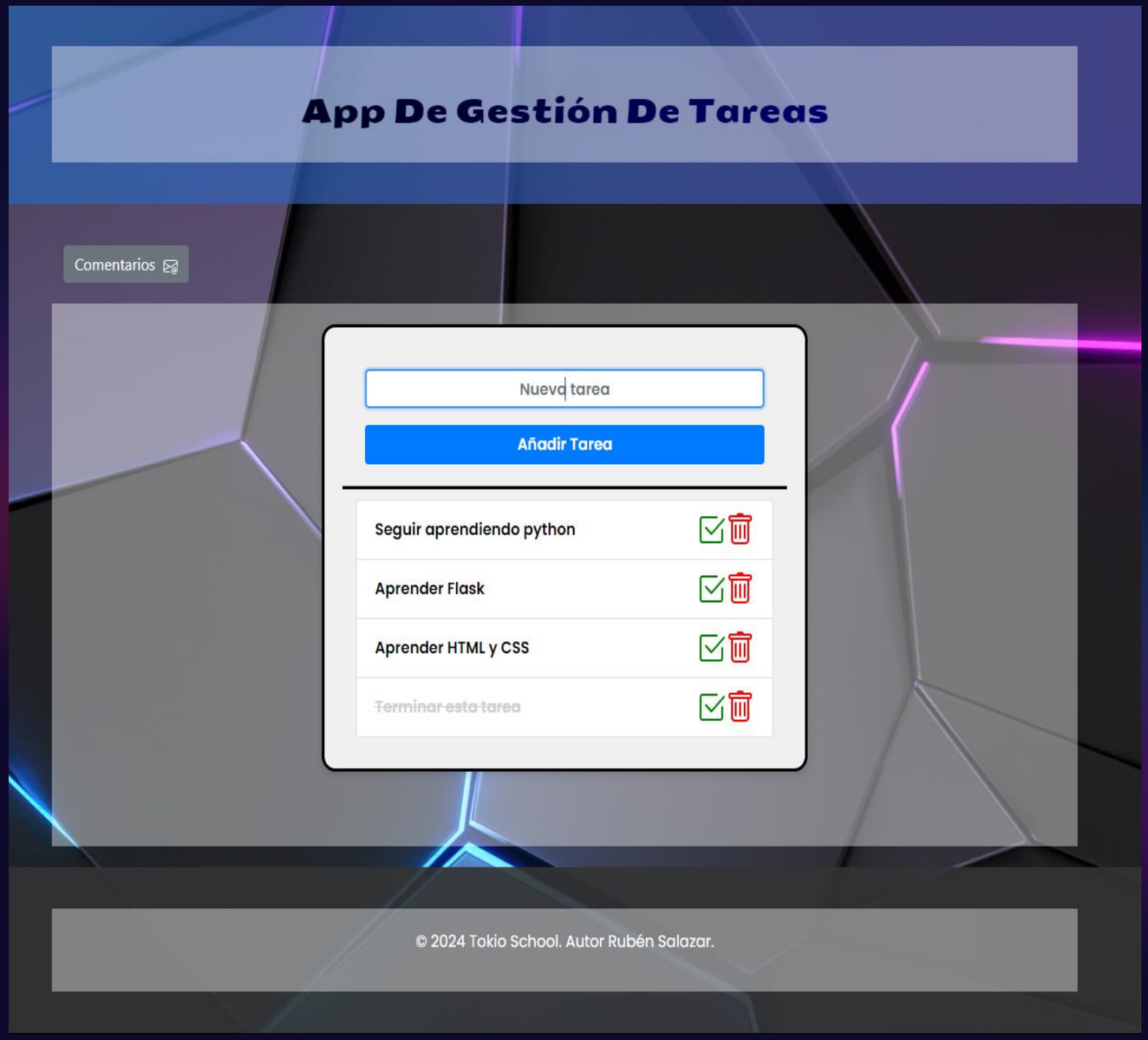
# Proyecto: App de Gestión de Tareas

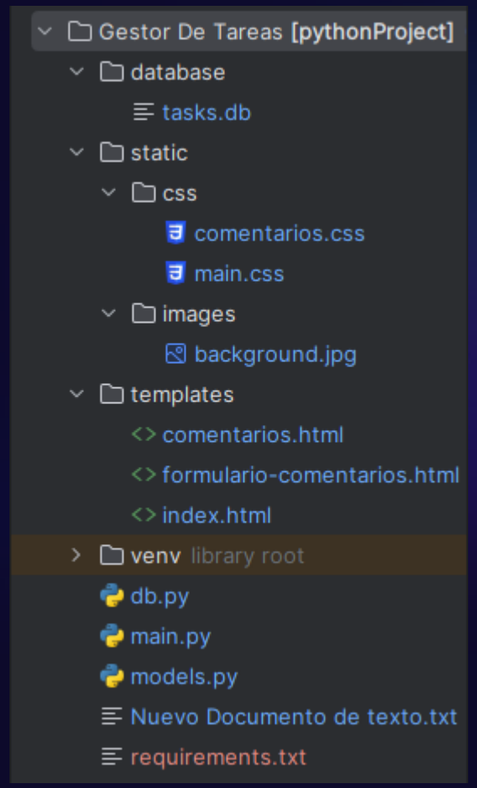
· La principal funcionalidad de este proyecto es crear una app-web que tenga la capacidad de gestionar tareas, visualizar las pendientes, poder marcarlas como hechas, y eliminarlas una vez no las necesitemos.

· A estas funcionalidades he añadido una sección de comentarios, en la cual puedes visualizar los comentarios existentes, y otra página con un formulario para poder añadir un nuevo comentario, además de tener la posibilidad de borrar cualquiera en cualquier momento.



# Estructura y configuración

## Estructura




## Tecnologías

```
SQLAlchemy~=2.0.30
Flask~=3.0.3
```

Nos basaremos en estas dos, pero dentro de ellas intervienen alguna más, como Jinja y datetime para la gestón de fechas entre otras.

## Configuraciones básicas

```python
def init_data():
    # Verificar si ya existen tareas y comentarios en la base de datos
    existing_tasks = db.session.query(Task).count()
    existing_comments = db.session.query(Comment).count()

    if existing_tasks == 0 and existing_comments == 0:
        #Si ambas estan vacias, el programa entiende que es la primera vez que se
        #ejecuta, por lo que creará las tareas y comentarios por defecto
        # Creamos instancias de Task
        task1 = Task(content="Seguir aprendiendo python",completed=False)
        task2 = Task(content="Aprender Flask",completed=False)
        task3 = Task(content="Aprender HTML y CSS",completed=False)
        task4 = Task(content="Terminar esta tarea",completed=True)

        # Creamos instancias de Comment
        comment1 = Comment(name="Ana",
                           comment="¡Increible apicación, "
