Object-Oriented Programming
Semester 2025-III
Workshop No. 2 — Object-Oriented Implementation

Ronaldo Andres Alvarado Doria – 20251020122
Juliana Valentina Reyes Moreno – 20251020109
David Esteban Ramirez Gordillo - 20242024110

Fecha: 17 de octubre de 2025

PROJECT REPORT
TEND

INTRODUCTION

At present, mental health has become a fundamental priority. Factors such as stress, anxiety, information overload, and lifestyle changes significantly impact people's emotional well-being. However, individuals often lack an accessible, safe, and consistent space where they can reflect on their emotions and monitor their own progress.

The **business purpose** of this application is to provide an empathetic, accessible, and reliable digital tool that promotes emotional self-management and strengthens personal well-being. Its purpose is to offer a personalized experience in which the user can record their moods, habits, and daily thoughts, visualize their progress through graphs and indicators, and receive reminders and recommendations tailored to their emotional health. Through the use of user-centered technology, the goal is to help each person better understand their emotional patterns, set improvement goals, and maintain healthy habits over time.

The main objectives of the project are:

- To promote emotional self-awareness through the daily recording of thoughts and emotions.
- To facilitate the development of healthy habits related to rest, nutrition, and relaxation.
- To offer a simple, secure, and motivating interface that encourages users to interact consistently with the application.

- To provide visual and practical feedback on the evolution of emotional wellbeing.

DEVELOPMENT OF ACTIVITIES

Below are the main activities carried out during this phase of development:

Activity	Description
Conceptual Design Updates	Functional and non-functional requirements were reviewed and adjusted according to the course timeline, available resources, and skill level, selecting the most realistic and relevant ones. The correct structure of user stories was implemented, and updates were made to the *mockups* based on class feedback, improving aspects such as color palette, interface clarity, and visual organization.
Technical Design	CRC cards were used to define class responsibilities and collaborations, which allowed for the creation of a more coherent class diagram. Object-Oriented Programming (OOP) principles were applied to establish proper class relationships, and a sequence diagram was developed to model the interactions between the main objects of the application.
Implementation Plan for OOP Concepts	A plan was outlined to describe how OOP principles will be applied during the implementation stage. In addition, a preliminary project directory structure was proposed to organize system files and modules in a clear and scalable way.

.

1. CONCEPTUAL DESIGN UPDATES

REQUIREMENTS

Functional Requirements

- **User Management**
  - Users can register and log in using email or Google account ○ Users can edit basic profile information (name, age, preferences). *(Only essential functions are kept: registration and basic profile editing.)*
- **Emotional and Habit Tracking** ○ Users can record their daily mood using a scale or emoji. ○ The app shows moods in a calendar view with simple color indicators.
  - Users can track one or two basic habits such as sleep or alimentation. *(Focused on features that can be shown in a prototype or simple interface.)*
- **Personal Journal** ○ Users have a space to write daily reflections or notes.
  - Each entry can include text and emoji (no multimedia for simplicity). *(Simplified by removing attachments or biometric options.)*
- **Reminders**
  - Users can receive reminders to record their mood or write in the journal.
  - Notifications can be enabled or disabled in settings. *(Limited to basic reminders, easy to simulate or design.)*
- **Statistics** ○ The app shows simple charts with the user's emotional progress over time. ○ 5.2 Users can view summaries of their moods and habits. *(Only visual summaries; export options were removed.)*
- ***Suggested Actions*** ○ Provides wellness recommendations such as breathing exercises, meditation, or short walks.
  - Adjusts suggestions based on user mood and behavior patterns.
  - Allows users to accept, postpone, or dismiss recommendations.
- **Settings** ○ Users can delete all their personal data if they decide to reset the app. *(Basic and realistic settings for the project.)*

Non-Functional Requierements

- **Security and Privacy** ○ The app must protect user information with a simple password or PIN.

- Personal notes and emotional records should only be accessible to the logged-in user.
  (Encryption and complex authentication are not included)

- **Usability** ○ The interface must be clear and easy to understand for any user. ○ Main actions (like recording a mood or writing in the journal) should take only a few simple steps. ○ The colors and layout must be calm and visually comfortable. (Focused on simplicity and user comfort instead of full accessibility features.)
- **Reliability** ○ The app should save information automatically or with a visible "Save" button to prevent data loss.
- **Maintainability** ○ The code must be organized and modular, making it easy to understand and update.
  - The app's logic should be divided into clear classes with well-defined responsibilities.
    (Encourages clean code and good practices, matching the goals of an OOP course.)
- **Ethics** ○ The app must not provide clinical advice or simulate a psychologist. ○ It should only offer motivational and reflective content.

