Habitra

*An Habit Tracker focused on fun by making it a video game

Fabián Cabrera
Universidad Distrital Francisco Jose de Caldas
Systems Engineering
Bogotá, Colombia
20242020020

Juan Avila
Universidad Distrital Francisco Jose de Caldas
Systems Engineering
Bogotá, Colombia
20242020030

***Abstract*—Habit tracking is a strategy used by a large group of people to keep good and life-changing habits; however, it can quickly become a tedious chore to deal with. Habitra proposes a simple yet elegant solution to this problem: making a habit tracker that rewards its users, by making it just like a video game, granting the user experience, as if in a role-playing game. In this paper, we explain our investigation, how we developed the app, and what were our results.**

***Index Terms*—habit, task, track, reward**

I. Introduction

Habitra is a project that aims to develop an easy to understand and user-friendly habit and task tracker, that rewards the user with in-app rewards, represented by an experience and levels system, similar to a video game of the role playing genre.

Multiple habit trackers have been made over the years, however, only one of them, Habitica [3], rewards the user with ”experience points” in a manner similar to a video game. This generates a necessity for new habit tracking apps that apply this same design.

This design philosophy comes from the fact that rewarding good habits can make them intrinsic to a person’s behavior if applied correctly, however, it should be noted that tracking habits becomes an habit of itself, and stopping to do so can greatly impact other habits [2]. Habitra gives a small but substantial reward that can improve someone’s lifestyle, given they are dedicated to their habits and compromise themselves with tracking their habits.

Books such as Atomic Habits [1] have profusely inquired on how rewarding habits in a small but meaningful way has a profound impact on keeping habits and prevents losing them.

In a similar manner, one is more likely to accomplish a given task or goal if given a deadline to complete it.

On the off-hand, punishing bad habits can greatly provide to losing them, improving a person’s quality of life along with gaining good habits.

Habitra helps the user by reminding them of what their habits and non-accomplished tasks are, while making it rewarding and stimulating.

II. Methods and Materials

Fig. 1. Main menu of the application

A. Design

The application design was made to be reminiscent of a video game, one of the role-playing genre. This gives the application a pleasant and pretty look.

B. Coding

The apps backend was coded in Java, due to its focus on Objects, which makes the different parts of the application easier to differentiate, by having different classes that clearly state what their attributes and behaviors are. This facilitates development and communication within the team. It was also chosen because of its potency and industrial capabilities, making it an ideal programming language for commercial applications.

Each object was coded and designed with object oriented design in mind, using tactics such as SOLID. This gives a clear insight into how the objects should behave, which greatly helps when coding.

C. Rewards System

The rewards part of the application came from the matters treated by books such as Atomic Habits, which explains how rewarding habits can greatly affect whether they are kept or not. This rewards system has also been implemented by another habit tracker called Habitica.

Along with this, habit tracking as itself is a powerful yet simple tool that can help users remember their habits on a daily basis. By having a tracker that is readily available in the users

smart devices, such as their computer, can be significantly beneficial, since they have an almost permanent access to these devices, while tracking on a physical medium can be not only more tiresome, but also has the possibility of being damaged by external means.

D. Background

Other habit trackers were studied in order to identify what made them work and what were the core elements that all of them shared, those being the ability of the user to add new habits, a menu that displays the habits and a checkmark for each of them that indicates if the habit was completed or not.

III. Results

A. Expected Results

We expected that the application would accomplish all of it functional requirements: Adding habits, tasks, and bad habits; being able to see them in the main menu and check them as completed, rewarding, or punishing appropriately.

We also expected it to fulfill the user requirements and needs: Completing habits and tasks feels rewarding; the interface is pleasant to look at and also functional.

B. Design

The app design phase consisted of conceptualizing the idea, identifying and analyzing the different parts of the program and how they could be represented as Java classes and objects; making mockups of the user interface, and different diagrams that illustrate how the classes are related and the flow of the program.

The identified classes were Habits and Tasks, both of which share behaviors, so they were abstracted into a single interface: Item, as seen in the diagram below.

