Task Management Project

1stJavier Murcia
Faculty of engineering, systems engineering
Distrital University Fransisco Jose de Caldas
Bogotá, Colombia
jcmurcian@udistrital.edu.co

2nd Andres Acevedo
Faculty of engineering, systems engineering
Distrital University Fransisco Jose de Caldas
Bogotá, Colombia
ajacevedoa@udistrital.edu.co

***Abstract*—blank**

***Index Terms*—blank**

I. Introduction

In these times, the efficient organization of tasks and projects has become something really fundamental for people and companies of all kinds, since the optimization of time has become essential, since the more optimized the tasks and projects are, the better development can be achieved. The complexity and magnitude of work activities, in addition to the large amount of information that has to be processed every day, can result in chaos if the tools and applications that are available are not used and taken advantage of. This can bring as a consequence a decrease in productivity, with difficulties with deadlines and objectives, as well as bringing consequences for team members such as increased stress and frustration. Taking into account that this is also a recurring problem among students, the decision has been made to replicate an application of this style called "Focus to do" by replicating this we will be able to understand how a task and project management app works, which allows to organize their activities, increase their productivity and maintain better control over their projects to those who wish to make such optimization, The application will be deconstructed in a data model based on object-oriented programming that allows efficient management of different entities, such as tasks, subtasks, projects, customers, among others, to understand the development of such application. The main objective of this project is to identify and replicate a comprehensive tool that will be of help to anyone who aims to improve their efficiency and organization with respect to their work or study, identifying key functionalities such as creation and tracking of tasks, organization of activities, generation of statistics and/or reports, Pomodoro type timer and all the functionalities associated with this type of applications. On the other hand, it will allow you to understand the whole development behind such an application, from its planning, to its theoretical development, diagrams and its implementation.

II. Methods and Materials

First, we made a definition of the **business model**, that consist in a task management platform that operates on a freemium business model, offering basic features for free, such as starting a timer for the Pomodoro technique. However, if you desire premium features like advanced task analytics and better organization, you have to buy a subscription. After defining the model, the business rules were defined, which are:

- The client must provide a valid email address and create a password to register an account.
- Each email can only be associated with one account.
- If the client wants to access the app, they must enter their registered email along with their corresponding password.
- To access premium features, the client must have purchased the premium subscription.
- Subscription plans can only be purchased on a monthly or annual basis.
- Clients can create, edit, and delete tasks.
- Clients without a premium subscription can create a maximum of 5 projects.
- Clients with a premium subscription can create folders.
- Clients with premium subscriptions can view productivity statistics.
- Tasks must have attributes such as description (mandatory), due date, priority, and tags.
- Clients can set a timer to use the Pomodoro technique.
- By default, the Pomodoro lasts 25 minutes, the short break 5 minutes, and the long break 15 minutes.
- Clients can edit the default times for the Pomodoro, short break, and long break.
- Clients can organize tasks into projects for better organization.
- Clients can provide feedback and report issues.

According to the business rules, we define user stories, to understand the needs and requirements of end users. That's how the following stories were established, as seen below:

- **As a** client, **I want** to be able to create, edit and delete tasks and subtasks, **so what** I can organize my work and daily activities.
- **As a** client, **I want** to be able to set a custom Pomodoro timer, **so what** I can improve my focus and productivity.
- **As a** client, **I want** to be able to assign priorities and tags to my tasks, **so what** I can determine their importance and categorize them to plan my time more efficiently.
- **As a** client, **I want** to receive notifications, **so what** I can remember pending tasks and Pomodoro breaks.
- **As a** client, **I want** to see all plans for subscriptions, **so what** I can choose the best plan for my needs.
- **As a** client, **I want** to pay for subscriptions, **so what** I

can access to all features of the platform.

- **As a** premium client, **I want** to be able to see my report of completed tasks and the time spent on each one, **so what** I can evaluate my productivity.
- **As a** premium client, **I want** to be able to automatically repeat tasks, **so what** I can plan my work more efficiently.
- **As a** premium client, **I want** to be able to create unlimited projects, **so what** I can incorporate all areas of my life into productivity.
- **As a** premium client, **I want** to be able to create folders, **so what** I can organize my projects by life categories.
- **As an** admin, **I want** to have a report of the number of clients, **so what** I can make decisions to increase the number of clients.
- **As an** admin, **I want** to have a report of the number of subscribed clients, **so what** I can make decisions to increase the number of premium clients.

