EXPERIS IT
Wauthia Andréa

Frontend & Backend Test

Table des matières

CONTEXT

As part of the technical test, SERVITIA requests the development of a message exchange application between users. This document serves as a functional and technical analysis of the issued requirements.

FUNCTIONAL ANALYSIS

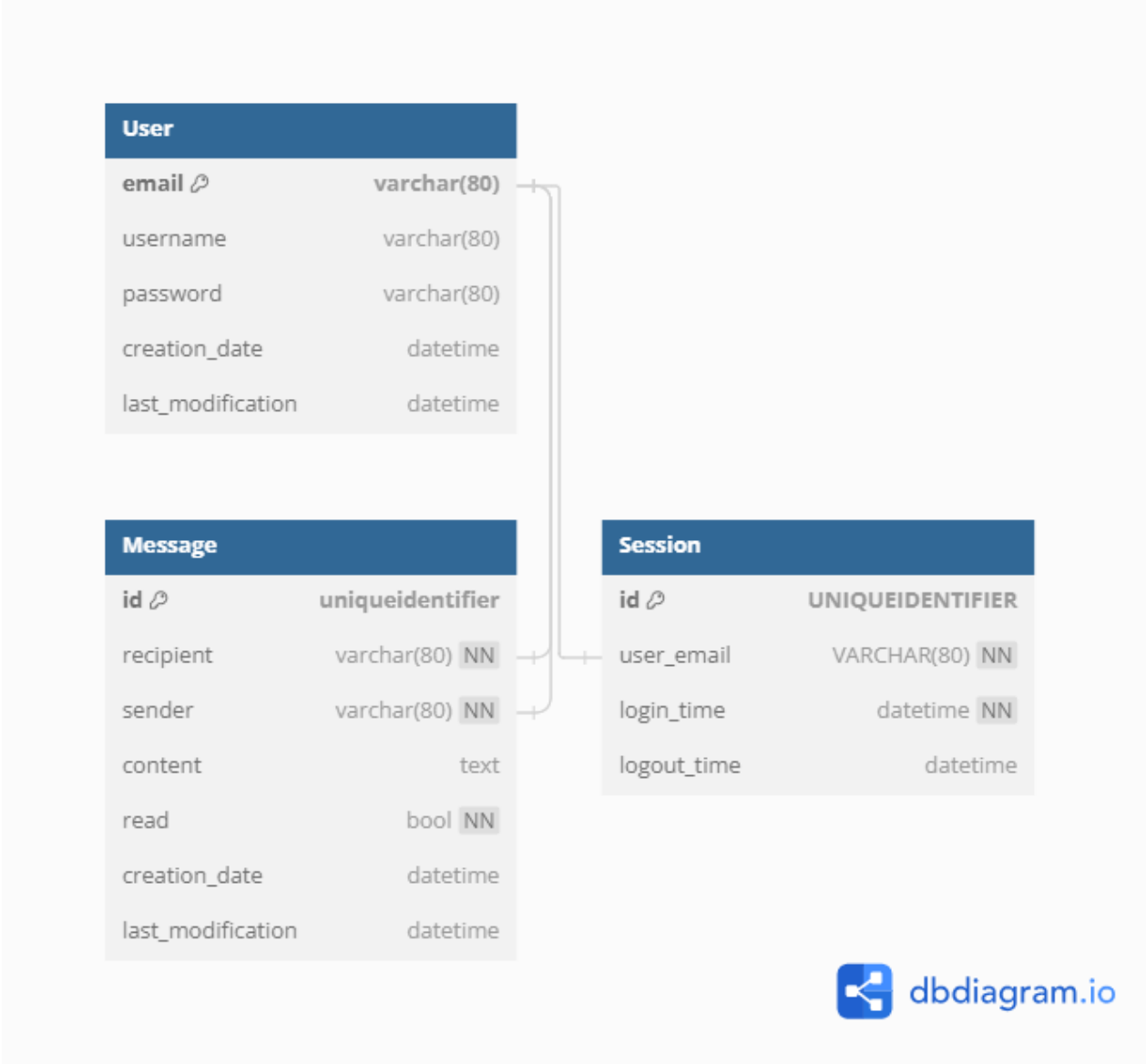
User story

- User Authentication
  As a user, I want to log in to the application with my email and password to access the messaging service.
    - Acceptance Criteria:
        - The user should be able to enter their email and password.
        - On success, the user is redirected to the main page.
        - On failure, a generic error message is displayed (account creation error handling not included).

