Laboratorio

Materia: Seminario de lenguajes Python

Carrera: Licenciatura en Sistemas

Docentes: Mg. Lic. María Alejandra Vranic
Lic. Nicolás Borea
Lic. Gonzalo Cerbelli

Año cursada: 2025

Índice

1. Objetivo

El objetivo de este trabajo práctico es aplicar los conocimientos adquiridos durante la cursada para el desarrollo de una API REST que permite gestionar turnos entre personas. El enfoque estará puesto en la utilización del stack propuesto, buenas prácticas de desarrollo de código, manejo de datos, operaciones básicas (CRUD) y generación de reportes.

A. Stack tecnológico obligatorio

- Python 3.12
- FastAPI
- SQLite
- SQLAlchemy
- Pandas
- Borb

2. Formato de entrega

El formato de entrega se deberá respetar en la entrega de cada hito y en la entrega del trabajo final.

- Se deberá entregar video grupal con el aporte realizado en el trabajo.
- Entregar el link al repositorio github, donde se almacenará y versionará el código fuente del trabajo.

3. Evaluación

La cursada se aprueba con dos instancias de evaluación. La 2° instancia es también recuperatorio de la 1° instancia Por último hay una 3° instancia que es recuperatoria de la 2° instancia de evaluación.

Con respecto al examen final, será un nuevo hito de entrega del trabajo que se subirá/presentará mediante el campus, con fecha anterior a la del examen.

En la fecha del examen final, se deberá asistir con el trabajo, y se deberá abrir en las computadoras del laboratorio o personales, ejecutar casos de uso y defenderlos.

4. Metodología de trabajo en Github

Deberán crear un repositorio en GitHub donde alojarán el código de la aplicación, el primer commit deberá ser el proyecto base (configuraciones necesarias para levantar la aplicación) y además deberán actualizar el archivo Readme (lo crean al momento de crear el repositorio) con el nombre del grupo, integrantes y funcionalidades realizadas por cada uno.

Cada nueva funcionalidad deberá ser trabajada en una branch distinta a la main, y mergearse mediante Pull Request, el nombre de las branchs con las funcionalidades nuevas tendrá que tener el siguiente formato:

grupoNro-usuarioGitHub-nombreFuncionalidad.

Por ejemplo con la funcionalidad del Login: **grupo01-OscarRuina-VistaLogin**.

Al momento de *mergear* los cambios de la branch a main pueden utilizar la opción Squash and Merge (compacta todos los commits que tienen la branch en uno solo).

Importante:

- Tener en cuenta que los videos son la Defensa del Trabajo por lo que deben aparecer en ellos todos los miembros del grupo, claramente identificados y con cámara prendida.
- El tiempo del video es **estimado**, puede variar levemente en cada caso, pero siempre deben visualizarse todas las funcionalidades implementadas.
- En el video cada miembro del equipo debe indicar claramente qué funcionalidad requerida realizó.
- La defensa y el proyecto serán válidos con las condiciones planteadas por el equipo docente.

Se calificará el trabajo con una nota grupal y una nota personal en base a lo realizado por cada integrante del equipo.

5. Consignas

A. ABM de personas

Se deberá implementar una API que permita gestionar personas que solicitan turnos, permitiendo crear una nueva persona, obtener el listado de personas registradas en la tabla de base de datos, obtener una persona en particular, actualizar los datos de una persona y eliminar una persona. Para ello, los endpoints serán:

- POST /personas
- GET /personas
- GET /personas/{id}
- PUT /personas/{id}
- DELETE /personas/{id}

Las personas deberán contar con los siguientes datos:

- Nombre
- Email
- DNI
- Teléfono
- Fecha de nacimiento
- Edad (dato calculado según fecha de nacimiento)
- Habilitado o inhabilitado para sacar turno. Por defecto una persona está habilitada.

Ejemplo de cuerpo de endpoint POST /personas:

```
{
 "nombre": "Carla Gómez",
```

```
 "email": "carla.gomez@gmail.com",
 "dni": "11111111",
 "telefono": "1162223333",
 "fecha_nacimiento": "2000-01-01",
 "edad": 25
}
```

B. ABM de turnos

Se deberán implementar nuevos endpoints para gestionar turnos que serán solicitados por las personas, permitiendo crear un nuevo turno, obtener el listado de turnos registrados en la tabla de base de datos, obtener un turno en particular, actualizar los datos de un turno y eliminar un turno. Para ello, los endpoints serán:

- POST /turnos
- GET /turnos
- GET /turnos/{id}
- PUT /turnos/{id}
- DELETE /turnos/{id}

Los turnos deberán contar con los siguientes datos:

- Fecha
- Hora
- Estado
- Persona que solicitó el turno

Los estados posibles de un turno son pendiente, cancelado, confirmado y asistido. Los turnos deberán ser creados con estado "pendiente".

