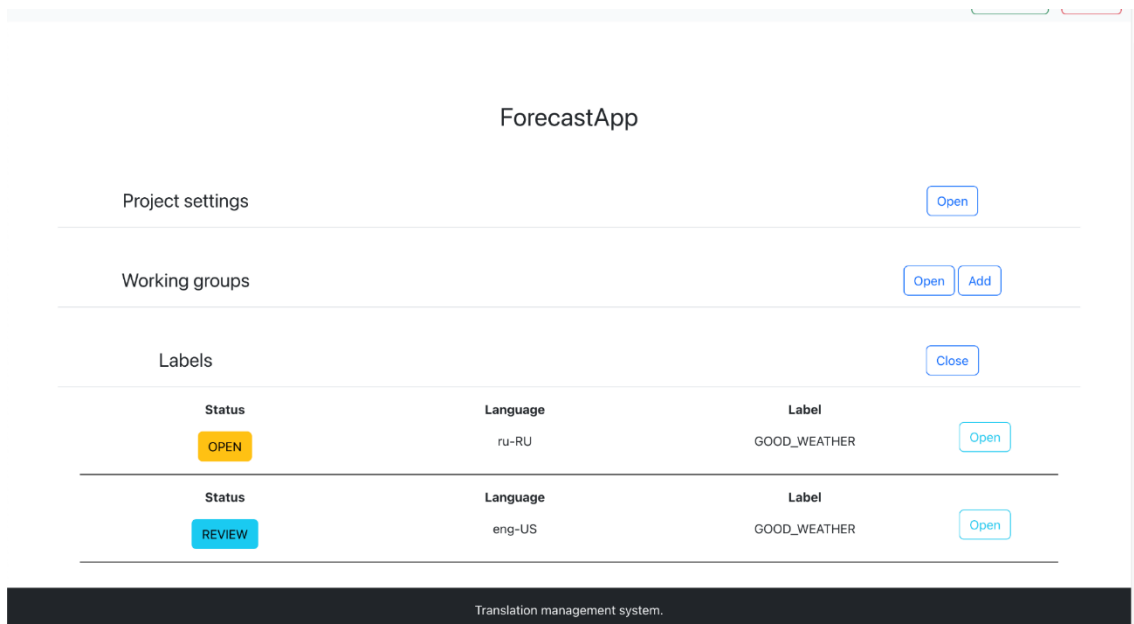


Figure 15. The project labels.

### 5.1.5 Label Management Page

The label management page serves as the main place for the label's translation creation. Here translators can add translations for the label using the provided context and during this process translators can add notes for moderators, or the moderators can add notes with what to improve. Also, moderators should approve the translations before they are eligible for pushing. The label management page is shown in Figure 16 and Figure 17.

In Figure 16 is shown the label management page. Users can see the label original name and the label context that is provided by the developers who created the label. After providing the translation for the requested language, the user will confirm it and wait the feedback or approval by the moderators.