- Display Logged-in User Email
  As a logged-in user, I want to see my email displayed on the main page to confirm my identity.
- Send Messages
  As a logged-in user, I want to send messages to other users by selecting a recipient from a list and entering my message.
    - Acceptance Criteria:
        - The user should be able to select another user (excluding themselves) from a dropdown list
        - The user should be able to enter a message in a text area and click "Send."
        - The message should appear in the exchanged messages list.

- Prevent Self-Messaging
  As a logged-in user, I do not want to be able to send messages to myself to avoid errors.
- Real-time Messaging
  As a user, I want messages to be exchanged in real time for instant communication.
    - Acceptance Criteria:
        - Sent and received messages should appear in real-time in the message list without requiring a manual page refresh.
- Receive Notifications for New Messages
  As a logged-in user, I want to receive notifications when a new message is sent to me to be informed immediately.
- View Message History
  As a logged-in user, I want to see a list of all messages exchanged with other users to follow the conversation.
- Refresh Message List
  As a logged-in user, I want to click on a notification to refresh the message list and see the new messages received.
- Logout
  As a logged-in user, I want to log out of the application to secure my account when not in use.

Database design



Api

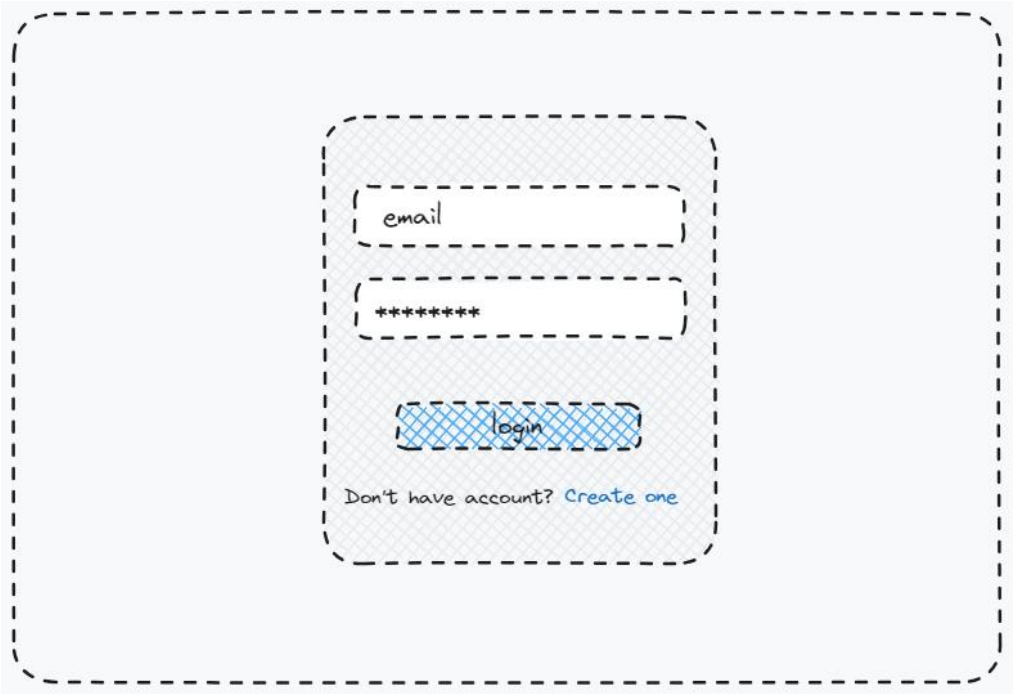
- Authentication
  - Login
    Parameter : email, password
    Description : Check if an user is record with this password and craft a jwttoken, and create a new session in database.
    Error : InvalidData if no user is found
    Out : AuthToken
  - Logout
    Parameter : email
    Description : End the session
    Out : boolean

