



H

A

B

I

T

R

A

TAKE CONTROL OF YOUR HABITS

Fabian Cabrera
Juan Avila

WHAT IS HABITRA?

Habitra is a project that aims to make a Habit Tracker to help accomplish goals, keep tabs on all of your good habits, and help getting rid of bad habits, all of this while making it fun, resembling a kind of videogame. The app is primarily intended for mobile devices.

OUR PURPOSE ?

Habitra aims to make an interactive Habit Tracker that

- Keep the habits in check

- Rewards completing said habits

- fulfill your tasks

The app keeps interactive by the use of this tools

DEADLINE :

It allows to put a deadline to the tasks

D . L I N E :

DD/MM/YYYY

DIFFICULTY : It allows to put a difficulty to the habits and tasks

D I F F :

EASY

MEDIUM

HARD

EXPERIENCE :

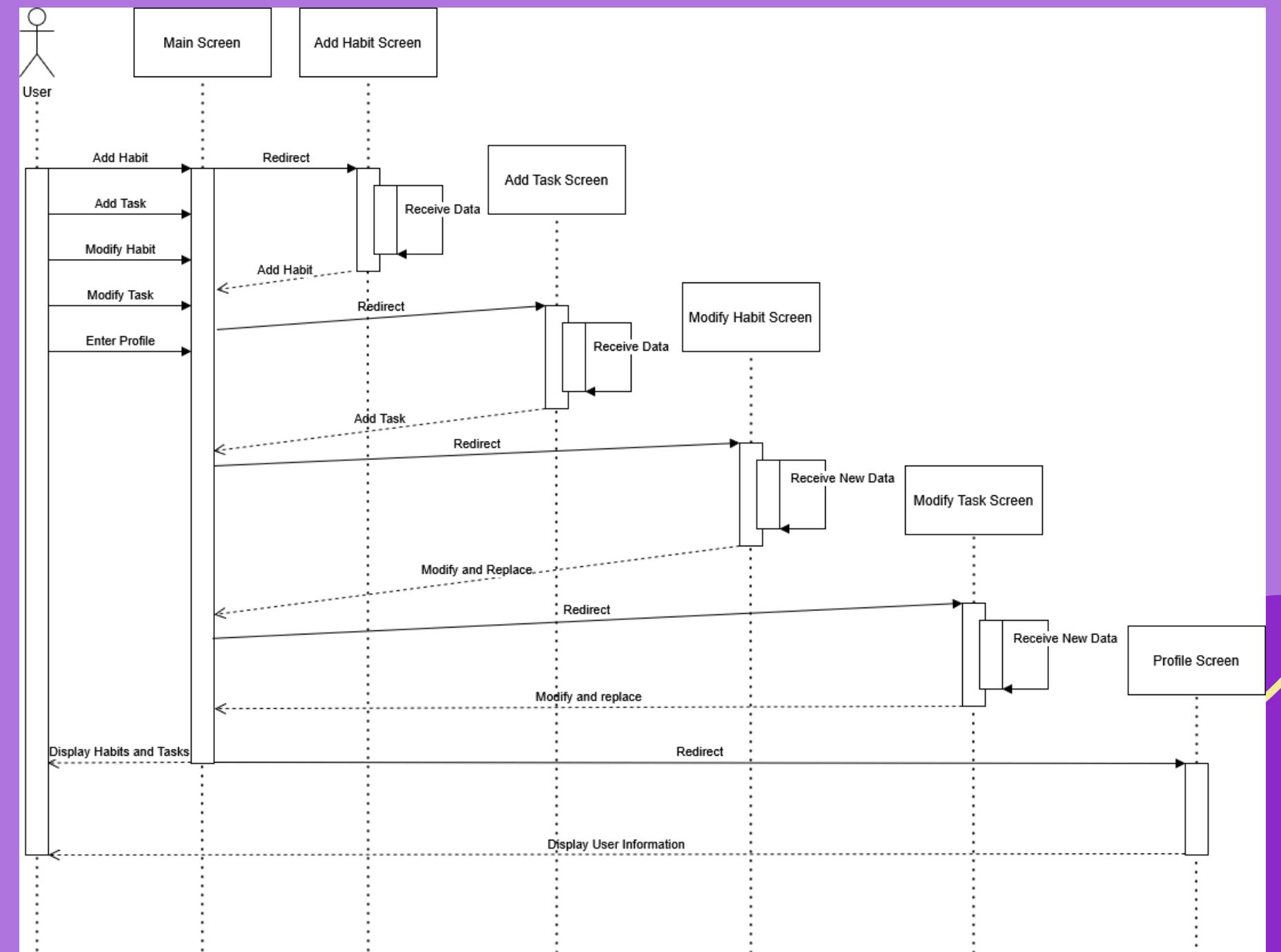
It shows you the experience gained by complete tasks and habits.