                                   "me ha gustado mucho tanto el diseño como la funcionalidad!",
                           date="24 / 09 / 2020  -  10:45")
        comment2 = Comment(name="Pedro",
                           comment="Creo que necesita algunos cambios,"
                                   "como un inicio de sesión, por lo demás es una pagina web estupenda.",
                           date="06 / 10 / 2021  -  15:20")
        comment3 = Comment(name="Maria",
                           comment="Me parece estupendo y muy acertado "
                                   "el enfoque que se le ha dado a esta página web, es ¡ INCREIBLE !",
                           date="29 / 06 / 2022  -  09:10")
        comment4 = Comment(name="Juan",
                           comment="Quería aprovechar esta caja de comentarios para "
                                   "felicitar a Rubén Salazar Diaz por su esfuerzo en este curso y su gran trabajo, "
                                   "reflejado en esta página web tan bonita y práctica",
                           date="15 / 03 / 2023  -  14:00")

        # Agregamos instancias en la base de datos
        db.session.add_all([task1, task2, task3, task4, comment1, comment2, comment3, comment4])
        db.session.commit()
        db.session.close()
```

Dentro de las primeras configuraciones, se ha creado una función que detecte si la base de datos carece de entradas para crear unos ejemplos para el primer inicio, Tanto de tareas como de comentarios.

# Características y funcionalidades clave

1 Gestor de tareas

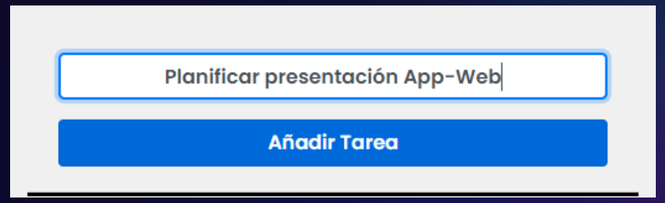
Permite a los usuarios crear y hacer seguimiento de tareas de manera eficiente.