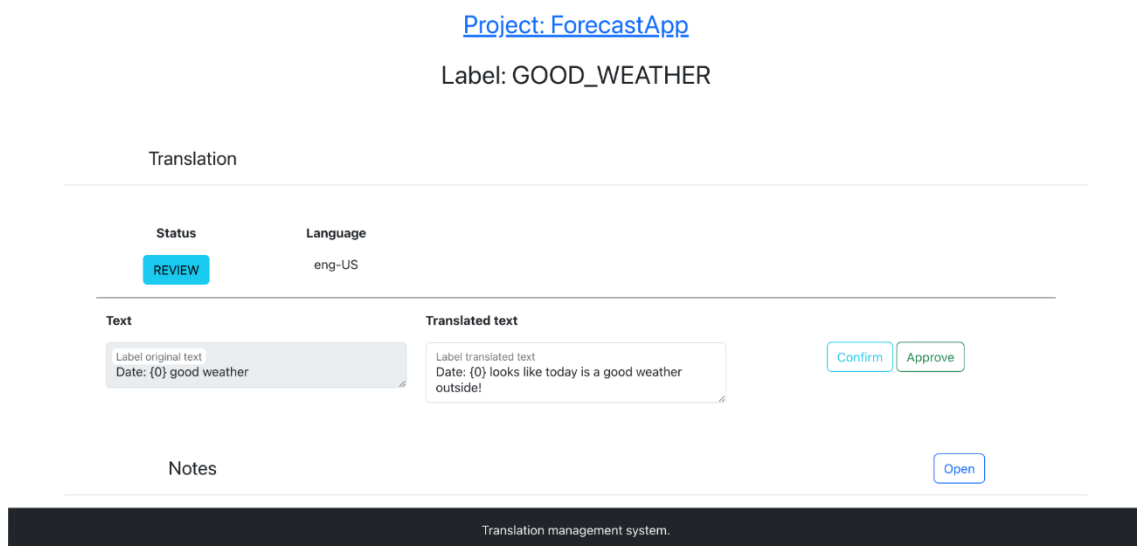


Figure 16. The label management page.

In Figure 17 are shown the label notes. Users can leave notes that will request any help or moderators while reviewing can request some changes through the notes.

Label: GOOD\_WEATHER

Translation

---

Status	Language
<b>REVIEW</b>	eng-US

---

Text	Translated text	
Label original text Date: {0} good weather	Label translated text Date: {0} looks like today is a good weather outside!	<b>Confirm</b> <b>Approve</b>

---

Notes **Close**

User	Note	
Artjom	I need feedback.	
	Add note Add new note!	<b>Send</b>

---

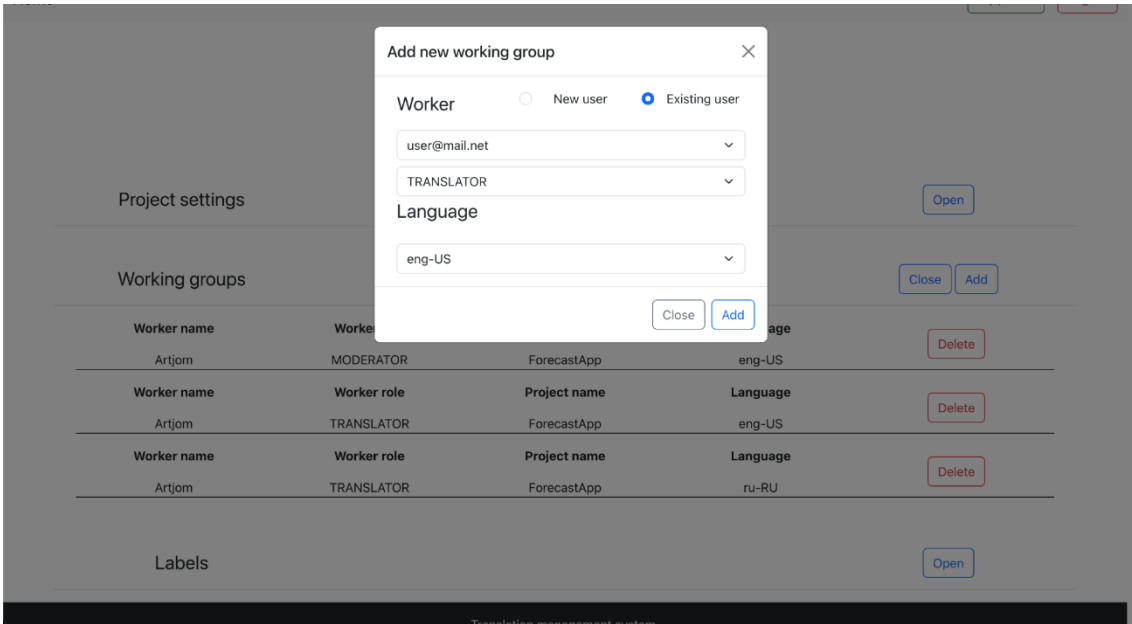
Translation management system.

Figure 17. The label notes.

### 5.1.6 New Working Group Creation with User Creation Page

The working groups page displays all the users working on the project. Moderators have the capability to create new users or assign existing users to the project. This page facilitates efficient management of project teams and ensures effective collaboration among team members. Screenshots of the working groups and user management page can be found in Figure 18 and Figure 19.

