Manual del Usuario

Elaboración:

Juan Pablo Florez Rubio

Nicolás Calderón

Asignatura:

Programación Orientada a Objetos

Docente:

Néstor German Bolívar Pulgarín

Universidad Nacional de Colombia

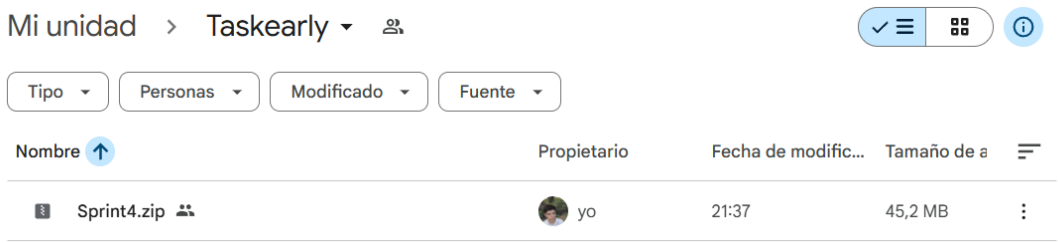
2025-2

Bogotá D.C.

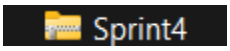
INSTALACIÓN DE LA APLICACIÓN

1. Abrir el drive donde se encuentran la carpeta de archivos comprimida en archivo .zip. mediante el siguiente link:

   https://drive.google.com/drive/folders/1vgH19rKqsBW2hAeeXnAJjKBmOunOwqrn?usp=sharing

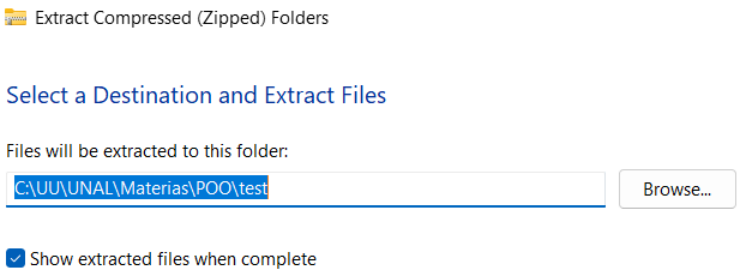


2. Descargar en la biblioteca del computador, en descargas debería aparecer el .zip

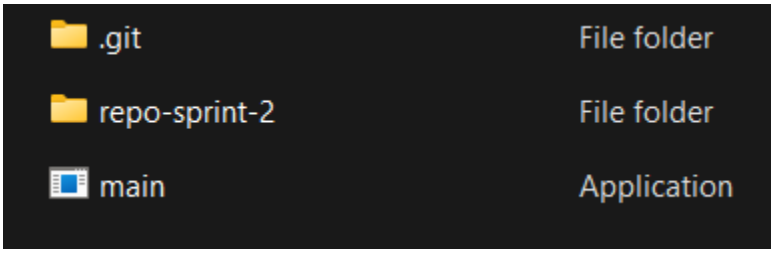


3. Descomprimir el archivo .zip con el extractor.

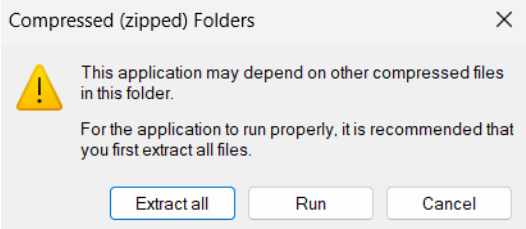
4. Seleccionar la localización deseada



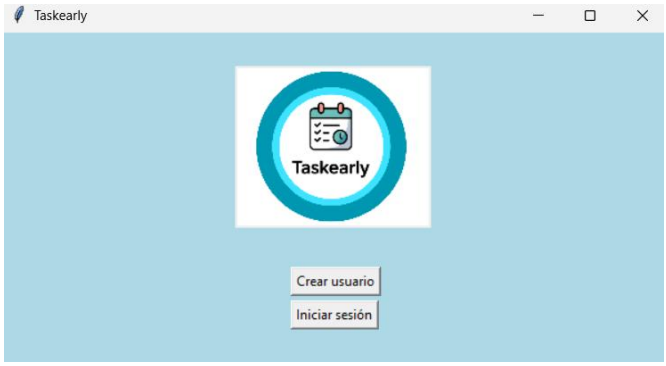
5. Abrir la carpeta sprint3 y asegurarse de estas características:



6. En caso de pedir más descompresiones, permitirlo en “Extract all” y seleccionar la carpeta sprint3:



7. Ejecutar la aplicación “main” dentro de sprint3 dándole doble clic:



Actividades dentro de la carpeta “Model”

1. Se descargan las librerías que conectan con la base de datos Firebase

```
from typing import Optional, Dict, Any
from firebase.config_firebase import FirebaseConfig
from firebase.tarea import Tarea
import firebase_admin
from firebase_admin import db, credentials
from view import view_consola
from model import repo_usuario
```

2. Se crea la clase TareaRepository, la cual guarda todas las operaciones y estructura de las tareas en la base de datos:

```
class TareaRepository:
    #repo para operaciones de Tarea en Firebase
    def __init__(self):
        self.config = FirebaseConfig.get_instance()
        self.config.initialize()
        self.usuarios_ref = self.config.get_reference("Usuarios")
```

3. Los métodos de la clase que refieren al manejo de tareas incluyen: Agregar, leer, actualizar nombre o datos, eliminar tareas, marcar tareas completas y una operación de sistema de recompensa en relación con las tareas realizadas.