2 Sistema de comentarios

Facilita la comunicación y el intercambio de ideas entre usuarios.

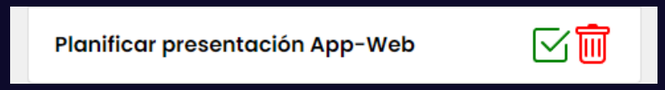
# Casos de uso y ejemplos de aplicación: Gestor de Tareas

## Creación de Tarea :



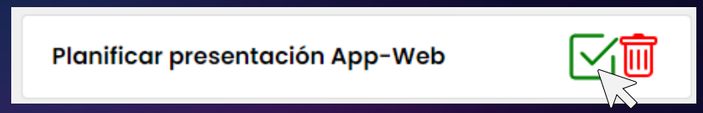

```python
@app.route( rule: "/create_task",methods=["POST"])
def create_task():
    task=Task(content=request.form["content_task"],completed=False)
    db.session.add(task)
    db.session.commit()
    return redirect(url_for("home"))
```

| | id | content | completed |
|---|---|---|---|
| | Filtro | Filtro | Filtro |
| 1 | 5 | Planificar presentación App-Web | 0 |



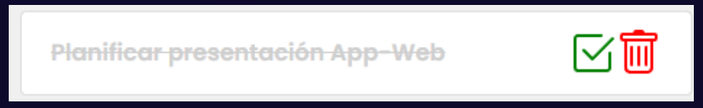

· Al añadir una tarea, se creará una entrada en la “db”, redirige al index y muestra todas las entradas que se encuentren el la “db”.

## Interactuar con las tareas :



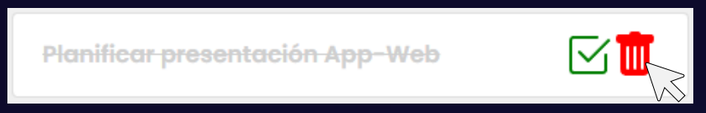

```python
@app.route("/completed_task/<int:task_id>")
def completed_task(task_id):
    task = db.session.query(Task).filter(Task.id == task_id).first()
    if task:
        task.completed = not task.completed  # Invertir el estado completado
        db.session.commit()
    return redirect(url_for("home"))
```

| | id | content | completed |
|---|---|---|---|
| | Filtro | Filtro | Filtro |
| 1 | 5 | Planificar presentación App-Web | 1 |




· Tenemos la posibilidad de marcar cualquier tarea como “completada”.
Al hacer click en el botón, actuará la función que cambia el estado dentro de la base de datos, y aplica los cambios en el estilo del texto mostrado.

## Eliminar las tareas :




```python
@app.route('/delete_task/<id>')
def delete(id):
    task = db.session.query(Task).filter_by(id=int(id)).delete()
    # Se busca dentro de la base de datos, aquel registro cuyo id coincida
    # con el aportado por el parametro de la ruta. Cuando se encuentra se elimina
    db.session.commit() # Ejecutar la operación pendiente de la base de datos
    return redirect(url_for('home')) # Esto nos redirecciona a la función home()
    # y si ha ido bien, al refrescar, la tarea eliminada ya no aparecera en el listado
```

| id | content | completed |
|---|---|---|
| Filtro | Filtro | Filtro |

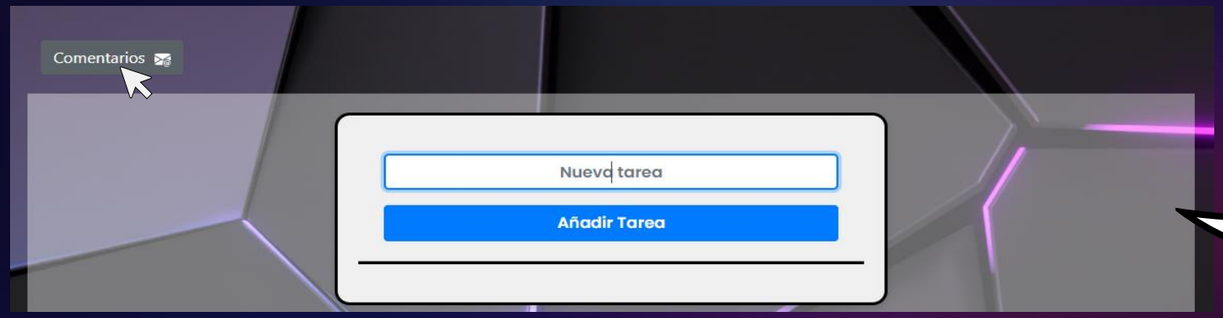

· Cuando actuamos sobre el botón de borrado, eliminará la entrada en la “db”, refresca la página, y desaparece dicha entrada.