In Figure 18 is shown the new working group creation process. Moderator should select the existing user, the rights for the user and the language that will be assigned to the user.



In Figure 19 is shown the new working group creation process with a completely new user. Moderator should enter the username or email of the new user and the temporary password after that the rights for the user and the language that will be assigned to the user.

Figure 18. New working group creation with existing user.

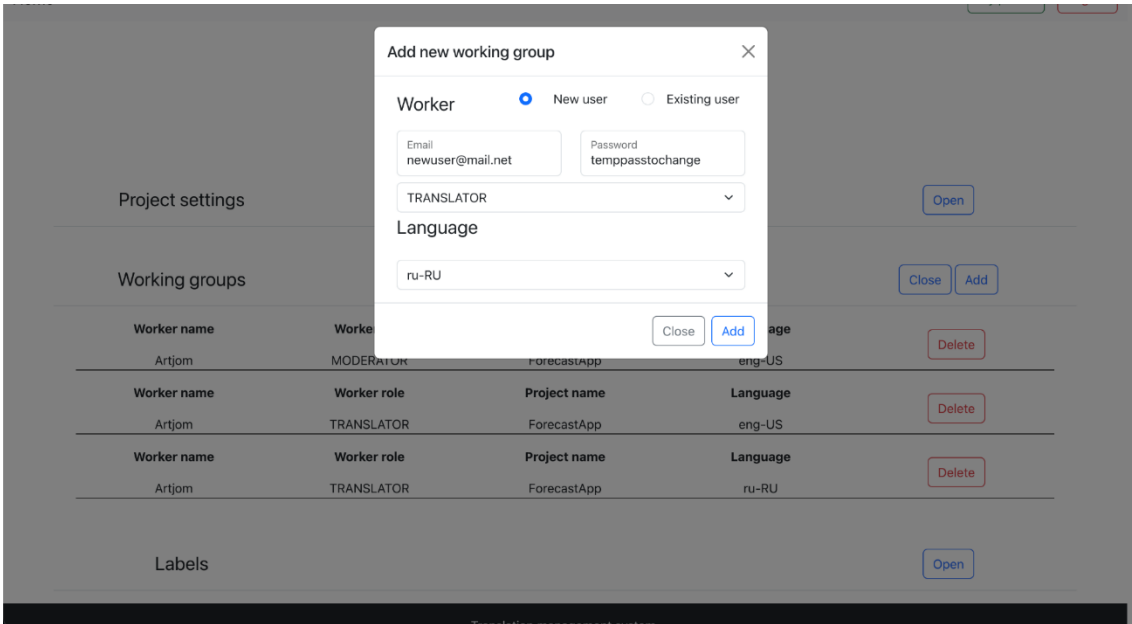


Figure 19. New working group creation with new user.

## 5.2 System Evaluation and Impact

The research undertaken in this thesis has successfully led to the development of a secure, customizable, and adaptable open-source localization and translation management system. This new system addresses the limitations of existing solutions, such as the data privacy risks associated with closed-source, cloud-based platforms like Crowdin, and the outdated technology and limited integrations offered by open-source alternatives like Pootle. The developed system's design, which is based on the Hexagonal architecture, allows for a more flexible and modular solution that can be easily maintained and adapted to accommodate various business requirements.

Translation Management System final architecture is shown in Figure 20.