USER STORIES

Title: Enable user registration	**Priority: Hight**	**Estimate: 3 points**
USER STORY: As a new user, I want to register an account so that I can manage my progress in a personalized way		
ACCEPTANCE CRITERIA: Given that the user is on the login page, when they enter valid credentials and click the "Login" button, then they are redirected to their dashboard with a welcome message.		

Title: Receive reminders and notifications	**Priority: Medium**	**Estimate: 2 points**
USER STORY: As a registered user, I want to receive notifications and reminders so that I can have a good process		
ACCEPTANCE CRITERIA: Given that the user is on the welcome page, when they click the Notification bell button, then daily notifications will be activated		

Title: Personal journal	**Priority: Hight**	**Estimate: 5 points**
USER STORY: As a user, I want to have a private diary space so that I can reflect and unburden myself		
ACCEPTANCE CRITERIA: Given that the user is logged into their account, when they navigate to the “Personal Journal” section and create a new entry, then the system saves the entry securely, visible only to that user.		

Title: Control of emotions and habits	**Priority: Hight**	**Estimate: 4 points**
USER STORY: As a user, I want to register my mood and habits daily, with the option to use colors, emojis, or scales. So that I can view graphs showing trends over a defined period		
ACCEPTANCE CRITERIA: Given the user has logged in to make the entry, when they click on EMOTIONS, SLEEP and FOOD section and select a mood using colors, emojis, or a scale, then the system saves that entry with the current date.		

Title: Recommendations	**Priority: Low**	**Estimate: 3 points**
USER STORY: As a user, I want to receive recommendations based on my analysis so that I can improve my results.		
ACCEPTANCE CRITERIA: Given that the user has finished registering their day, when they save the information, then the system should provide personalized recommendations based on their analysis.		

MOCKUPS

Improvements were made to the user interface based on the feedback received. One of the main changes was the replacement of the logo with a mascot, aiming to make the application more friendly and approachable for users. In addition, the color palette was adjusted to achieve a clearer and more visually appealing appearance. Some buttons and shapes within the interface were also modified, optimizing their design and layout to enhance the overall user experience.

WELCOME TO TEND

Sign in

Create account

TEND

Email

Password

Sign in

Quererme es cuidarme y
cuidarme es quererme

TEND

Quererme es cuidarme y
cuidarme es quererme

Name

Email

Password

Password

Create account

Hello Juliana

SEPTEMBER

1	2	2	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

2025

September

2
Day

How did it go today?

Today I had a calculus exam :

Emotions

Food

Sleep

September

2
Day

how are you feeling?