Ejemplo de cuerpo de endpoint POST /turnos:

```
{
 "fecha": "2000-01-01",
 "hora": "10:30",
 "estado": "pendiente",
 "persona": {datos de la persona}
}
```

Para esta consigna, deberán implementarse las siguientes reglas de negocio:

- No se deberá permitir asignar un turno a la persona si esta ya tiene 5 turnos o más con estado cancelado en los últimos 6 meses

C. Cálculo de turnos disponibles

Se deberá implementar un nuevo endpoint que permita obtener los turnos disponibles de una fecha específica.

El endpoint deberá retornar un JSON con los horarios disponibles de la fecha solicitada.

- GET /turnos-disponibles?fecha=YYYY-MM-DD

Los turnos disponibles son aquellos que no fueron asignados, o que su estado es "cancelado".

El cálculo de turnos disponibles se deberá realizar en base a los turnos solicitados para una fecha específica en la base de datos. Los turnos podrán ser solicitados de 09 a 17 horas, en intervalos de 30 minutos. Es decir, si el turno de 13:00 hs está tomado, el endpoint de consulta de turnos disponibles debería retornar el turno de 12:30 y 13:30 horas.

Ejemplo de respuesta del endpoint con el caso mencionado arriba:

```
{
 "fecha": "2000-01-01",
 "horarios_disponibles": ["09:00", "09:30", "10:00", "10:30", "11:00",
"11:30", "12:00", "12:30", "13:30", "14:00", "14:30", "15:00", "15:30",
"16:00", "16:30"]
}
```

D. Gestión de estado de turno

Se deberá implementar la lógica para gestionar el ciclo de vida de un turno. Los estados posibles son: pendiente, cancelado, confirmado y asistido.

Deberán implementarse dos nuevos endpoints que permitan cancelar un turno o confirmar su asistencia. Para ello, los endpoints serán:

- PUT /turnos/{id}/cancelar
- PUT /turnos/{id}/confirmar

Ejemplo de respuesta del endpoint:

```
{
 "id": 3,
 "persona": {datos de la persona},
 "fecha": "2000-01-01,
 "hora": "13:30",
 "estado": "cancelado"
}
```

Para esta consigna, deberán implementarse las siguientes reglas de negocio:

- No se puede modificar un turno asistido o cancelado.
- No se pueden eliminar o cancelar turnos asistidos.
- Los turnos cancelados liberan el horario porque la eliminación es lógica.

E. Endpoints de reportes

Se deberán implementar endpoints que permitan consultar:

- Turnos de una fecha: recibir fecha por parámetro y devolver todos los turnos de ese día con la información de la persona que los solicitó (nombre y DNI), sin discriminar por estado del turno
- Turnos cancelados mes actual: realizar la query a la base de datos con un group by y devolver los turnos cancelados en el mes en curso, con su información
- Turnos de una persona: recibir DNI de la persona por parámetro y devolver todos los turnos de esa persona, sin discriminar por estado del turno
- Personas con 5 turnos cancelados como mínimo: devolver un arreglo con la información de las personas que tengan mayor cantidad a 5 de turnos cancelados, indicando cuantos turnos cancelados tienen y su detalle
- Turnos confirmados en un período de tiempo: recibir una fecha desde y hasta (inclusive) y buscar todos los turnos confirmados en la base de datos del período indicado. Paginar los resultados mostrando sólo 5 registros por página.
- Personas habilitadas o inhabilitadas a sacar turno:

Deberán implementarse cuatro endpoints que permitan obtener estos reportes. Para ello, los endpoints serán:

- GET /reportes/turnos-por-fecha?fecha=YYYY-MM-DD
- GET /reportes/turnos-cancelados-por-mes
- GET /reportes/turnos-por-persona?dni=12345678
- GET /reportes/turnos-cancelados?min=5
- GET /reportes/turnos-confirmados?desde=YYYY-MM-DD&hasta=YYYY-MM-DD
- GET /reportes/estado-personas?habilitada=true/false

Ejemplo de respuesta del endpoint de turnos cancelados en el mes en curso:

```
{
    "anio": 2000,
    "mes": "enero",
    "cantidad": 2,
    "turnos": [
        {
            "id": 3,
            "persona_id": 454,
            "fecha": "2000-01-01",
            "hora": "13:30",
            "estado": "cancelado"
        },
        {
            "id": 4,
            "persona_id": 455,
```

```
            "fecha": "2001-01-01",
            "hora": "10:00",
            "estado": "cancelado"
        }
    ]
}
```