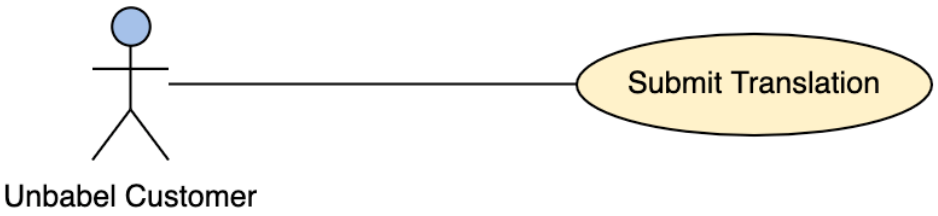
Use Cases



Submit Translation

Description

A Unbabel Customer will fill a form with the text he desires to translate, when submitted the text will be translated using Unbabel Service

Principal Actor

Unbabel Customer

Preconditions

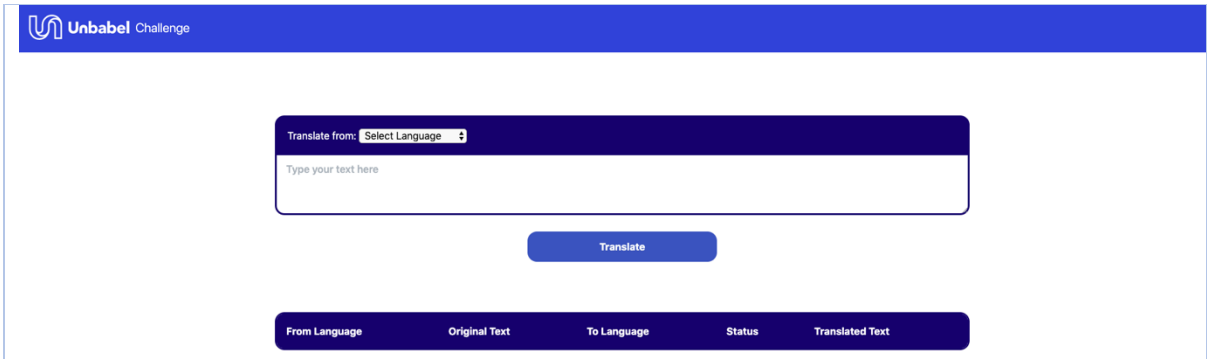
1 - The customer has to be registered on Unbabel Service
2 - The Customer has to provide mandatory data to perform translation

Postconditions

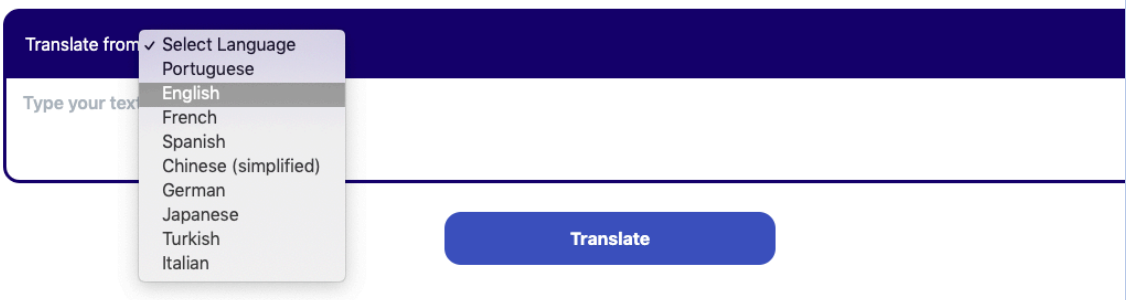
1- A translation request will be created with an associated status, with progress the translation status will change.

