

Manual del Usuario

Elaboración:

Juan Pablo Florez Rubio

Nicolás Calderón

Asignatura:

Programación Orientada a Objetos

Docente:

Néstor German Bolívar Pulgarín

Universidad Nacional de Colombia

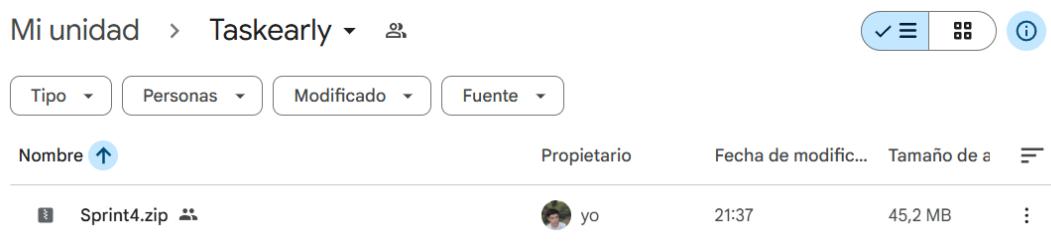
2025-2

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>



A screenshot of a Google Drive folder named "Taskearly". Inside the folder, there is a single item: a file named "Sprint4.zip". The file has a size of 45,2 MB and was modified at 21:37 by the user "yo". The interface includes standard Google Drive filters for Tipo, Personas, Modificado, and Fuente.

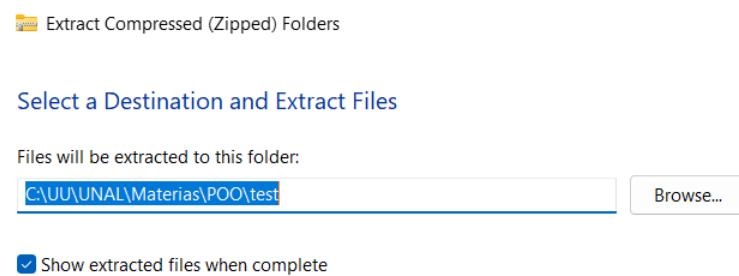
2. Descargar en la biblioteca del computador, en descargas debería aparecer el .zip



3. Descomprimir el archivo .zip con el extractor.

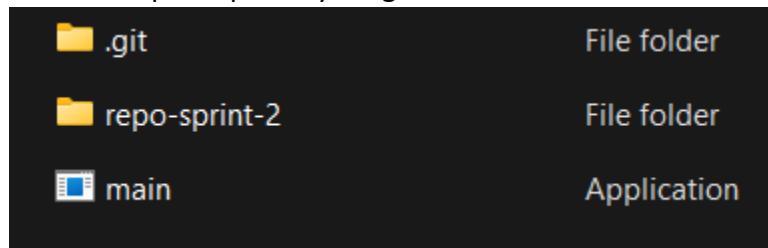


4. Seleccionar la localización deseada

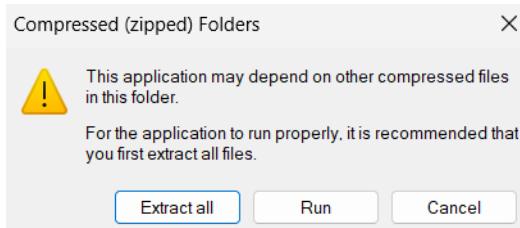


A screenshot of a file extraction dialog box. It shows the option "Extract Compressed (Zipped) Folders" and a "Select a Destination and Extract Files" section. The destination path is set to "C:\UU\UNAL\Materias\POO\test". There is also a checked checkbox for "Show extracted files when complete".