Happy | Proud | Angry

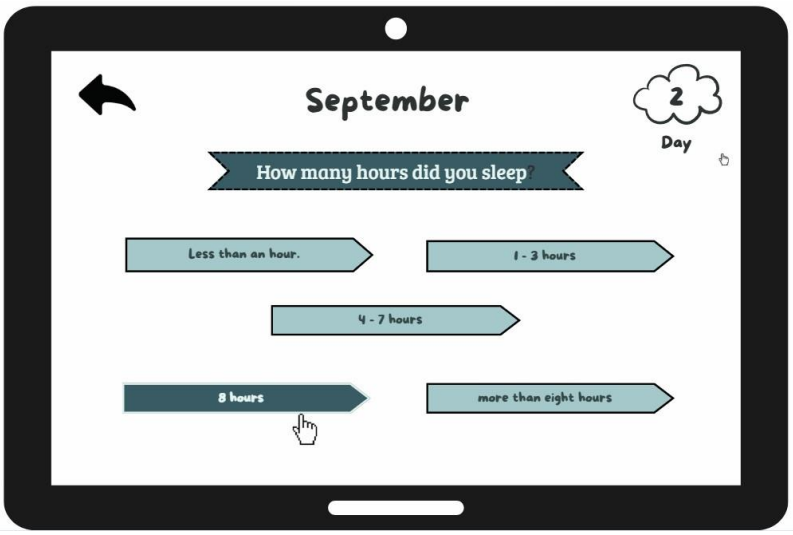
Sad | Hopeful | Worried

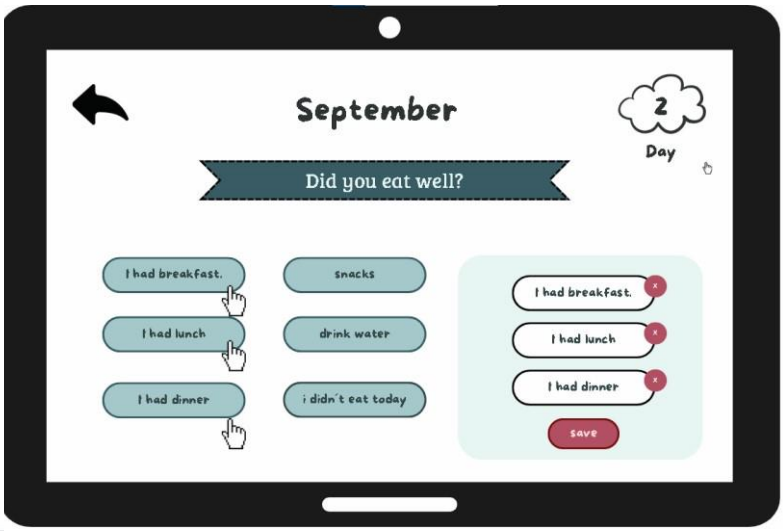
Calm | Relaxed | Lonely

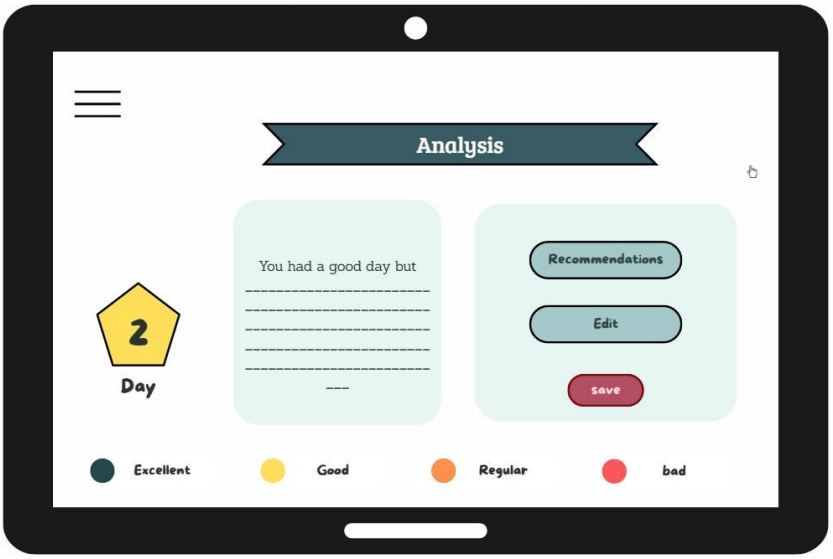
Grateful | Confident | Tired

Worried

"One day at a time,
everything will be fine"


September
2
Day
How many hours did you sleep
Less than an hour.
1 - 3 hours
4 - 7 hours
8 hours
more than eight hours


September
2
Day
Did you eat well?
I had breakfast.
I had lunch
I had dinner
snacks
drink water
i didn´t eat today
I had breakfast.
I had lunch
I had dinner
save


Analysis
2
Day
You had a good day but
Recommendations
Edit
save
Excellent
Good
Regular
bad