# Casos de uso y ejemplos de aplicación: Comentarios

Comentarios :

· Existe un botón en la página principal, el cual te redirige a una página en la cual se muestran por defecto unos comentarios de ejemplo.

Se conecta con la “db” y muestra las entradas de la tabla de comentarios




```python
@app.route('/comentarios')
def comments():
    all_comments = db.session.query(Comment).order_by(Comment.id.desc()).all()
    return render_template( template_name_or_list: 'comentarios.html',comments_list=all_comments)
    #Busca en el archivo Comentarios para proporcionar la info que se encuentre en el.
    #Envia además la info de la base de datos al HTML
```

| | id | name | comment | date |
|---|---|---|---|---|
| | Filtro | Filtro | Filtro | Filtro |
| 1 | 1 | Ana | ¡Increible apicación, me ha gustado mucho tanto el diseño como la funcionalidad! | 24 / 09 / 2020 - 10:45 |
| 2 | 2 | Pedro | Creo que necesita algunos cambios,como un inicio de sesión, por lo demás es una ... | 06 / 10 / 2021 - 15:20 |
| 3 | 3 | María | Me parece estupendo y muy acertado el enfoque que se le ha dado a esta página ... | 29 / 06 / 2022 - 09:10 |
| 4 | 4 | Juan | Quería aprovechar esta caja de comentarios para felicitar a Rubén Salazar Diaz por ... | 15 / 03 / 2023 - 14:00 |

# Casos de uso y ejemplos de aplicación: Interactuar con los comentarios

## Borrar Comentarios :

· Podemos observer que la tarjeta de comentarios está dividida en dos partes:

- La parte de arriba muestra el nombre, la fecha en la que se publicó y un botón interactivo de borrado.
- La segunda muestra el contenido de dicho comentario.



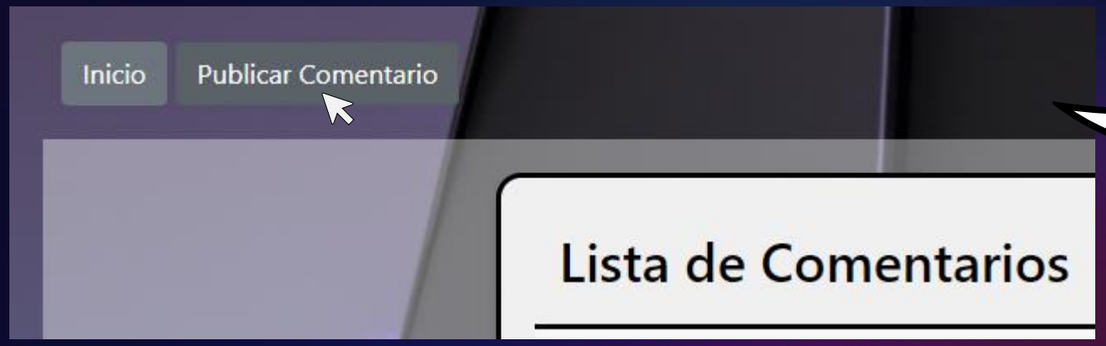
```python
@app.route('/delete_comment/<id>')
def delete_comment(id):
    comment=db.session.query(Comment).filter_by(id=int(id)).delete()
    # Se busca dentro de la base de datos, aquel registro cuyo id coincida
    # con el aportado por el parametro de la ruta. Cuando se encuentra se elimina
    db.session.commit() # Ejecutar la operación pendiente de la base de datos
    return redirect(url_for("comments")) # Esto nos redirecciona a la función comments()
    # y si ha ido bien, al refrescar, el comentario eliminado ya no aparecera en el listado
```