- User
  - Create
    Parameter : email, password, username
    Description : Create a new user
    Error : InvalidDataException if an user already exists with this mail
    Out : User
  - GetRecipients
    Parameter : email
    Description : Get all other users and a flag if this user send an unread message.
    Out : List<Recipient> (Recipient : User + HasMessage)
- Message
  - Create
    Parameter : Message
    Description : Add a new Message in the database then send a notification to the recipient
    Out : Message
  - GetUnreadMessage
    Parameter : email (recipient)
    Description : Get the number of unread message sent by the recipient
    Out : int
  - GetChatWith
    Parameter : emailSender, emailRecipient
    Description : Get all the messages exchanged by the stakeholder soted by datetime, and mark the unread message sent by the recipient as read and then send a notification to the recipient to said that the discussion is updated
    Out : List<Message>

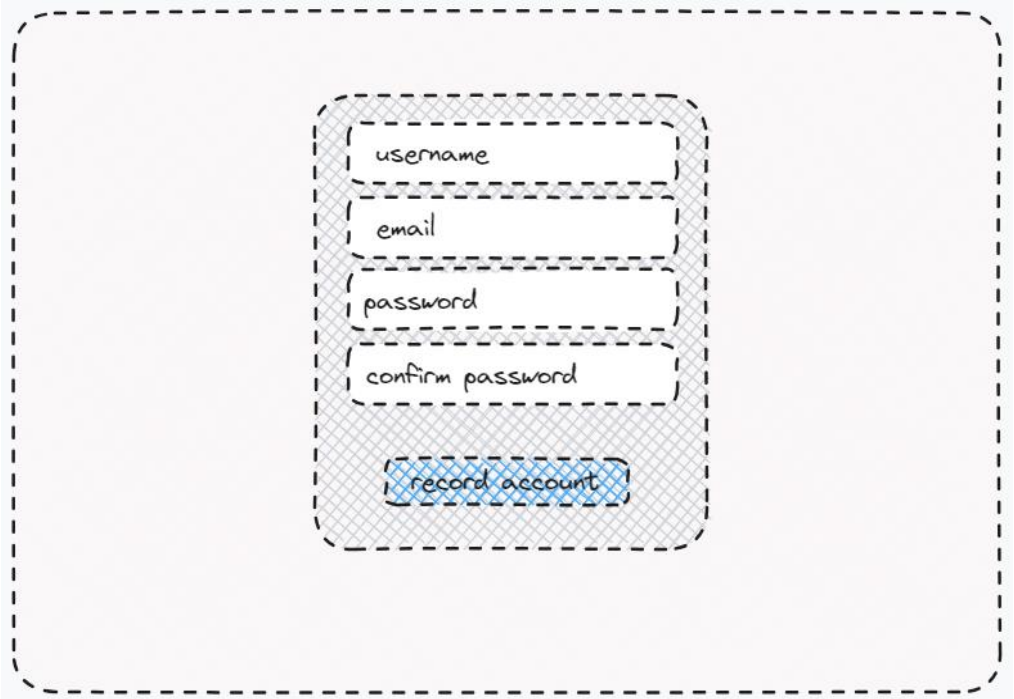
Front component

For those requirements, 6 mains components are identified :