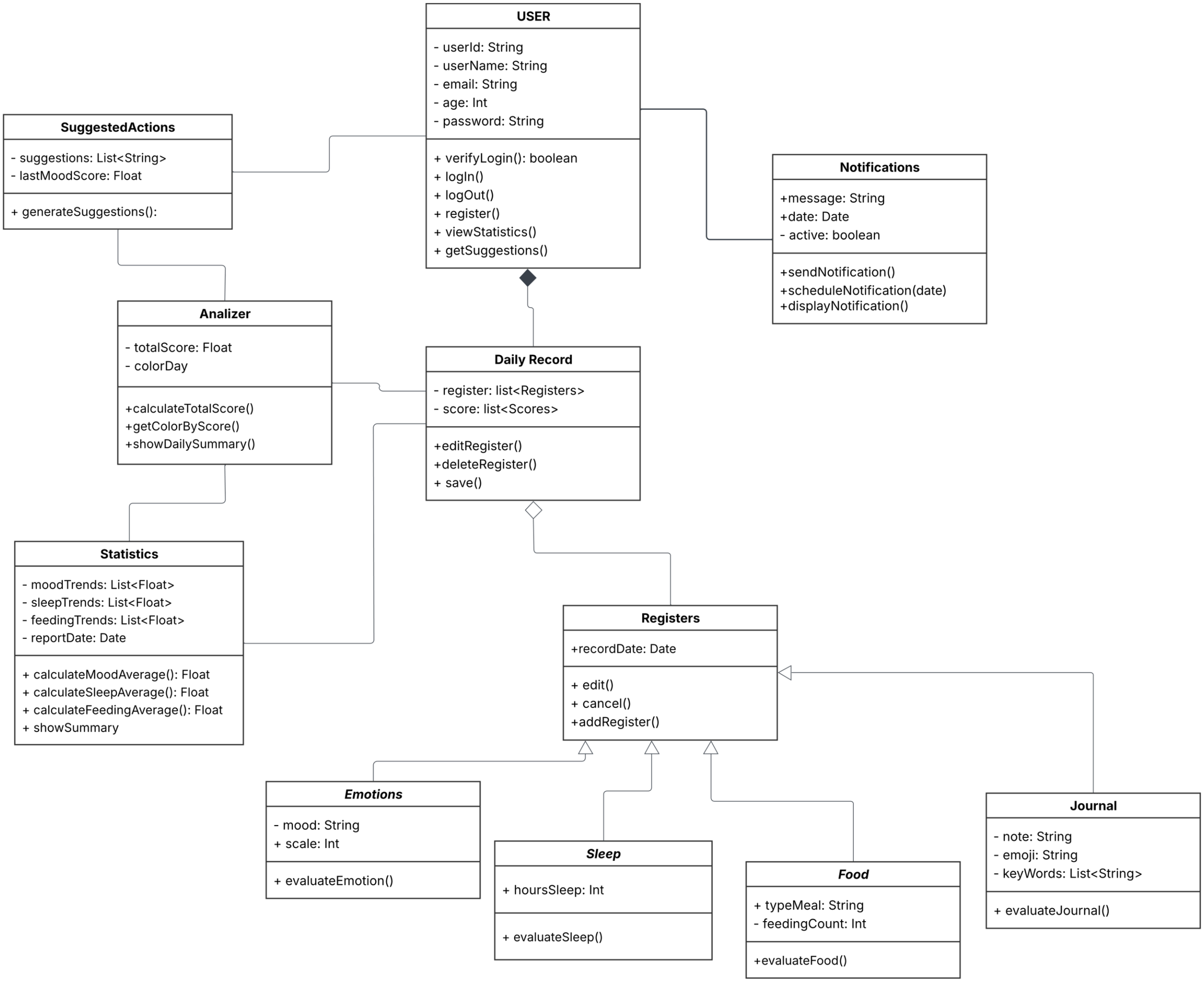
2. TECHNICAL DESIGN (UML DIAGRAMS)

Each class keeps its data protected through private attributes, such as `password` in *User* or `totalScore` in *Analizer*. Access or modification of this information is only performed through public methods (`logIn()`, `calculateTotalScore()`, `evaluateFood()`), ensuring the integrity and security of the data.

Classes like *Analizer* and *Statistics* hide the complexity of data processing behind simple methods such as `calculateTotalScore()` or `showSummary()`. This allows other classes to use the results without knowing the internal logic, simplifying the use of the system and keeping the code cleaner.