By the using of crc cards and the uml diagrams we could guide to create the classes and the mockups

MAKING USE

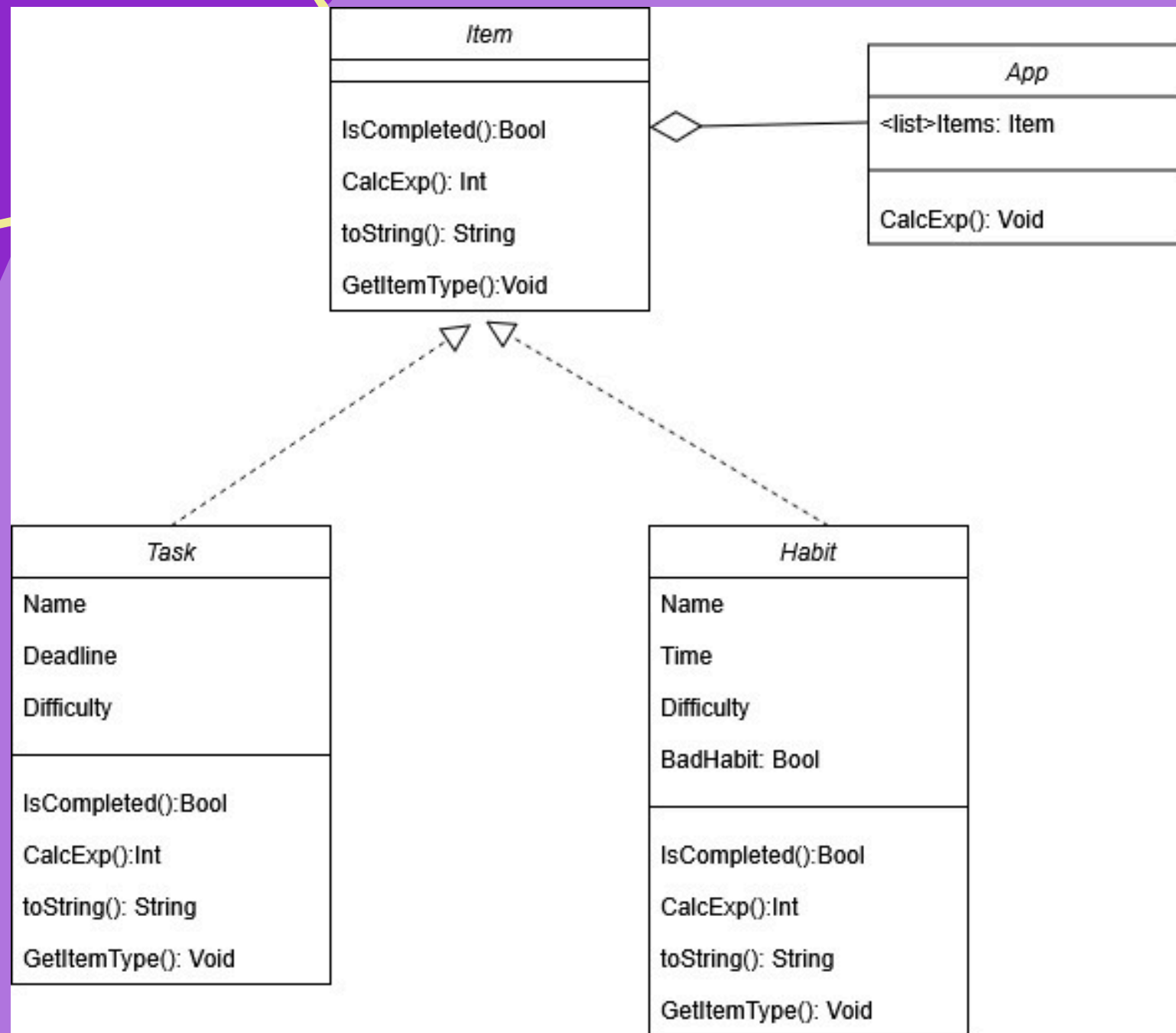
OF PRINCIPLES LIKE:

- INHERITANCE
- ABSTRACTION



Sequence diagram showing all the operation of the app

CLASS DIAGRAM



CRC CARDS

Task

Responsibilities:

- Be completed
- Give experience when is completed
- Be modified
- Be expired

Colaborators:

- The user
- Habit Class

Habit

Responsibilities:

- Be completed
- Give experience when is completed
- Be modified
- Be updated

Colaborators:

- The user
- Task Class

MOCKUPS



MAIN SCREEN OF HABITRA

NEW HABIT SCREEN



IMPLEMENTATION

In this project was useful the implementation of some of the OOP and SOLID principles by help to structure all the code.