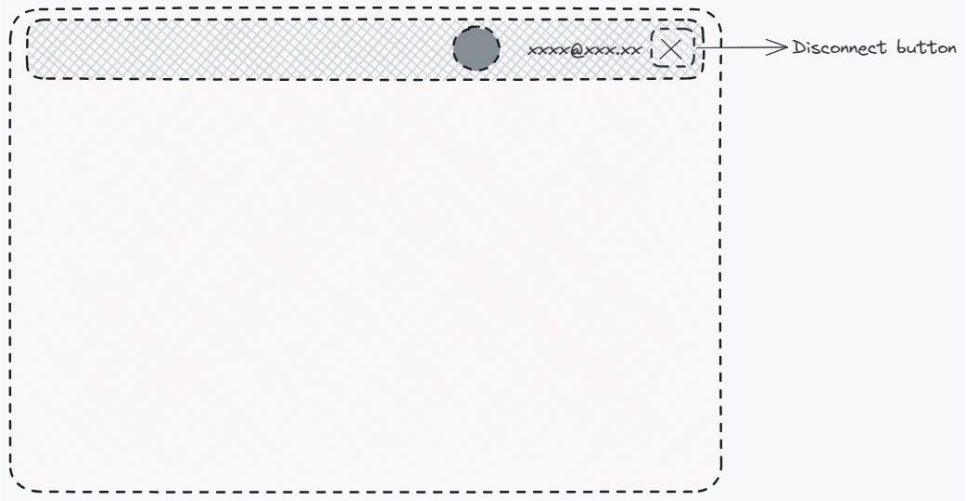
- Login form



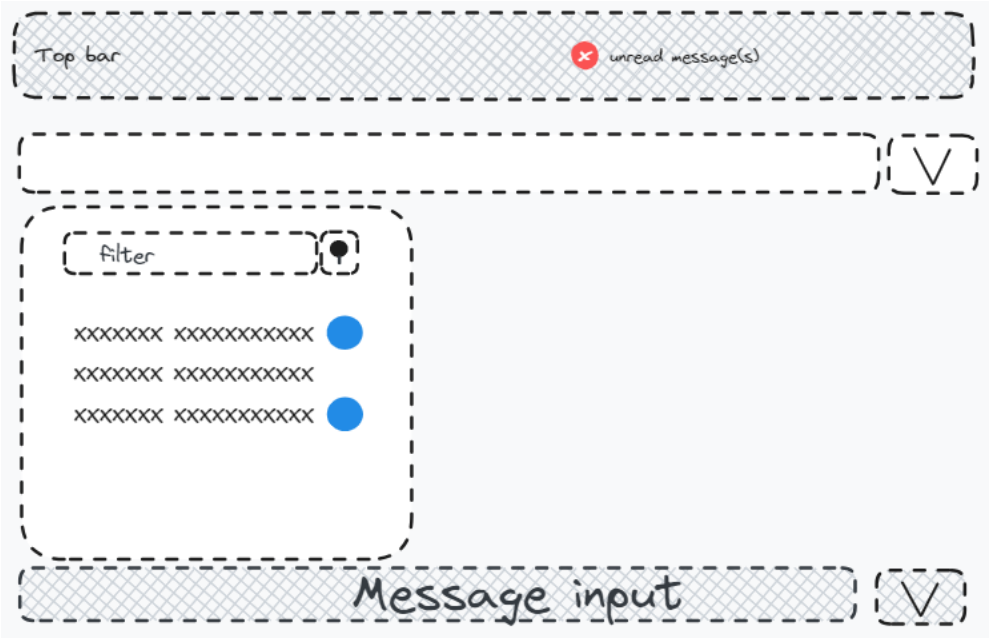
- Sign up form



- Topbar



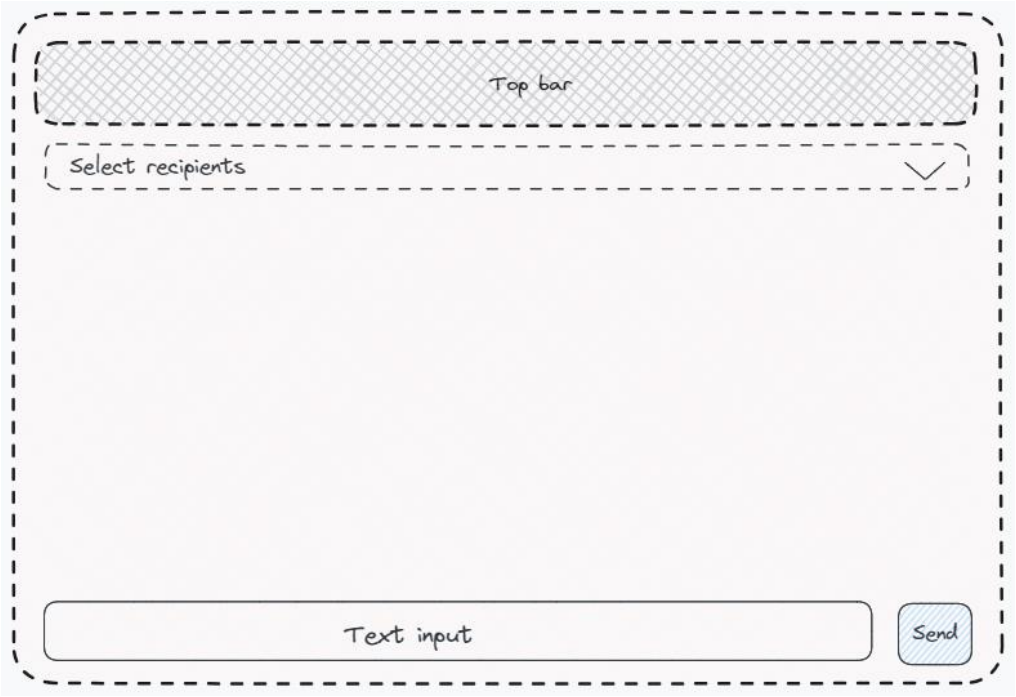
- Recipients selection dropdown