With the user stories, the definition of entities, along with their attributes and behaviors, was facilitated. So, based on the stories, the following entities were defined:

- **User**: with attributes such as name, ID, email, and password, and methods for login and logout.
- **Administrator** (inherits from User): with additional methods to get reports of clients and premium clients.
- **Client** (inherits from User): with methods to create, edit, and delete tasks, subtasks, tags, projects, as well as other functionalities such as receiving notifications, using the Pomodoro timer, viewing plans, and subscribing.
- **PremiumClient** (inherits from Client): with additional methods to view productivity statistics, create, edit, and delete folders, and repeat tasks.

Other key entities include Task, Subtask, Tag, Notification, Pomodoro, Project, Folder, Plan, Subscription, Report, TaskStats, and ClientsStats. These entities are related to each other according to the application's needs, implementing object-oriented design principles such as inheritance, composition, and agregation. Also, with the definition of entities and their behaviors, we define the processes of the software, breaking it down one by one, we have several processes that the software performs, as can be seen below:

- Login
- Logout
- Create a Task
- Create Tag
- Create a Project
- Create a Folder
- Create Subtask
- Edit a Task
- Edit a Project
- Edit a Folder

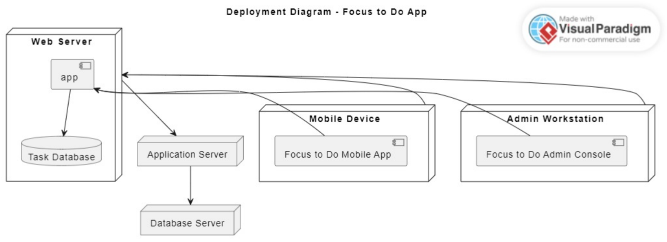

Fig. 1. Deployment Diagram

- Edit Subtask
- Delete a Task
- Delete Tag
- Delete a Project
- Delete a Folder
- Delete Subtask
- View Tasks
- View Projects
- View Folders
- View Subtasks
- Start a Pomodoro
- Stop a Pomodoro
- Customize a Pomodoro
- See Plans
- Buy Subscription
- Create Task Stats
- View Task Stats
- Create Folder
- Edit Folder
- Delete Folder
- Repeat Tasks
- Send Notification
- Show Notifications
- Add a Task to the Project
- Remove a Task from the Project
- Add a Project to the Folder
- Remove a Project from the Folder
- Add a Client to the Subscription
- Remove a Client from the Subscription
- Create a Client Report
- Create a Task Report

To build the conceptual model, we made a deployment diagram, some activity diagrams, a sequence diagram, and a state diagram. Each diagram were made according to the business model, business rules and user stories.

So, with *www.websequencediagrams.com* we made a deployment diagram, to visualize how the software elements, are deployed in the execution environment. We can see this diagram in the Figure 1.

Then, with processes that we identified, we made an activity diagram for each one, although some activities had very similar flows, so some were merged into a single diagram. These allows visualize the control flow of each activity. In figure

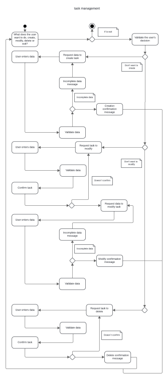

Fig. 2. Activity diagram of create, edit or delete task

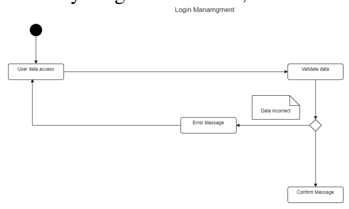

Fig. 3. Activity Diagram of login

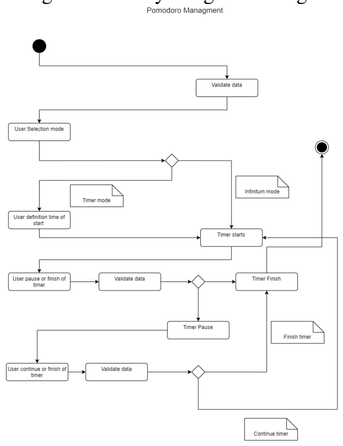

Fig. 4. Activity diagram of Pomodoro

2, figure 3, and figure 4 you can see some of the diagrams created Activity diagrams were made taking into account the specific functionalities of the FocusToDo application.

Finally, for the conceptual model of the project, we made a sequence diagram, that can be seen in Figure 5. And the state diagram, that can be seen in Figure 6.

To define the backend architecure of the project, we made CRC cards, that we can observe in Figure 7, so we can understand the functions of each class in a general way.

These CRC Cards and user stories were greatly useful in the construction of the Class Diagram, that we can observe in Figure 8, because it helped to establish the relations between entities.

For the application implementation, modern technologies were used, such as programming languages (Python)... Additionally, software engineering best practices were implemented, such as unit testing, and extensive documentation.

