Task and project management system

Project Context: The idea is to create an online platform that allows users to organize, manage and follow up their tasks and projects in an efficient way. The application should have functionalities such as creating, editing, viewing and deleting tasks, according to the immediate needs of the user.

1. Definition of Quality Objectives: We identified the following quality objectives:

- **Usability:** The system should be easy to learn and use so that users can intuitively add, edit and complete tasks. A clean and friendly user interface is essential.
- **Performance:** The system must be fast and efficient, allowing users to perform operations without significant delays. This is especially important as the number of tasks and users increases.
- **Availability:** The system must be available when users need it. This implies high availability and the ability to recover quickly from failures.
- **Customization:** Users must have the ability to customize the system according to their individual needs, such as the organization of tasks, labels and priorities.
- **Maintainability:** The system architecture should be modular and easy to maintain. This implies clean code, adequate documentation and the ability to perform upgrades without significant interruptions.

2. Scenario Identification: We define specific scenarios to evaluate each objective:

- **Usability Scenario:** "The user needs to easily and intuitively create a task".
- **Performance Scenario:** "The application will allow to manage the active tasks of any user quickly even when the application has a high volume of active users at the same time".
- **Availability Scenario:** "The user will be able to access his dashboard at any time of the day".
- **Personalization Scenario:** "The system will allow the dashboard to be customized to the user's liking and will allow the user to adjust the labels according to the work he/she is doing".

3. Creation of the Initial Architecture Model: For this deliverable, we developed an architectural model that includes components for authentication and task management of a user. This model can be found in the file attached to this document called "Model C4 - Pre Trade-offs.pdf".

4. Preliminary Analysis:

- We evaluate how the architecture addresses each scenario. For example, for the security scenario, we review how access controls and encryption are handled.
- We identify strengths and potential weaknesses in the initial architecture in relation to the quality objectives.

5. Identification of trade-offs:

- **Usability vs. customization:** We found that, to improve usability, we could decrease the scope of customization, because a highly customizable system with different pages serving different functions may be more difficult to learn and use, due to users having to familiarize themselves with a variety of options and settings.

6. Iteration and Refinement: We made adjustments to the architecture to address identified trade-offs.

The change made to the initial C4 model was as follows:

Component Diagram: We reduced the "Main Page" to a single page called "DashBoard Page" so that it can function as an SPA. As the "Main Page" was reduced, we detail how the "Task Management API" works for its communication with the "DashBoard". This is in order to improve its usability so as not to confuse the user with so many pages. Personalization was reduced as a consequence.

The file called "Model C4 - Post Trade-offs.pdf" is attached to this document for better understanding.

7. Results:

The comprehensive analysis of the quality attributes provided us with a deeper understanding of how decisions directly influence the implementation of an architectural model. This process allowed us to carefully document the trade-offs identified, as well as the crucial decisions made and improvements actually implemented in the architecture.

The trade-off identified in our evaluation was improved usability in exchange for reduced customization. This is because a highly customizable system, with multiple pages serving different functions, could complicate the user experience by requiring users to become familiar with a wide variety of options and settings.