This component can be splitted in 3 minor components

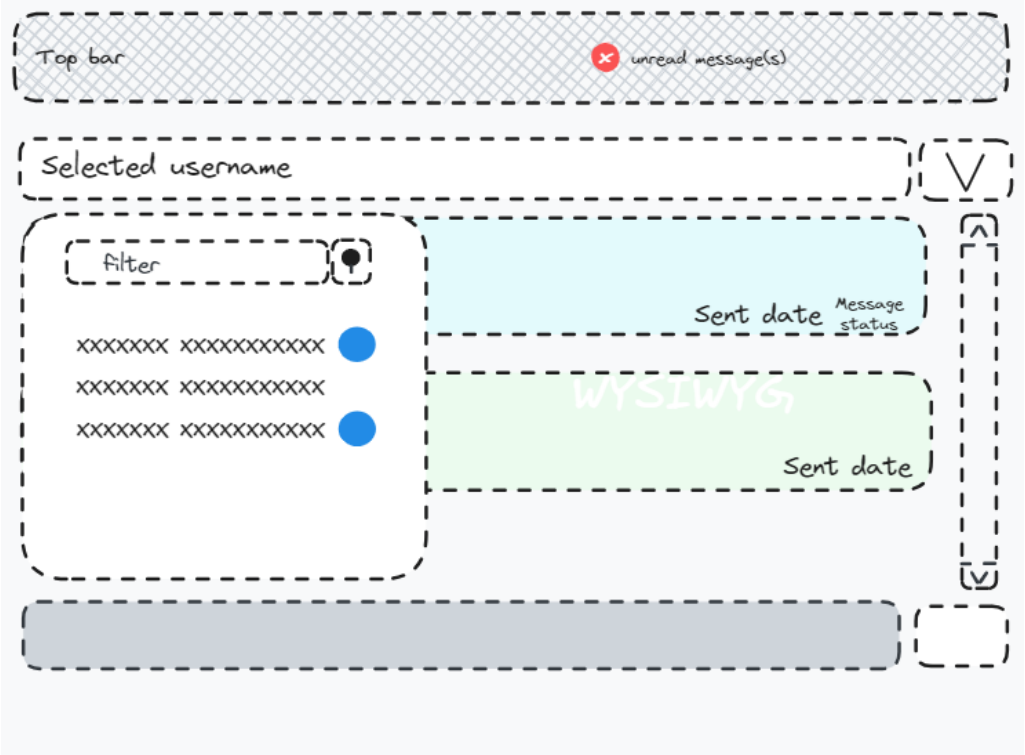
- The text input
- The button that will generate the dropdown menu
- The popup with the list

