Imagen que contiene Gráfico

Descripción generada automáticamente

**Content**

[1. INTRODUCTION. 3](#_Toc147251019)

[1.1. Components and functionalities (SGTR) 3](#_Toc147251020)

[1.1.1. Task storage and reminders 3](#_Toc147251021)

[1.1.2. User Interface 3](#_Toc147251022)

[1.1.3. Priority Management 4](#_Toc147251023)

[1.1.4. Undo function 4](#_Toc147251024)

[2. REQUIREMENTS ANALYSIS. 5](#_Toc147251025)

[1.2. Case study: 5](#_Toc147251026)

[2.1.1. Functional Requirement N°1 6](#_Toc147251027)

[2.1.2. Functional Requirement N°2 7](#_Toc147251028)

[2.1.3. Functional Requirement N°3 8](#_Toc147251029)

[2.1.4. Functional Requirement N°4 9](#_Toc147251030)

[2.1.1. Functional Requirement N°5 10](#_Toc147251031)

[2.1.2. Functional Requirement N°6 11](#_Toc147251032)

# INTRODUCTION.

Efficient task and reminder management is essential for personal and professional productivity and success. To address this need, the development of a task and reminder management system has been proposed that will allow users to effectively add, organize and manage their daily to-dos.

This project aims to design and implement a comprehensive system that addresses all dimensions of task management, from the storage of information to the presentation of an intuitive user interface and advanced functionalities. The system will be based on efficient algorithms and data structures to ensure optimal performance.

## Components and functionalities (SGTR)

### Task storage and reminders

We will use a hash table to store tasks and reminders. Each entry will have a unique identifier as a key and the task/reminder information as a value. This information includes the title, description, due date, and priority.

### User Interface

We will design a user interface that allows users to add, modify, and delete tasks and reminders. Users will be able to see a list of all tasks and reminders, sorted by due date or priority.

### Priority Management

There will be two categories of tasks: "priority" and "non-priority".

* For priority tasks, we will use a priority queue to organize them according to their importance. When a user adds a new priority task, it will be added to the queue according to its importance, which will ensure that important tasks are handled first.
* Non-priority tasks are handled on a first-in-first-out (**FIFO**) basis.

### Undo function

We will implement a method to undo the actions performed by a user in the system. We will use a stack (**LIFO**) to keep track of the actions performed. The general process for the "Undo" function will be:

1. Create an action stack to keep track of the user's actions.
2. Record each action performed by the user in the stack, including details of the action and the task affected.
3. Implement a last action undo method that unstacks the last action and undoes the corresponding action based on the information stored in the stack.
4. Provide the user interface with an "Undo" option to undo the last action performed.

With these features, our task and reminder management system will be efficient and allow users to organize and manage their tasks effectively.

# REQUIREMENTS ANALYSIS.

## Case study:

Tabla

Descripción generada automáticamente con confianza media

### Functional Requirement N°1

Tabla

Descripción generada automáticamente

### Functional Requirement N°2

Tabla

Descripción generada automáticamente

### Functional Requirement N°3

Tabla

Descripción generada automáticamente

### Functional Requirement N°4

Tabla

Descripción generada automáticamente

### Functional Requirement N°5

Tabla

Descripción generada automáticamente

### Functional Requirement N°6

Tabla

Descripción generada automáticamente