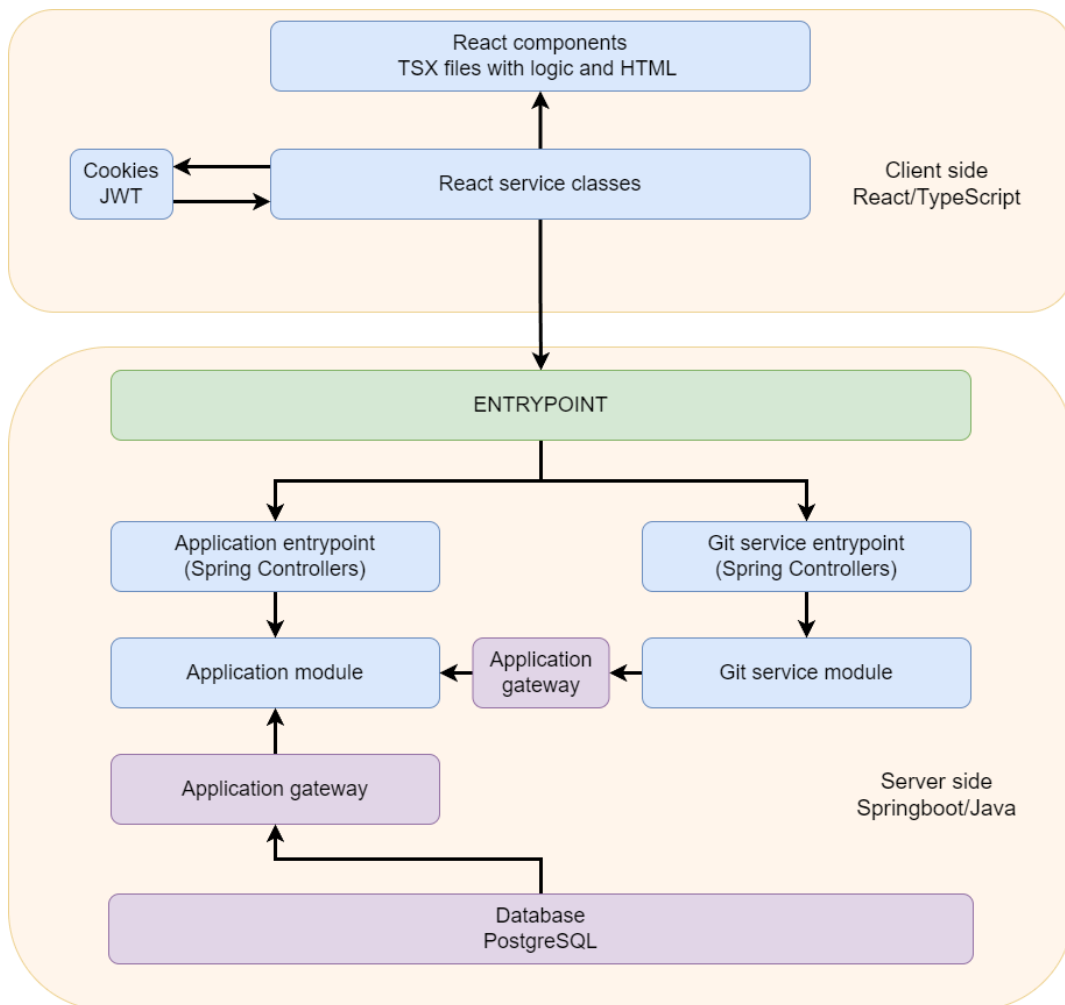


Figure 20. Translation Management System final architecture

The requirement gathering process played a crucial role in identifying and prioritizing essential features and functionalities required by businesses in the localization and

translation management domain. A primary source of this information was Swedbank, specifically its Mobile Team, which currently uses Crowdin for its translation management needs but is seeking a better open-source solution. Their valuable insights and practical experience contributed significantly to understanding the challenges and expectations of companies engaged in localization and translation management. Consequently, this comprehension led to the development of a versatile, efficient, and effective system tailored to the varied requirements of such businesses, offering a competitive alternative to existing solutions.

Throughout the software development process, the system was designed and constructed with a focus on adaptability, scalability, and security. The server-side architecture employed contemporary technologies, including Java 17 and Spring Boot, while the client-side utilized React with TypeScript, delivering a future-proof solution. Moreover, the adoption of PostgreSQL as the database management system aligns with the project's open-source essence.

User acceptance testing validated that the system met the needs and expectations of potential users, including the Mobile Team at Swedbank, who actively participated in the requirement gathering process. This testing ensured that the developed platform effectively addresses the localization and translation management challenges faced by businesses and provides a viable open-source alternative to solutions like Crowdin. Additionally, the system's open-source nature allows companies to run it internally, offering more control, adaptability, and security compared to existing solutions, catering to organizations that prefer an in-house solution for their translation management needs.