POLYMORPHISM
INHERITANCE

ABSTRACTION



```
1 public abstract void CalcExp();
```



```
1  /**Calculates experience */  
2  public void CalcExp(){  
3      this.exp= 100*(diff);
```


IMPLEMENTATION

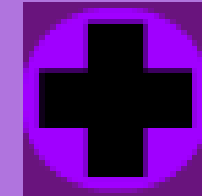
INTERFACE SEGREGATION

```
1  /**This is an interface for Tasks and habits */
2
3  public interface Item{
4      public abstract Boolean IsCompleted();
5      public abstract void CalcExp();
6      public abstract String toString(); //Override of toString Java method
7      public abstract String getName();
8      public abstract Integer getExp();
9  }
```

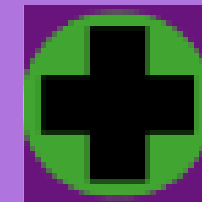
```
1  /**This is a class for Habits */
2  public class Habit implements Item{
3      private String habName;
4      private int time;
5      private int exp;
6      private int diff;
7      private Boolean status;
8      private Boolean isBad;
9
10     /**Constructor for habits
11      * @param name: The name of the habit
12      * @param time: Periodicity of the habit
13      * @param difficulty: The difficulty of the habit
14      * @param status: If the habit is completed or not
15      * @param bad: Wheter the habit is flagged as bad or not
16      */
17     public Habit(String name, int time, int difficulty, boolean status, Boolean bad){
18         this.habName= name;
19         this.time= time;
20         this.diff= difficulty;
21         this.status= status;
22         this.isBad= bad;
23     }
```

LSKOV SUBSTITUTION

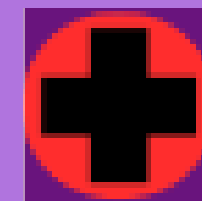
MAIN SCREEN



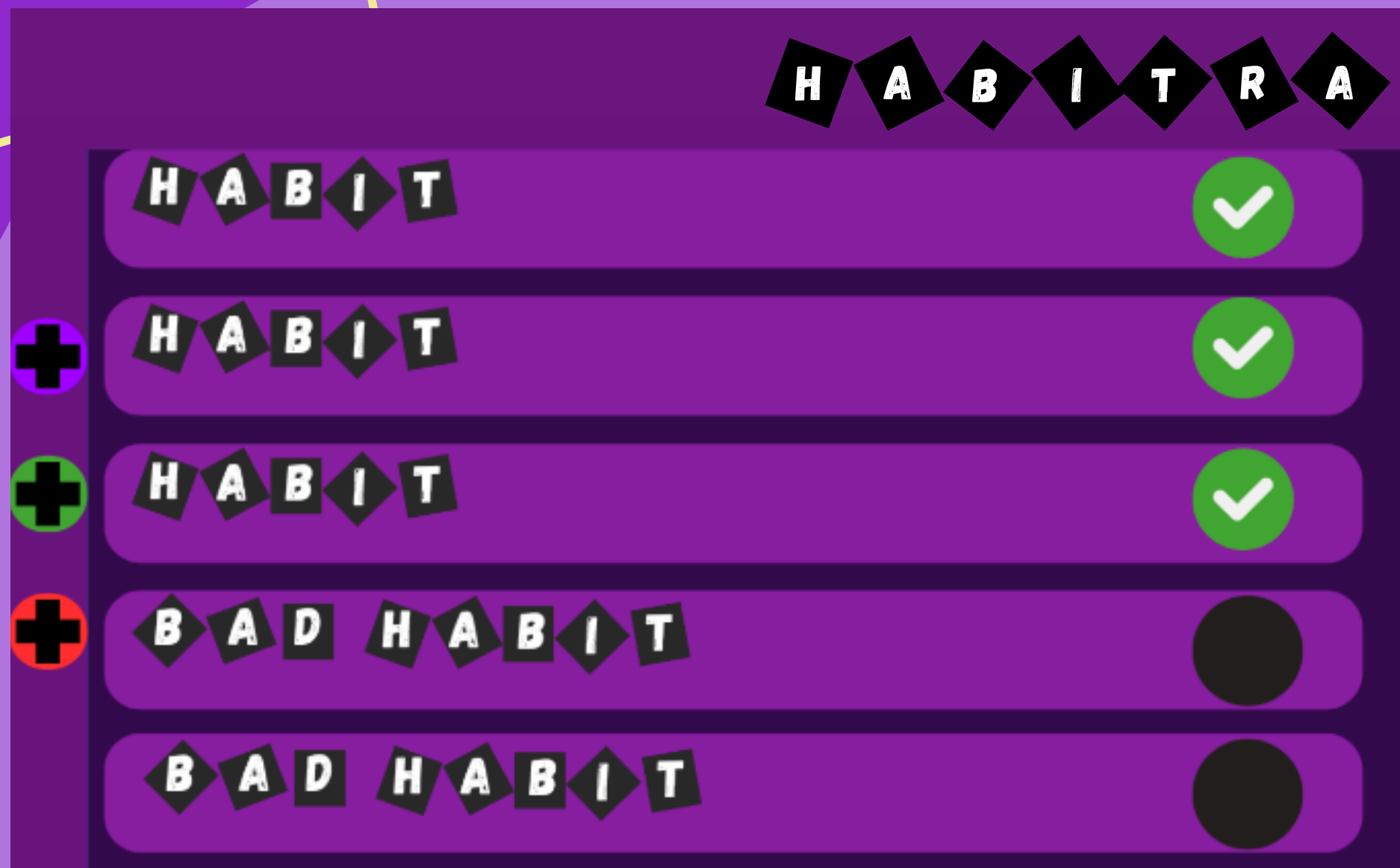
This button send you to another module in wich you can add a new habit.