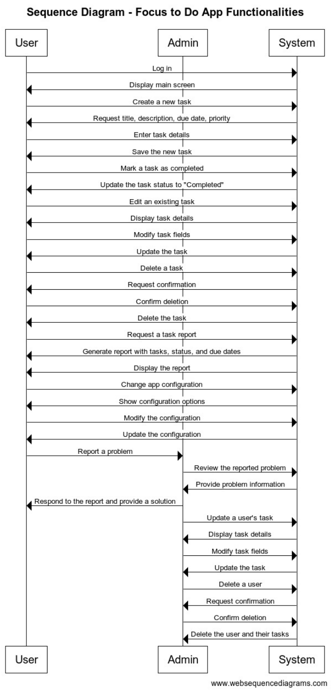

Fig. 5. Deployment Diagram

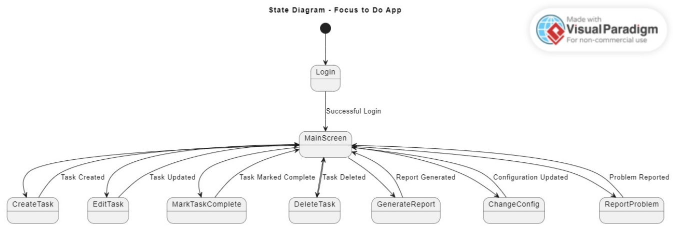

Fig. 6. StateDiagram

The application's workflow begins with user registration and authentication, where users can be either clients or administrators. Clients can create, edit, and delete tasks, subtasks, tags, and projects, as well as receive notifications, start and pause the Pomodoro timer, and subscribe to plans. Premium clients have additional functionalities, such as generating productivity statistics, managing folders, and repeating tasks. While administrators can get reports of clients and premium clients, which allows them to monitor the application's usage and make informed decisions about developing new functionalities or improving existing ones.

III. Experiments

To be able to run tests on the application, it is essential to determine what information will be persistently stored by default in a database or any other data storage system being used. The Python ‘faker‘ library allows you to generate simulated data for the data that will be used by the database

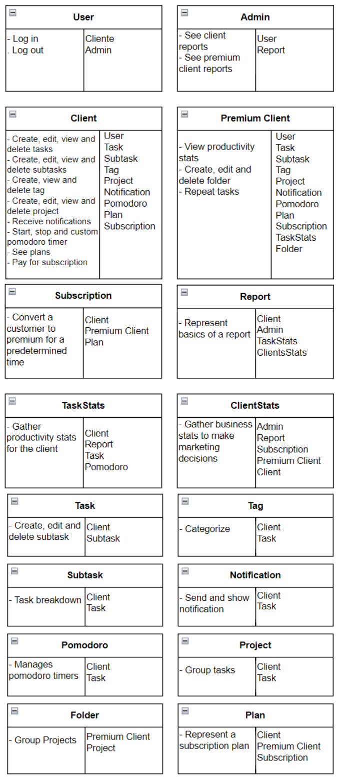

Fig. 7. CRC Cards

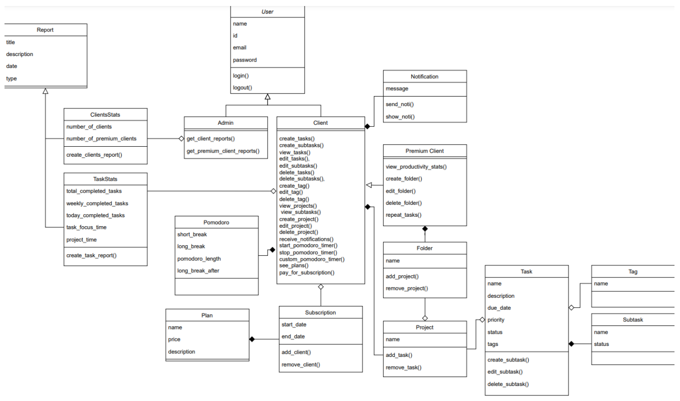

Fig. 8. Diagram Class

with information from a realistic test. Here are the data that should be persistently stored during the tests: begin

- Users: Name, Email, Password, Subscription Status
- Tasks: Description, Due Date, Priority, Tag, Status, Assigned User
- Projects: Project Name, Description, List of Associated Tasks, Project Owner User
- Subtasks: Description, Status, Main Task they are associated with
- Subscription Plans: Plan Name, Price, Description of Included Features
- Subscriptions: Associated Client, Subscription Plan, Start Date, End Date
- Folders: Folder Name, List of Associated Projects, Folder Owner User
- Task Statistics: Total Tasks Completed by User, Time Spent on Each Task by User
- Client Statistics: Total Registered Clients, Total Clients with Premium Subscriptions

Using the ‘faker‘ library, you can generate random data for each of these areas.

References

[1] book ”Agile Project Management” of Jim Highsmith. .