Main Flow

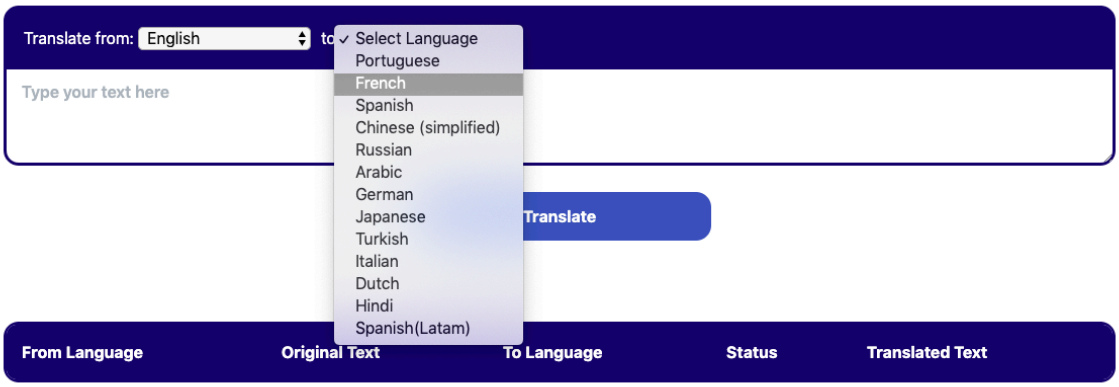
1 – Go to Unbabel Home page with the url: **http://localhost:8080**
The home page should be loaded:



2- Select Source Language clicking on the first dropdown box
A second dropdown box should be loaded into the form.