Actividades dentro de la carpeta “ViewModel”

1. Se importan las librerías y modelos para realizar acciones y funciones con respecto a modelos:

```
from typing import Optional, Dict
from firebase.tarea import Tarea
from model.repo_tarea import TareaRepository
```

2. Se inicia la clase TareaViewModel y UsuarioViewModel en dos respectivos archivos, los cuales contiene las operaciones y estructuras iniciales que relacionan interacciones del usuario con operaciones con la base de datos y funciones importantes según se requiera manipular tareas de cada usuario o el usuario en sí:

```
class TareaViewModel:
    #viewmodel para gestión de tareas
    def __init__(self):
        self.repository = TareaRepository()
```

```
class UsuarioViewModel:
    #viewModel para gestión de usuarios
    def __init__(self):
        self.repository = UsuarioRepository()
        self.usuario_actual: Optional[str] = None
```

Actividades dentro de la carpeta “View”

1. Contiene dos imágenes y el archivo que se especializa en las vistas y habilidades que obtiene el usuario:

1. El archivo contiene las librerías y referencias con archivos de model y viewmodel que facilitan la comunicación en todo el programa con el usuario:

```
from typing import Optional, Dict
from viewmodel.usuario_viewmodel import UsuarioViewModel
from viewmodel.tarea_viewmodel import TareaViewModel
import tkinter as tk
from tkinter import messagebox
from tkinter import ttk
from model import repo_usuario
from model import repo_tarea
import firebase_admin
from firebase_admin import db, credentials
from firebase.config_firebase import FirebaseConfig
```

2. Contiene la clase padre App, la cual hereda características comunes y gestiona el flujo de frames que en realidad son clases herederas:

```
class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Taskearly")
        self.geometry("600x300")
```

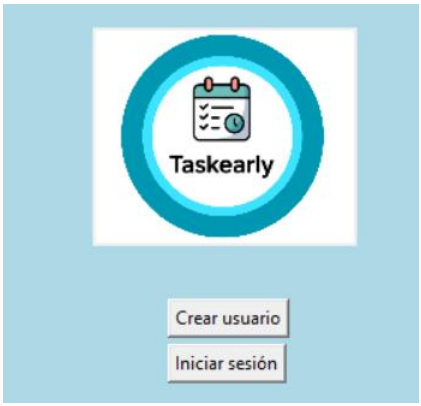
```
# Inicializar pantallas
for F in (Mainview, Loginview, Crearview, Homevi
    nombre = F.__name__
    frame = F(container, self)
    self.frames[nombre] = frame
    frame.place(relwidth=1, relheight=1)
```

3. Contiene clases secundarias a la clase App que representan diferentes frames de la aplicación y entre botones se comunican frames a través de la clase App:

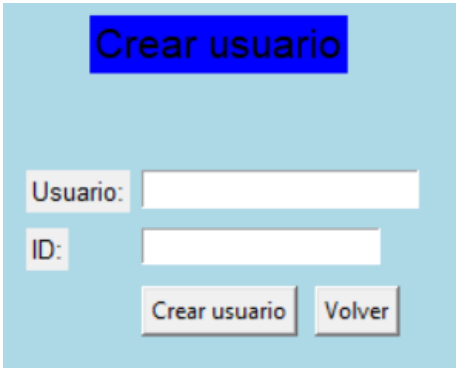
```
class Mainview(tk.Frame):
    def __init__(self, parent, controller):
        super().__init__(parent, bg="lightblue")
```

3.1. Las funciones que cumple cada clase se definen en:

- Mainview: Permite el inicio de sesión o creación de un usuario nuevo, de aquí se derivan los frames Crearview y Loginview:

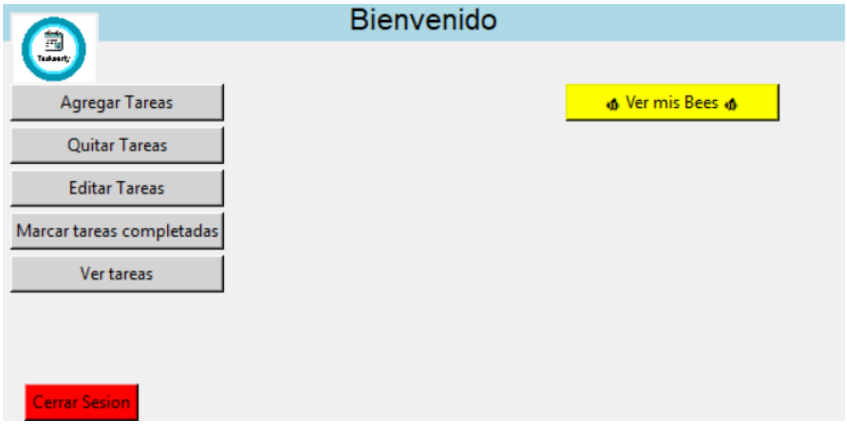


- Crearview: Abre un frame para crear un nuevo usuario en la base de datos:



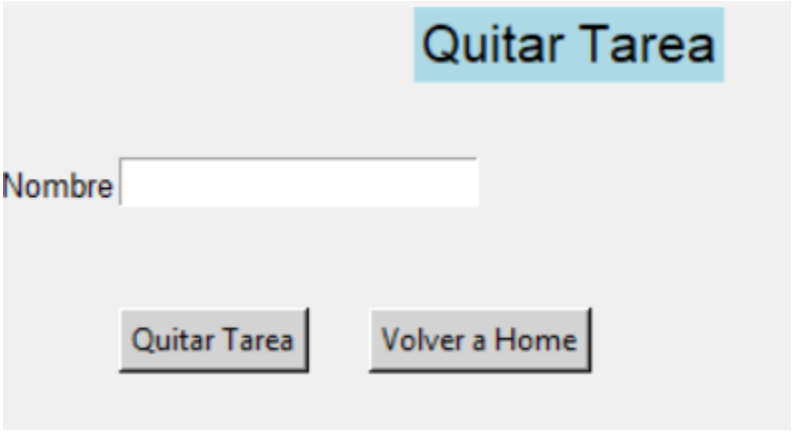
- Loginview: Abre un frame para ingresar en la cuenta de un usuario ya existente, comprobando sus datos:

- Homeview: Es el frame principal para el usuario típico de la aplicación, de aquí se derivan acciones principales dentro del objetivo principal de la aplicación:



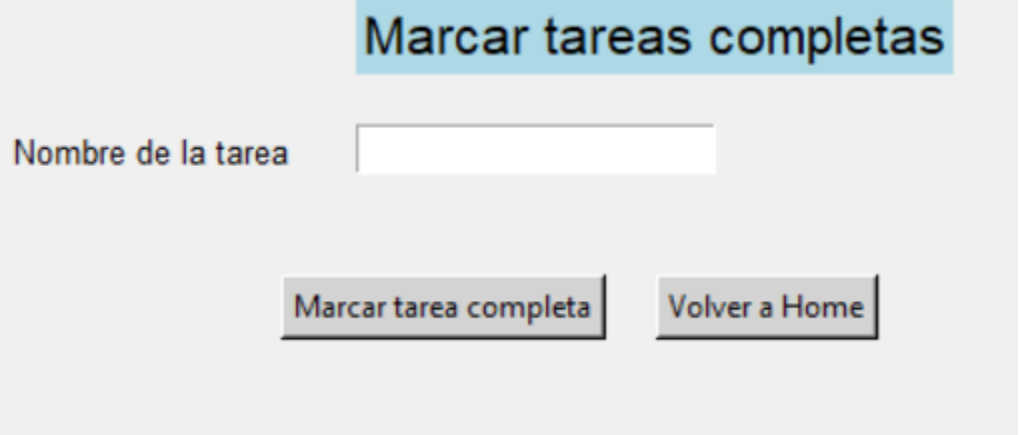
- Agregarview: Permite agregar nuevas tareas relacionadas al usuario:

- Quitarview: Permite eliminar tareas ya existentes (No significa haberlas completado):

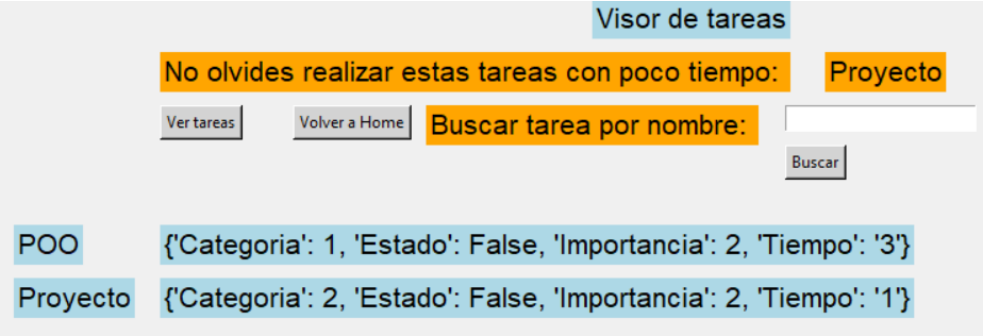


- Editarview: Permite editar tareas ya existentes y relacionadas al usuario:

- Marcarcompletasview: Permite marcar una tarea como completa:



- Vertareasview: Permite ver tareas pendientes, incluye recordatorio para tareas con menor tiempo y buscar tareas por nombre:



- Beesview: Funciona como incentivo y característica interesante de la aplicación: