Final Paper Project
Focus To Do, Task Management Platform

1st Javier 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

***Index Terms*—Focus, diagrams, Python**

I. Abstract

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. 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.

III. 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:

A. Business Rules

- Each username can only be associated with one account.

- If the client wants to access the app, they must enter their registered username along with their corresponding password.
- To access premium features, the client must have purchased the premium subscription.
- Subscription plans can only be purchased monthly or annually.
- Clients can create, and delete tasks
- Clients with a premium subscription can create folders.
- Clients with premium subscriptions can view productivity statistics.
- 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.

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** that automatically tasks and subtasks are deleted when are done **so what**I can get rid of tasks I don't need.
- **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 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** 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 add tag to my tasks, **so what** I can categorize them to plan my time 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 and premium, **so what** I can make decisions to increase the number of clients and 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:

- **Client**: with attributes such as nameUser and password, and methods for login and logout, also, 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.
- **Administrator** (inherits from client): with additional methods to get reports of clients and premium clients, also, to modify the plans offered to clients.
- **PremiumClient** (inherits from Client): with additional methods to view productivity statistics, create, edit, and delete folders.

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
- 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 activity diagrams, a sequence diagram, and a class diagram. Each diagram were made according to the business model, business rules and user stories. 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.

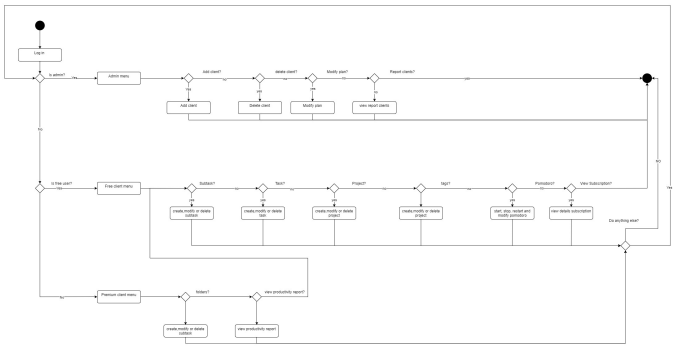

Fig. 1. Activity diagram

Activity diagrams were made taking into account the specific functionalities of the FocusToDo application.

The diagram starts with the user's login. Then, it checks if the user is an administrator. If he is an administrator, he accesses the administrator menu where he can add, delete or modify clients, as well as view client reports. If not an administrator, the system checks whether the user is a free or premium user. Free users access a menu where they can manage subtasks, tasks, projects, labels and the pomodoro. Premium users have an additional menu where they can manage folders and view productivity reports. Both free and premium users can create, modify or delete subtasks, tasks, projects and tags, as well as manage the pomodoro and view details of their subscriptions. Finally, after performing any action, users can choose to do something else or log out of the system.

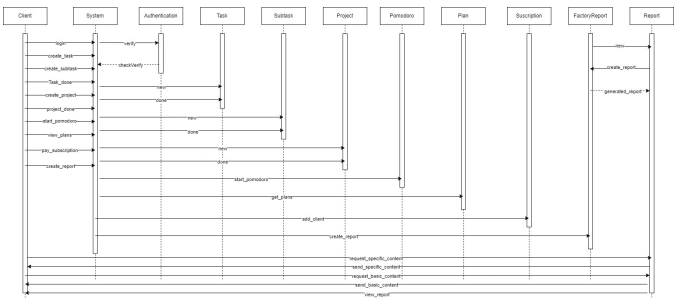

Fig. 2. Sequence diagram

For the conceptual model of the project, we made a sequence diagram, that can be seen in Figure. This sequence diagram shows the interaction between the client and various system components (such as authentication, tasks, subtasks, projects, pomodoro, plans, subscription and reports) for different actions. The client logs in, creates and completes tasks and subtasks, starts and manages pomodoros, views and pays for subscription plans, and requests the creation of reports. Each customer action is verified and processed through the corresponding components, which respond with the necessary confirmations and data to complete the operations.