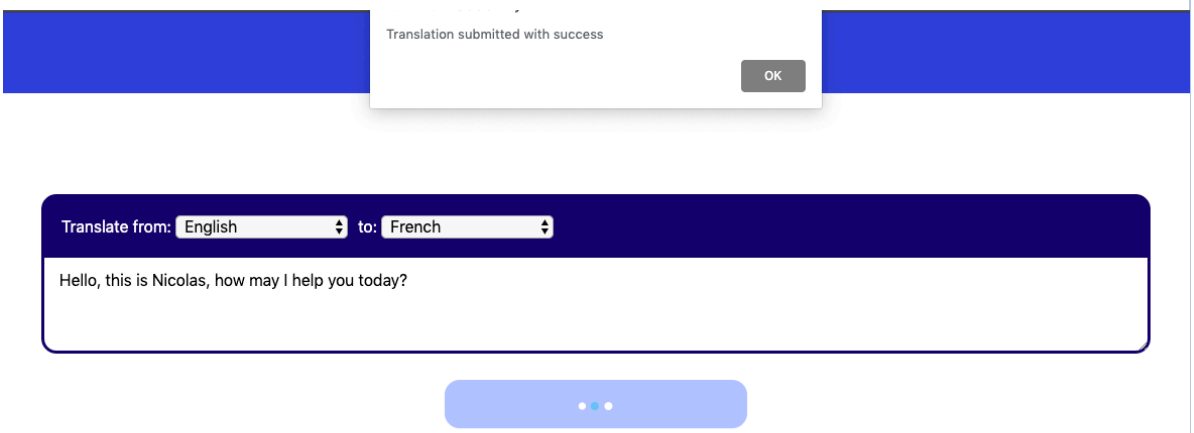


3-Select Target Language from the second dropdown box



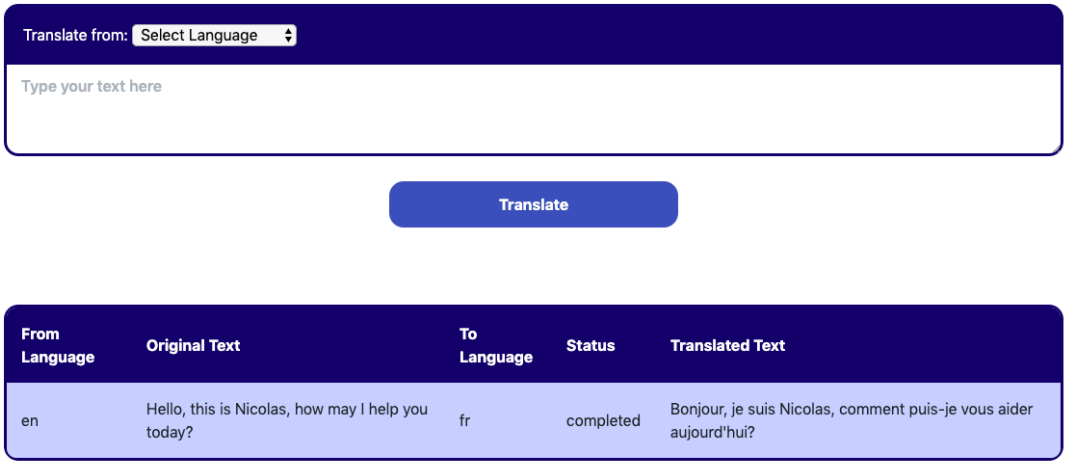
4 – Type the text you want to translate in the selected source language to the selected target language

5 – Click on the Translate button to submit the translation
A success message should be displayed, and a new entry will be loaded into the table



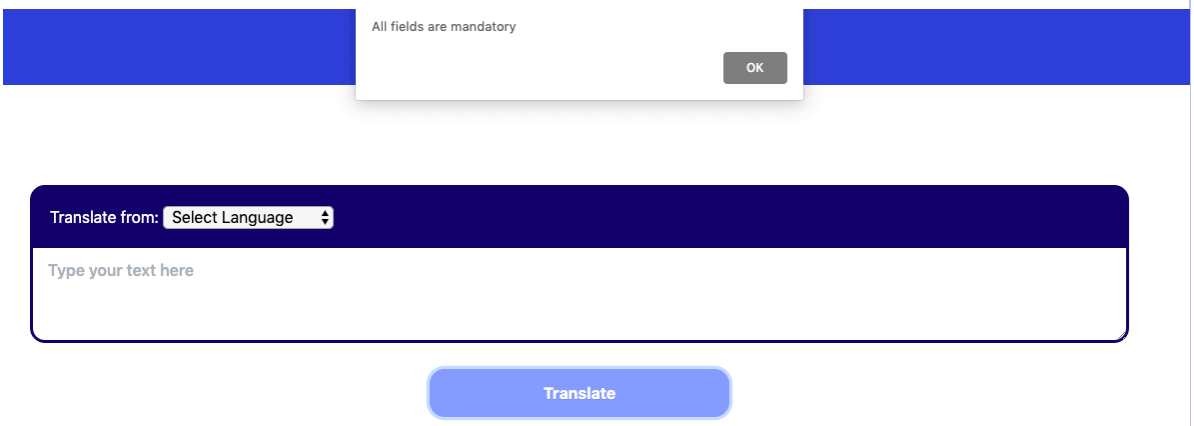
6 – Close the message warning to see the new entry in the table

7 – To update the table information, just refresh the page view in your browser



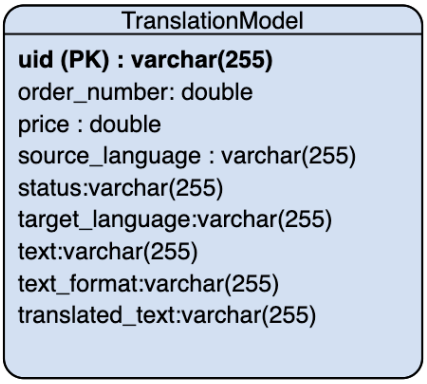
Alternative Flow

1 – Click on “Translate” button without any field filled
A validation message will be displayed preventing the form submission