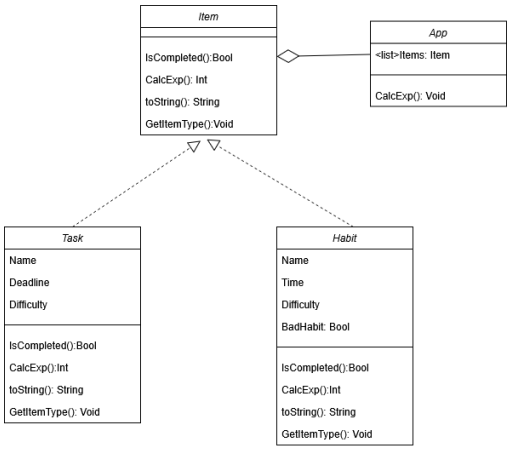

Fig. 2. Class Diagram

The GUI elements of the app were made with Swing, a graphical library for Java that allows for the creation of interactable objects, namely buttons, panes, lists and many other elements.

Regarding requirements and user histories, all these were completed, namely adding habits, tasks and bad habits, being able to see them on the main screen, completing them, and adding or removing experience when they are marked as completed. However, and regrettably, the design of the application couldn't be made exactly as conceptualized in mockups, due to limitations in the Swing library, such as its complicated nature and unnatural ways of adapting animations and other events that require a timer. Either way, this user interface is functional and serves as a medium for all the listed requirements.

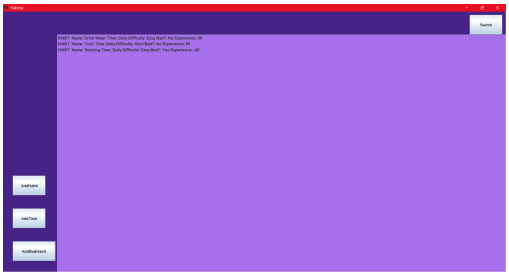

Fig. 3. Current Main menu of Habitra

C. Coding

The first instances of code were simple code snippets that work as the rest of the program placeholders. They depict how object-oriented programming concepts are implemented in the final program.

These code snippets evolved into more complex and robust systems that allowed for a complete and functional application.

The app was developed in a manner that responded to the expected flow of the application, allowing the user to add new habits at any given moment, as seen in the diagram below.

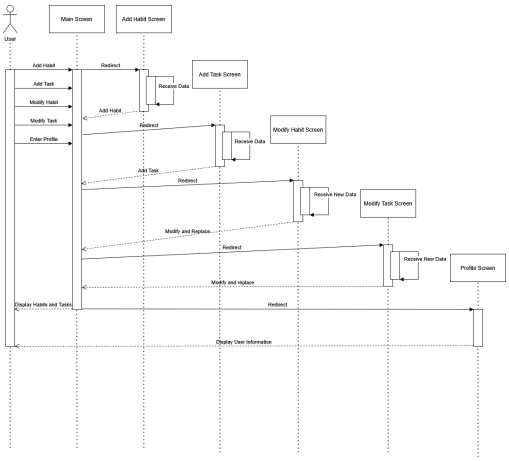

Fig. 4. Sequence Diagram

Ultimately, the application was successfully developed following its expected flow and sequence, having all of its classes coded in an understandable and coherent manner, while the main application was also coded, having all of its events and interactions working correctly.

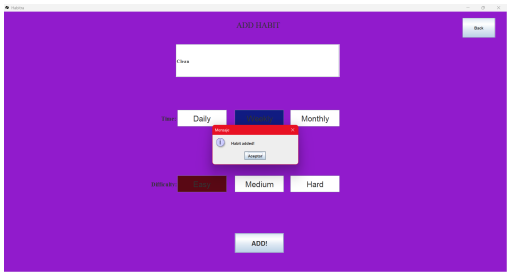

Fig. 5. Example of how adding an habit looks

D. Rewards System

The Rewards System was implemented by granting the user "experience points" when they complete an habit, and punishes them accordingly when they check a bad habit as completed.