- Send message area



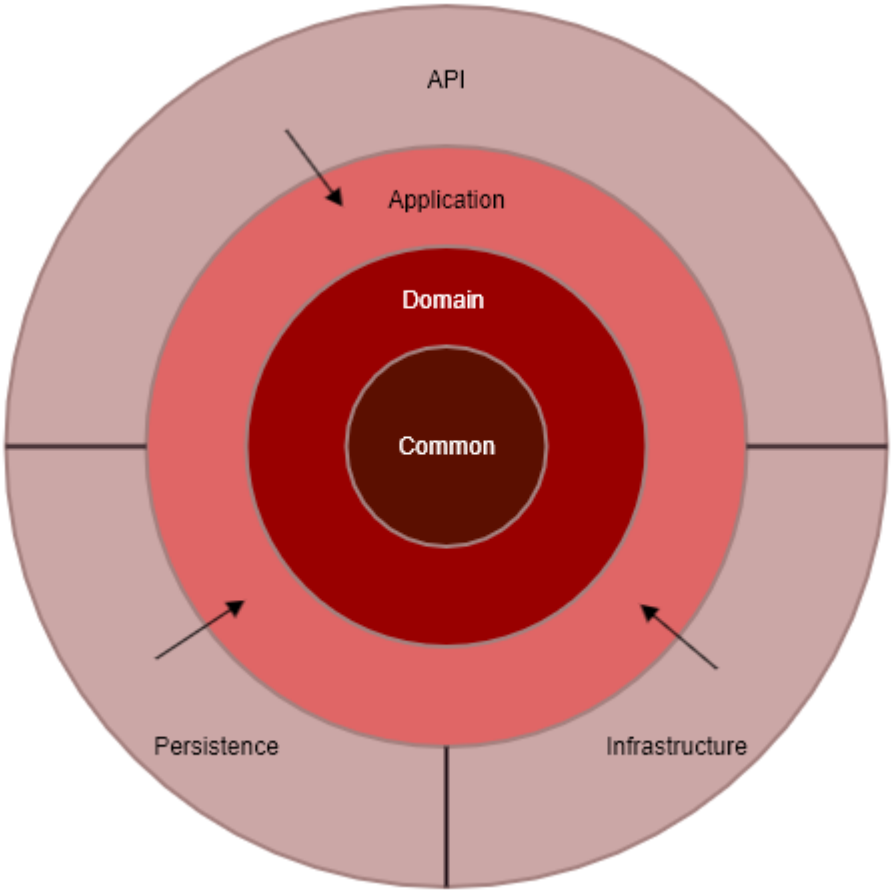
I override the original specification by rendering the text are on the bottom.

- Message history



Backend architecture

The clean architecture will be used for this backend application, this allows a better separation, modularity and an independance of framework despite the complexity of the initiation of the project.



- Common

Contains all common objects, configuration objects or extension methods. No dependency are required on this layer.

- Domain

Contains all persisted object

- Application

Contains all business logic of the application

- Persistence

Contains all database persistence logic

- Infrastructure

Contains all external dependency such smtp server, external Api

- Api

Contains all exposed Api, this layer is the entry point of the web application and will communicate with the application layer via a messaging library such as Mediatr, RabbitMQ.

Frontend architecture

In order to improve the maintenance of the project, I'll use the Domain Driven Development architecture in the frontend project.
This article represents a guideline of the good practice (https://alaedev.medium.com/applying-domain-driven-design-ddd-principles-in-frontend-development-fc67c8d28cc9)

TECHNICAL CHOICES

Backend

- Net7
- Mediatr
- EFCore
- SignalR

Frontend

- Vite (faster to deploy in this case)
- Axios
- React 18
- Tailwind css

The backend and frontend are completely separated.

SECURITY CONCERNS

This application is purposed to be a demonstrator of skill, so the security is not the main priority. For a production application, the message between the front and the backend should be encrypted.

But in order to implement a proper way to encrypt the exchange between the front and the backend, we need to implement an interceptor in the front and IResourceFilter, IAsyncActionFilter in the backend...