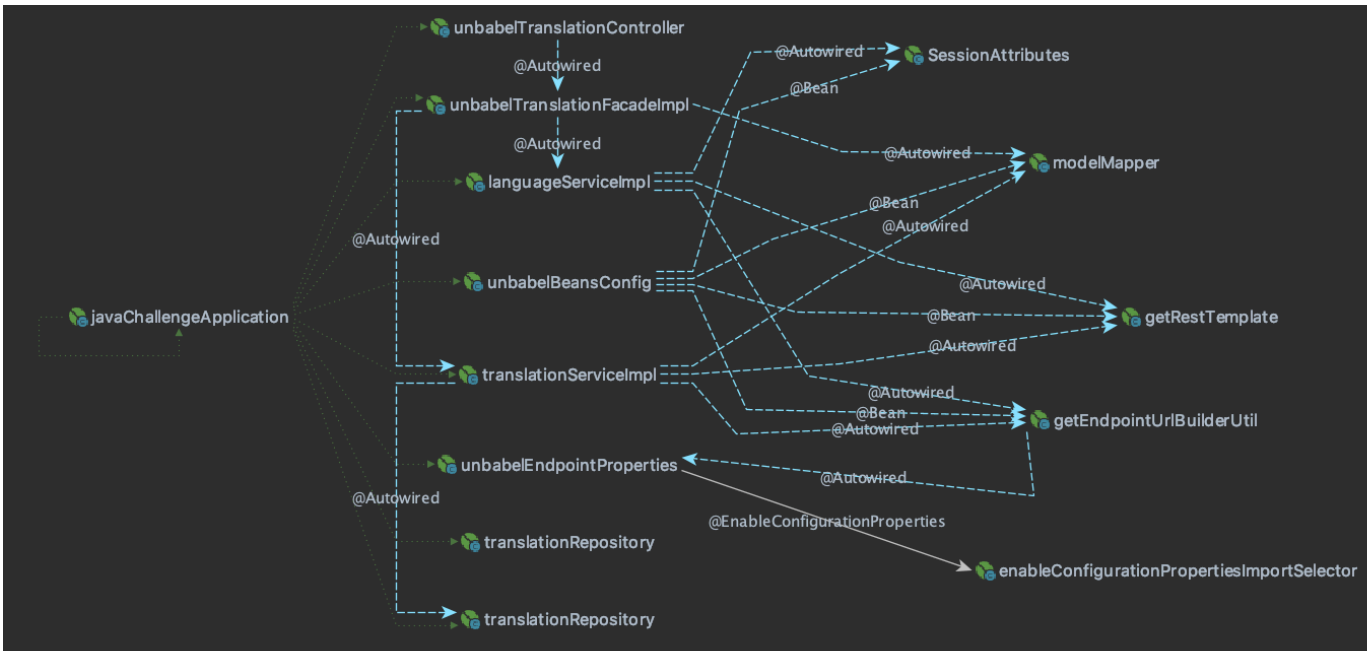


Data Model

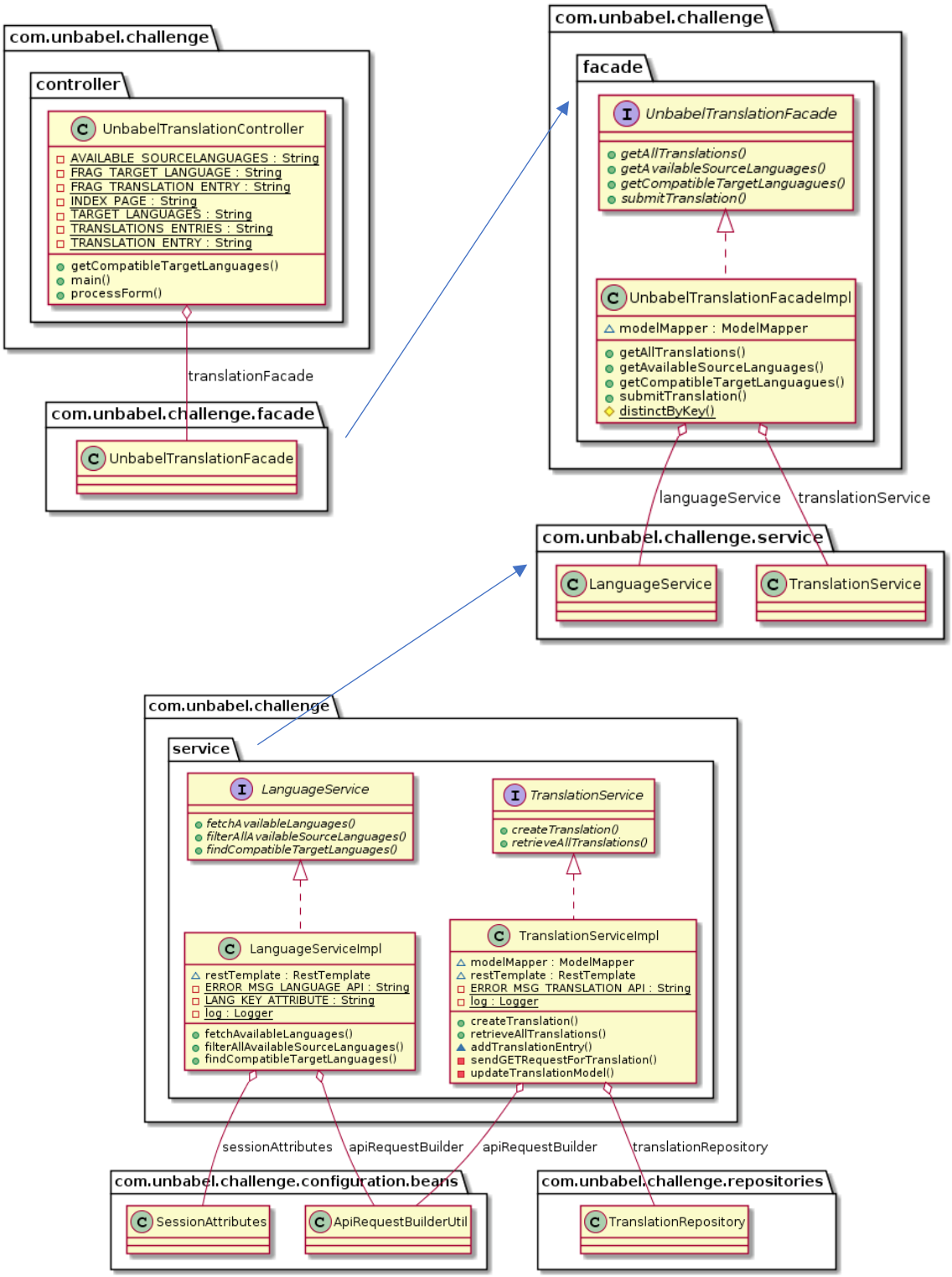
Since we don't have user authentication and other relevant information to store in this application, we only need a table to track the current user translations and to cache the status about the correspondent translations