This button send you to another module in which you can add a new bad habit



This button send you to another module in which you can add a task .

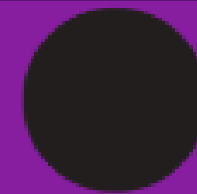


DONE

H A B I T



B A D H A B I T



NOT DONE

In this screen is shown the habits , bad habits and tasks that the user have saved .And they can be checked if the user has already made the task or habit.

ADD HABITS AND TASKS

N E W

N A M E :

T I M E :

D I F F :

A D D !

ADD HABITS AND BAD HABITS

ADD TASKS

T A S K

N A M E :

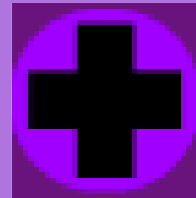
D . L I N E :

D I F F :

A D D !

ADD HABITS AND TASKS

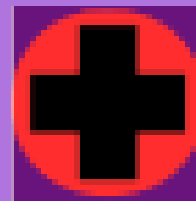
After using one of the three plus buttons on the main screen they will lead you to one of these screens



**ADD HABITS AND BAD
HABITS**



ADD TASKS



ADD HABITS AND TASKS

These screens are similar , they have the space where put the name of the habit/task and the difficulty. The difference between these two is the time that will be confirmed (daily,weekly,monthly) in the habits and bad habits and the deadline in the tasks

HABITS

TASKS

T I M E :

DAILY

WEEKLY

MONTHLY

D . L I N E :