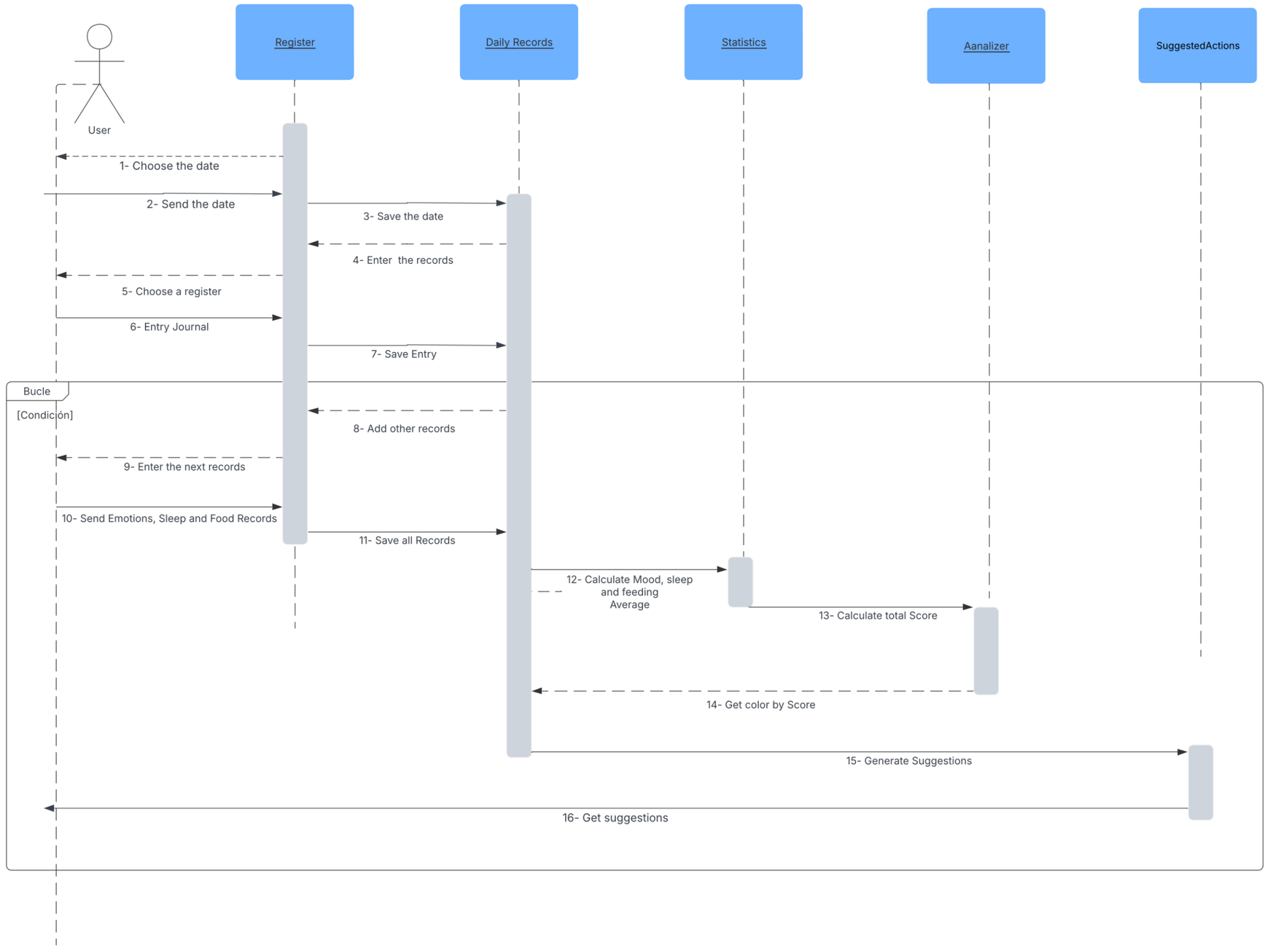
The classes *Emotions*, *Sleep*, *Food*, and *Journal* inherit from the *Registers* class, since they all represent different types of daily records that share common behaviors and attributes, such as the date or the methods `edit()`, `cancel()`, and `addRegister()`. Thanks to this relationship, code duplication is avoided and the system can be easily extended when new types of records are needed, applying the principle of code reuse.

Polymorphism is demonstrated when the child classes (*Emotions*, *Sleep*, *Food*, *Journal*) **override** some of the methods inherited from *Registers* to adapt them to their own functionality. For example, `evaluateEmotion()` analyzes the mood, while `evaluateSleep()` measures sleeping hours. Although the methods share the same structure, each class defines its own behavior, making the system flexible and adaptable.