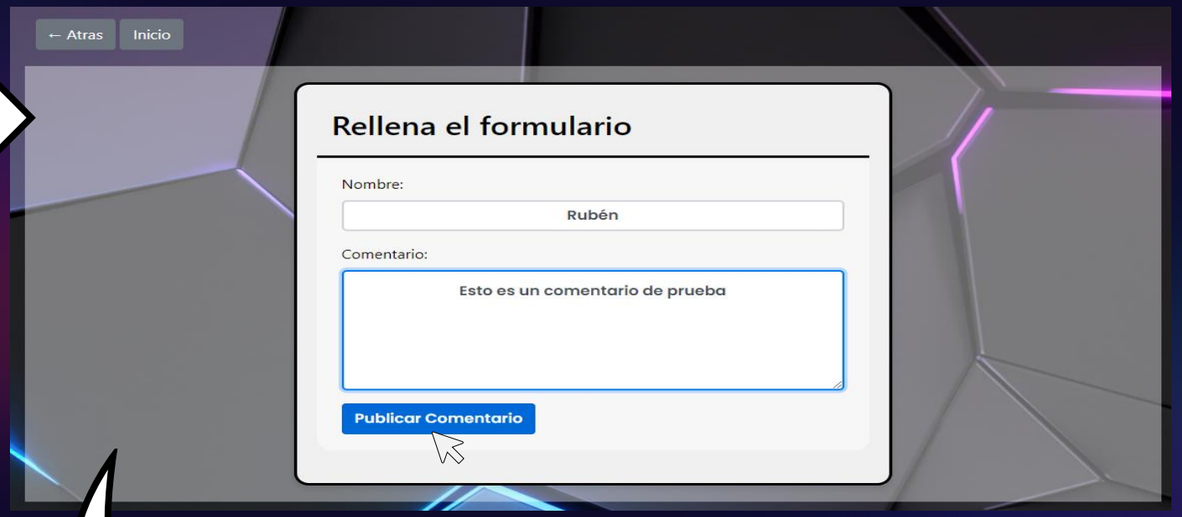
| | id | name | comment | date |
|---|---|---|---|---|
| | Filtro | Filtro | Filtro | Filtro |
| 1 | 1 | Ana | ¡Increible apicación, me ha gustado mucho tanto el diseño como la funcionalidad! | 24 / 09 / 2020 - 10:45 |
| 2 | 2 | Pedro | Creo que necesita algunos cambios,como un inicio de sesión, por lo demás es una ... | 06 / 10 / 2021 - 15:20 |
| 3 | 3 | María | Me parece estupendo y muy acertado el enfoque que se le ha dado a esta página ... | 29 / 06 / 2022 - 09:10 |

· Al hacer click en el botón de eliminar, actúa la función, busca en la “db”, y elimina la entrada, refresca la página y muestra las entradas existentes, omitiendo la recienmente eliminada.

# Casos de uso y ejemplos de aplicación: Formulario de Comentarios

· Los usuarios pueden enviar comentarios rellenando el formulario, actua la función y mediante “POST” creará una nueva entrada en la base de datos. Automaticamente redigirá a la página de comentarios y mostrará todos los comentarios que existan en la “db”.




```python
@app.route( rule: "/create_comment",methods=["POST"])
def create_comment():
    comment=Comment(name=request.form["name"],comment=request.form["comment"])
    db.session.add(comment)
    db.session.commit()
    return redirect(url_for("comments"))
```


| | id | name | comment | date |
|---|---|---|---|---|
| | Filtro | Filtro | Filtro | Filtro |
| 1 | 1 | Ana | ¡Increible apicación, me ha gustado mucho tanto el diseño como la funcionalidad! | 24 / 09 / 2020 - 10:45 |
| 2 | 2 | Pedro | Creo que necesita algunos cambios,como un inicio de sesión, por lo demás es una ... | 06 / 10 / 2021 - 15:20 |
| 3 | 3 | María | Me parece estupendo y muy acertado el enfoque que se le ha dado a esta página ... | 29 / 06 / 2022 - 09:10 |
| 4 | 5 | Rubén | Esto es un comentario de prueba | 20 / 07 / 2024 - 18:21 |

# Fuentes utilizadas para el Proyecto :

1

## Diseño

- Bootstrap
- Google Fonts
- Font Awesome

2

## Entornos de desarrollo

- Pycharm
- Db Browser ( para SQLite )

3

## Lenguajes de programación

- Python
- HTML
- CSS