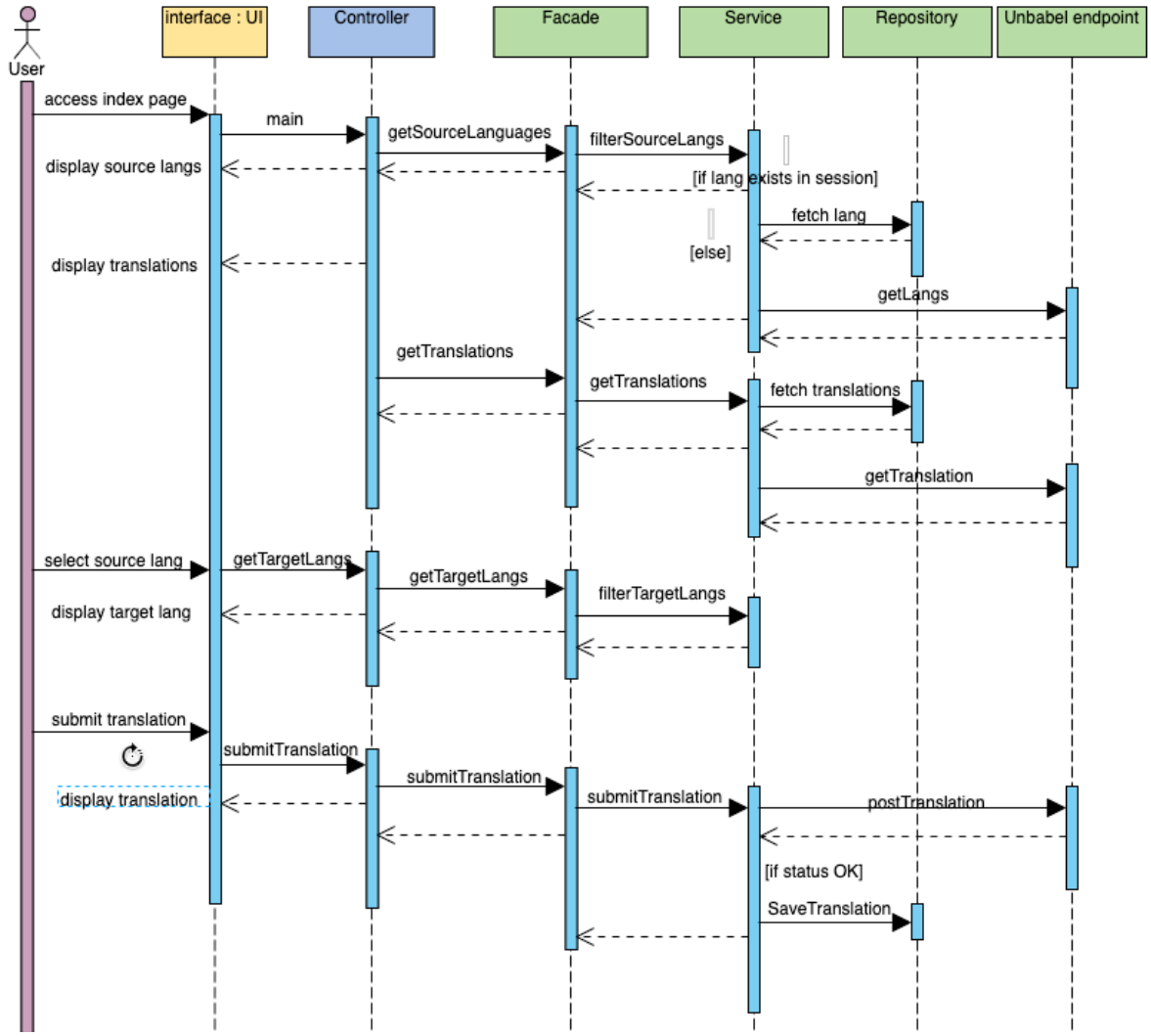
Spring Bean Diagram



Class Diagram



Sequence Diagrams



To improve

1 – A big improvement would be the migration of the frontend framework, instead of using a server-side rendering framework a client-side framework could be implemented using Angular or React, with this you could have a modular implementation of the backend and frontend. With this you could have a frontend more scalable and testable. My framework of choice would be React because of the virtual DOM feature.

2 – Improvements in my code:

- Full coverage in Unit and Integration Test (didn't provide full coverage, I only provide the necessary examples)
- Implementation of Selenium IDE testing
- Refactor TranslationServiceImpl.java methods: currently the methods are implementing more than two actions, API and database business logic, for have a system more scalable a method should only perform one action, this way we keep our implementation more modular. Another service or method should be created for database related business logic.
- Improve fetching translation information from API: in my current implementation, when calling the translate API to retrieve information about a translation, if this call fails, a translation entry will be lost, a call retry should be implemented to avoid data loss. Another important improvement, when the endpoint for some reason is down, the local cached information about the translations already made should be displayed