## **6 Conclusions and future work**

This research work focused on the problem of localization and translation management system which is becoming more important for businesses to extend their outreach and attract more customers and consumers. The research explored the available solutions and justified the need for an open-source localization and translation management system development.

This thesis contributes by presenting a solution that is secure, customizable, and adaptable for various types of business that need such a system. It eliminates the need for using cloud-based, closed-source solutions that can present data privacy risks and other challenges. By reflecting on the work done throughout the thesis, the developed system offers a valuable alternative for companies seeking a more flexible, open, and safe approach to localization and translation management.

Using the newly developed system it is possible to integrate the TMS into git repository and that will add the seamless automated processes for grabbing the labels from git and pushing them back to the git. TMS supports adding new users for moderators and translators, new projects with specific configuration for different git repositories with different branch names and file names. It also supports adding new languages for which will be automatically created new label translation variants that will request adding new translations. Moderators can add new users as workers for projects, for example translators for specific languages, and after the translation is finished moderators will review them and request needed changes or approve the labels and then push them back to the git.

While the system developed in this thesis represents a significant advancement in localization and translation management solutions, there is scope for further enhancement. With plans for ongoing development, several key improvements are expected.

Firstly, the client-side element, particularly the code, warrants attention. The current client was developed as a prototype during the thesis process and, as such, there is ample room for improvement. Large effort will be devoted to refining and streamlining the codebase to enhance its resilience and readability, because during the development process the client-side code was written poorly due to authors lack of experience with front-end development. This refinement is expected to facilitate future development activities, including the addition of new features. In future iterations of the system, it is proposed to integrate functionality such as filtering and searching capabilities within projects, working groups, and labels. This addition will significantly improve user experience and system efficiency, making it an even more powerful tool for businesses engaged in localization and translation management.



## References

- [1] "Crowdin," 16 04 2023. [Online]. Available: <https://crowdin.com/>.
- [2] "Pootle," 16 04 2023. [Online]. Available: <https://pootle.translatehouse.org/>.
- [3] J. M. G. d. Paz, "Ports and Adapters Pattern," 29 08 2018. [Online]. Available: <https://jmgarridopaz.github.io/content/hexagonalarchitecture.html>. [Accessed 16 04 2023].
- [4] Ionos, "Clean code: what makes programming code clean," 12 02 2022. [Online]. Available: <https://www.ionos.com/digitalguide/websites/web-development/clean-code-principles-advantages-and-examples/>. [Accessed 17 04 2023].
- [5] T. Hamilton, "What is Test Driven Development (TDD)? Example," 08 04 2023. [Online]. Available: <https://www.guru99.com/test-driven-development.html>. [Accessed 20 04 2023].
- [6] "APACHE LICENSE, VERSION 2.0," 17 04 2023. [Online]. Available: <https://www.apache.org/licenses/LICENSE-2.0>.
- [7] F. E. Team, "Open Source Licenses 101: Apache License 2.0," 05 02 2021. [Online]. Available: <https://fossa.com/blog/open-source-licenses-101-apache-license-2-0/>. [Accessed 21 04 2023].
- [8] "Vertabelo," 17 04 2023. [Online]. Available: <https://vertabelo.com/>.
- [9] S. Modan, "The benefits of ReactJS and reasons to choose it for your project," 09 01 2023. [Online]. Available: <https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html>. [Accessed 17 04 2023].
- [10] J. N, "Git Automation Using Shell Scripts," 12 05 2021. [Online]. Available: <https://jishnun789.medium.com/git-automation-using-shell-scripts-e420767df36f>. [Accessed 12 04 2023].
- [11] Reflectoring, "Getting started with Spring Security and Spring Boot," 28 02 2023. [Online]. Available: <https://reflectoring.io/spring-security/>. [Accessed 16 04 2023].

## **Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis<sup>1</sup>**

I Artjom Ljuboženko

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Open-source System for Localization and Translation Management”, supervised by Mohammad Tariq Meeran
  - 1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
  - 1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

14.05.2023

---

<sup>1</sup> The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.