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

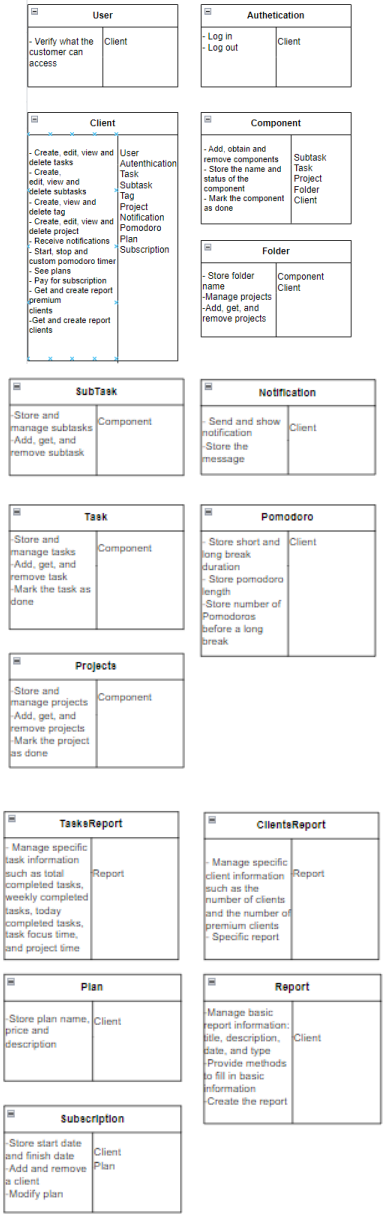

Fig. 3. CRC Cards

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

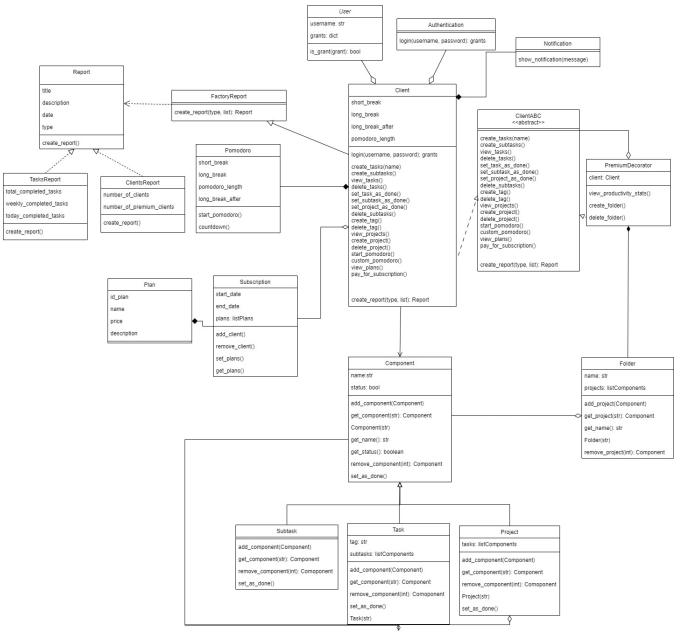

Fig. 4. Diagram Class

Finally, in this class diagram it can be seen that tasks and projects are made up of individual Components. Subtasks are grouped into Tasks, which in turn are organized into Projects. Folders allow structuring these elements. The Report class generates different types of reports on completed tasks, customers and productivity.

The Plan/Subscription class models subscription plans for customers, with attributes such as start and end dates, name and price. The system handles adding, deleting and obtaining plans. Overall, the diagram shows a detailed class structure covering user authentication, customer management, projects, tasks, components, reports and subscriptions, providing a complete representation of the system architecture.

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.

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.

B. Tools to Use

In this case, the backend will be built using Python 3.11, and some related technologies such as Fast API to serve functionalities, PyTest to apply some simple unit tests, and Black to auto-format the code and increase code readability.

C. Entities

These are the entities that were created

- User
  - Username: str
  - Grants: dict
  - IsGrant(grant: bool)
- Authentication
  - Login(username, password): grants
- Admin (User)
  - GetClientReports()
  - GetPremiumClientReports()
- Client (User)
  - ShortBreak
  - LongBreak
  - PomodoroLength
  - LongBreakAfter
  - CreateTasks(name)
  - CreateSubtasks(name)
  - ViewTasks()
  - EditTasks()
  - SetTasksAsDone()
  - DeleteTasks()
  - DeleteSubtasks()
  - CreateTag()
  - EditTag()
  - DeleteTag()
  - ViewProjects()
  - ViewSubtasks()
  - CreateProject()
  - EditProject()
  - DeleteProject()
  - ReceiveNotifications()
  - StartPomodoro()
  - StopPomodoro()
  - CustomPomodoro()
  - ViewPlans()
  - PayForSubscription()
  - CreateReport(type, list): Report
- PremiumDecorator (Client)
  - ViewProductivityStats()
  - CreateFolder()
  - EditFolder()
  - DeleteFolder()
- Task
  - Name: str

  - Status: boolean
  - Tag: str
  - Subtasks: list[Components]
  - AddComponent(Component)
  - GetComponent(str): Component
  - RemoveComponent(int): Component
  - SetAsDone()
- Subtask
  - Name: str
  - Status: boolean
  - AddComponent(Component)
  - GetComponent(str): Component
  - RemoveComponent(int): Component
  - SetAsDone()
- Project
  - Name: str
  - Tasks: list[Components]
  - AddComponent(Component)
  - GetComponent(str): Component
  - RemoveComponent(int): Component
  - SetAsDone()
- Folder
  - Name: str
  - Projects: list[Components]
  - AddProject(Component)
  - RemoveProject(int): Component
- Component
  - Name: str
  - Status: boolean
  - AddComponent(Component)
  - GetComponent(str): Component
  - RemoveComponent(int): Component
  - SetAsDone()
- Plan
  - IdPlan
  - Name
  - Price
  - Description
- Subscription
  - StartDate
  - EndDate
  - Plans: list[Plans]
  - AddClient()
  - RemoveClient()
  - SetPlans()
  - GetPlans()
- Report
  - Title
  - Description
  - Date
  - Type
  - CreateReport()
- TasksReport (Report)
  - TotalCompletedTasks
  - WeeklyCompletedTasks
  - TodayCompletedTasks
  - CreateTaskReport()
- ClientsReport (Report)
  - NumberOfClients
  - NumberOfPremiumClients
  - CreateClientReport()
- Pomodoro
  - ShortBreak
  - LongBreak
  - PomodoroLength
  - LongBreakAfter
  - StartPomodoro()
  - Countdown()
- Notification
  - Message
  - ShowNotification(message)

REFERENCES

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