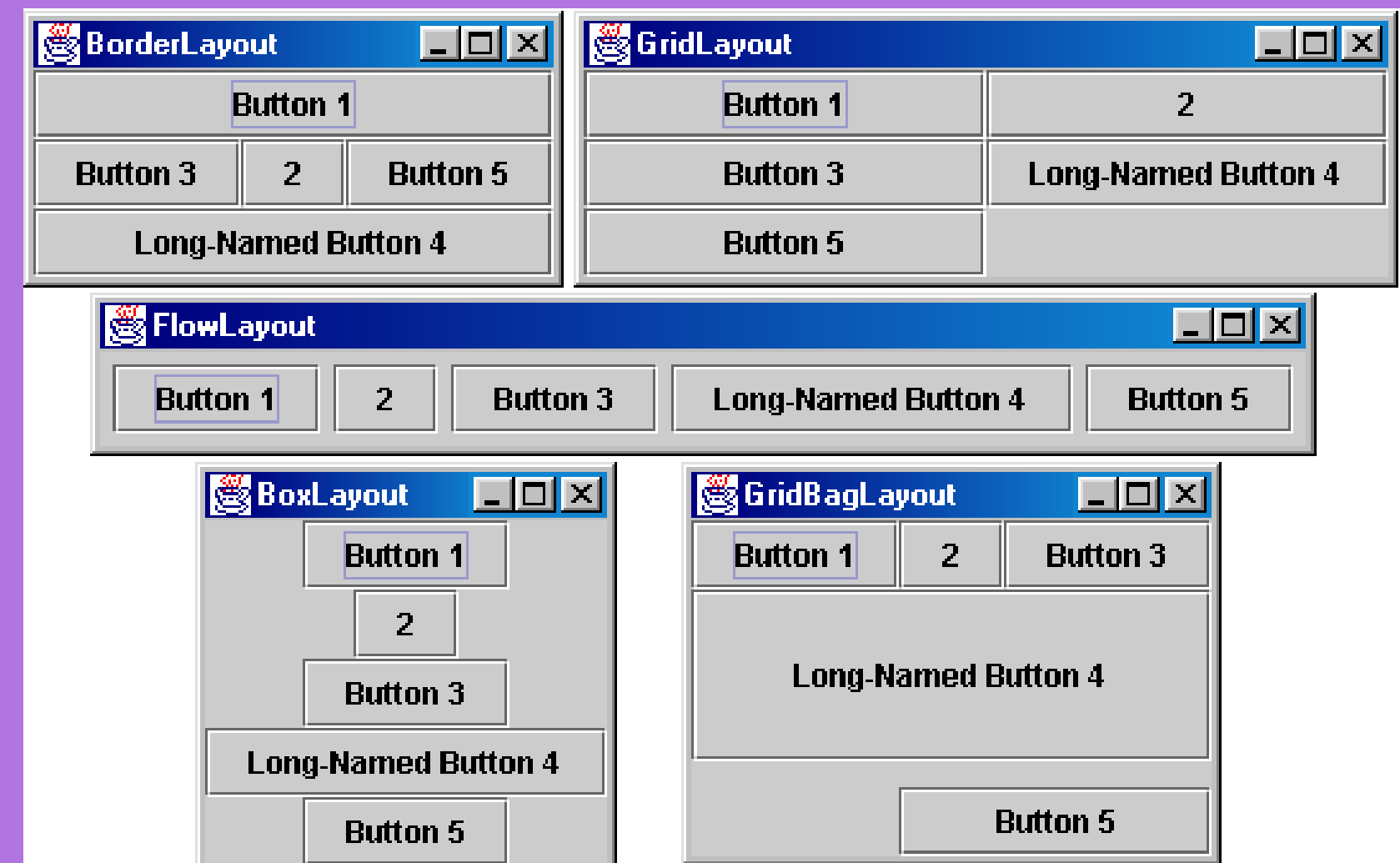
GUI

The GUI (graphical user interface) is one of the most important parts because it is what the user sees and interacts with.

In our case the tool we use to make this was Java swing supporting the MVC by separate the different functions when it comes to making an app

GUI

Java swing helps to create a simple but functional interface by the using of its own tools like JButtons JFrames JPanels and some other elements that ease the creation of what we expected with mockups.



GUI

ADD BAD HABIT

Back

Name :

Time:

Daily

Weekly

Monthly

Difficulty:

Easy

Medium

Hard

ADD!

ADD BAD HABITS

ADD HABIT

Back

Name :

Time:

Daily

Weekly

Monthly

Difficulty:

Easy

Medium


Hard

ADD!

ADD HABITS

RESULTS

- Fully implemented core logic in Java, using OOP and SOLID principles.
- Built a functional and user-friendly graphical interface based on initial mockups.
- All user stories were fulfilled, ensuring alignment between design and implementation.
- Achieved a modular and extensible architecture, ready for future features like data persistence and multi-user support.



T H A N K S !

Clear, J. (2018), Atomic Habits: An Easy and Proven Way to Build Good Habits and Break Bad Ones, Avery.

Habitica (2024), 'Habitica: Gamify your life', <https://habitica.com>. Available at: <https://habitica.com>.

Renfree, I., H. D. M. P. S. K. and Cox, A. (2016), Don't kick the habit: The role of dependency in habit formation apps, in 'Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems', ACM, San Jose, CA, USA, pp. 3222–3233.

<https://users.dcc.uchile.cl/~lmateu/CC60H/Trabajos/edavis/swing.html>