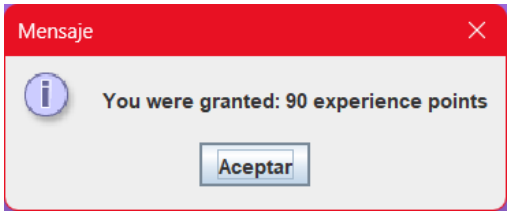

Fig. 6. Popup that indicates how much experience the user gained

It also shows them how much experience they have accumulated so far, and how much they need to level up.

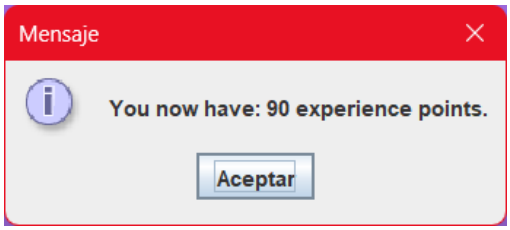

Fig. 7. Accumulated Experience

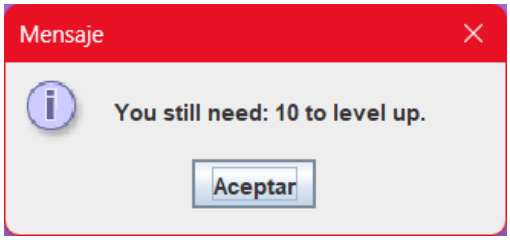

Fig. 8. "To go" experience

When the user has reached or gone past certain "experience threshold", they will "level up".

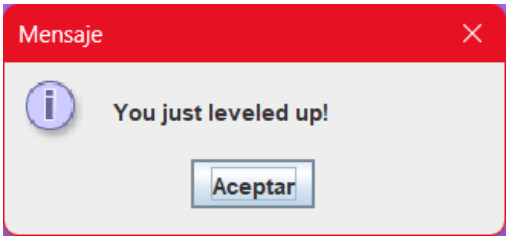

Fig. 9. Message that indicates when a user levels up

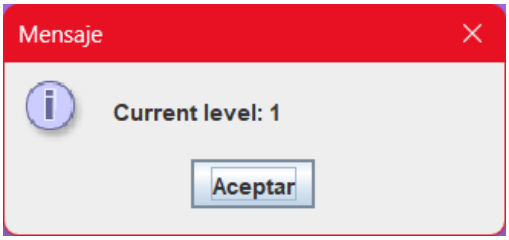

Fig. 10. User current level

This threshold grows exponentially and can be represented by the formula:

$$y = 100x^2 \tag{1}$$

Where "y" indicates the current experience threshold, and x is the users current level.

IV. Limitations

Thought Habitras development, we faced various limitations and roadblocks that forced us to change our approach while developing, especially regarding the visual aspect of Habitra.

Due to the Swing library intricacies and form of use, we were forced to take a different approach when adding the visual and interactable elements of the user interface.

Persistence in the application is achieved by writing and reading from text files. This generates problems with data integrity, since files can be easily modifiable. It can also cause problems with consumed space in disk, given enough data, it can grow until it has a considerable file size, despite text files usually being only a few kilobytes large.

Conclusions

Habitra is an application that successfully helps users add habits and tasks, keeping track on them, and giving a small but notable reward in the form of the experience that would be granted in a role-playing video game.

A. Future work

Habitra is a fully functional app as it stands, however, it has many areas in which it can grow and improve, such as:

1) GUI Refinement: Improving the GUI overall looks and responsiveness
2) New Features: Add new interactive elements such as habit streaks, deadline alerts, and additional reward types, such as customization.

3) Data Persistence: Allow Habitra to hold data in a robust and scalable data base.
4) Social Features: Allow users to interact and help each other.

These possible improvements aim to transform Habitra from simply functional, to a complete application that can aim to have a long future usage and development.

REFERENCES

[1] James clear, ”Atomic Habits: An Easy and Proven Way to Build Good Habits and Break Bad Ones”, 2018.
[2] Ian Renfree et al, ”Don’t Kick the Habit: The Role of Dependency in Habit Formation Apps”, 2016.
[3] Habitica (2024), ‘Habitica: Gamify your life’, https://habitica.com. Available at: https: //habitica.com.