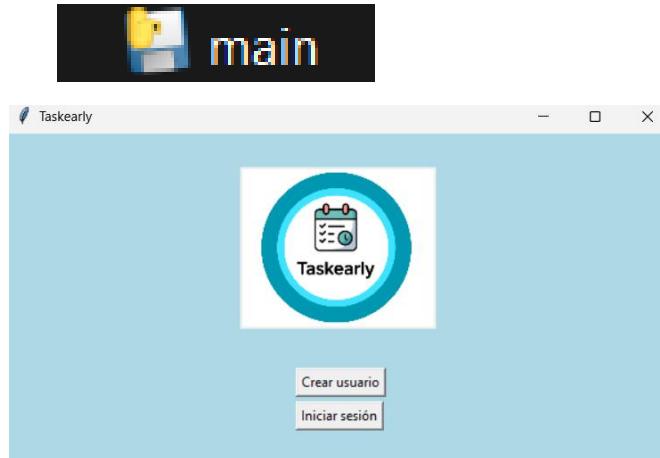
5. Abrir la carpeta sprint3 y asegurarse de estas características:



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, Homeview):
    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:



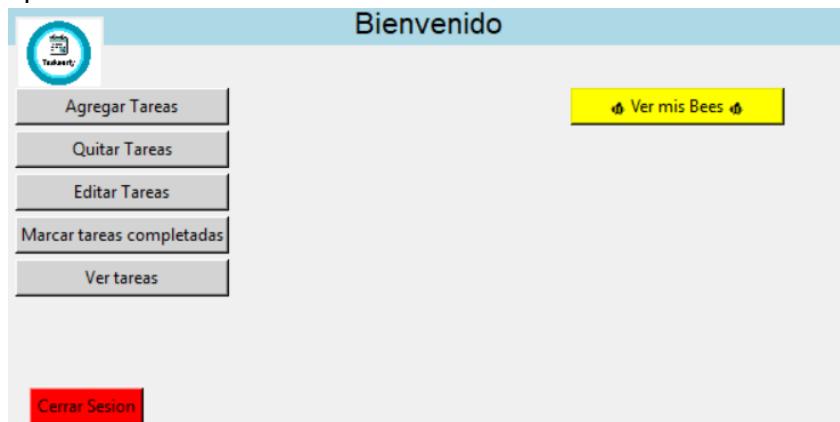
- Crearview: Abre un frame para crear un nuevo usuario en la base de datos:

The image shows a light blue rectangular frame with a dark blue header bar at the top containing the white text "Crear usuario". Below the header are two input fields: one labeled "Usuario:" with a text input field next to it, and another labeled "ID:" with a text input field next to it. At the bottom of the frame are two rectangular buttons: "Crear usuario" on the left and "Volver" on the right.

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

The image shows a light blue rectangular frame with a dark blue header bar at the top containing the white text "Iniciar sesión". Below the header are two input fields: one labeled "Usuario" with a text input field next to it, and another labeled "ID" with a text input field next to it. At the bottom of the frame are two rectangular buttons: "Iniciar sesión" on the left and "Volver" on the right.

- 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:

The screenshot shows a form titled "Agregar Tarea". It has three sections: "Importancia" with radio buttons for "Baja", "Media", and "Alta"; "Nombre:" with an input field; "Tiempo(Dias):" with an input field; "Categoria" with radio buttons for "Estudio", "Trabajo", and "Personal". At the bottom are two buttons: "Agregar Tarea" and "Volver a Home".

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

The screenshot shows a form titled "Quitar Tarea". It has a single input field labeled "Nombre". At the bottom are two buttons: "Quitar Tarea" and "Volver a Home".

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

Editar Tarea

Nombre de la tarea:

Editar Nombre
 Editar Tiempo
 Editar Importancia
 Editar Categoría

Editar Tarea **Volver a Home**

- Marcarcompletasview: Permite marcar una tarea como completa:

Marcar tareas completas

Nombre de la tarea

Marcar tarea completa **Volver a Home**

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

Visor de tareas

No olvides realizar estas tareas con poco tiempo: **Proyecto**

Ver tareas **Volver a Home** Buscar tarea por nombre:

POO {'Categoria': 1, 'Estado': False, 'Importancia': 2, 'Tiempo': '3'}
Proyecto {'Categoria': 2, 'Estado': False, 'Importancia': 2, 'Tiempo': '1'}

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