USER
- userId: String
- userName: String
- email: String
- age: Int
- password: String
+ verifyLogin(): boolean
+ logIn()
+ logOut()
+ register()
+ viewStatistics()
+ getSuggestions()
SuggestedActions
- suggestions: List<String>
- lastMoodScore: Float
+ generateSuggestions():
Notifications
+message: String
+date: Date
- active: boolean
+sendNotification()
+scheduleNotification(date)
+displayNotification()
Analizer
- totalScore: Float
- colorDay
+calculateTotalScore()
+getColorByScore()
+showDailySummary()
Daily Record
- register: list<Registers>
- score: list<Scores>
+editRegister()
+deleteRegister()
+ save()
Statistics
- moodTrends: List<Float>
- sleepTrends: List<Float>
- feedingTrends: List<Float>
- reportDate: Date
+ calculateMoodAverage(): Float
+ calculateSleepAverage(): Float
+ calculateFeedingAverage(): Float
+ showSummary
Registers
+recordDate: Date
+ edit()
+ cancel()
+addRegister()
Emotions
- mood: String
+ scale: Int
+ evaluateEmotion()
Sleep
+ hoursSleep: Int
+ evaluateSleep()
Food
+ typeMeal: String
- feedingCount: Int
+evaluateFood()
Journal
- note: String
- emoji: String
- keyWords: List<String>
+ evaluateJournal()

Diagramas de secuencia

Crea un diagrama de secuencia para hacer un modelo de la lógica de un procedimiento, función u operación sofisticada. Mira un tutorial básico y lee más en este artículo del Centro de Ayuda.



3. IMPLEMENTATION PLAN FOR OPP CONCEPTS

The system applies **encapsulation** by declaring class attributes as *private* (e.g., `password`, `totalScore`) and controlling access through *public getters and setters*. This ensures that sensitive data can only be read or modified through secure, validated methods.

Inheritance is implemented by defining *Registers* as a base class and having *Emotions*, *Sleep*, *Food*, and *Journal* as derived classes. These subclasses share common attributes and behaviors (such as `edit()`, `cancel()`, and `addRegister()`) while adding their own specialized methods for handling different types of daily records.

Polymorphism occurs when the derived classes override methods from the base class to implement specific functionality. For instance, `evaluateEmotion()`, `evaluateSleep()`, and `evaluateFood()` redefine inherited behaviors to perform unique evaluations. This use of method overriding allows the system to treat different record types uniformly while maintaining specialized logic in each subclass.

PRELIMINARY DIRECTORY STRUCTURE

```
mental-health-app/
│
├── src/
│   └── com/
│       └── mentalhealthapp/
│           ├── Main.java
│           │
│           ├── controller/
│           │   ├── UserController.java
│           │   ├── RecordController.java
│           │   ├── JournalController.java
│           │   ├── AnalysisController.java
│           │   └── NotificationController.java
│           │
│           ├── model/
│           │       ├── User.java
│           │       ├── DailyRecord.java
│           │       ├── Register.java
│           │       ├── Emotions.java
│           │       ├── Sleep.java
```

```
│           │       └── Food.java
│           │       ├── Journal.java
│           │       ├── Analyzer.java
│           │       ├── Statistics.java
│           │       ├── SuggestedActions.java
│           │       └── Notification.java
│           │
│           ├── view/
│           │      ├── MainView.java
│           │      ├── UserView.java
│           │      ├── RecordView.java
│           │      ├── JournalView.java
│           │      └── AnalysisView.java
│           │      ├── MainFrame.java
│           │      ├── DashboardPanel.java
│           │      └── RecordPanel.java
└──────────────────────
```

Reflexiones

We can see this project gradually coming to life, as we start to recognize everything our program truly needs and what it probably does not. Throughout this stage, we have noticed significant improvement in our understanding of how the code works and how to make it simpler and more efficient. This progress has been possible thanks to the use of Object-Oriented Programming concepts, which have helped us give structure and clarity to our ideas.

Even though we had some difficulties creating the graphics and diagrams to represent how the code functions, this experience allowed us to better visualize the system and understand the logical connections between each component. We know that this is only the first step before starting to code, but it has been essential for building a solid foundation.

During this process, we also rebranded the application by updating the logo and mascot, as well as replacing the color palette used in the mockups. These changes made the interface more cohesive, friendly, and visually appealing.

Overall, we are taking our first real steps toward creating a meaningful and well-designed program. Our goal is that **TEND** not only works effectively from a technical perspective but also brings a positive and uplifting experience to its users.