PRÁCTICA 4

RELACIONES CON UNA BASE DE DATOS

Equipo:
Beltrán Saucedo Axel Alejandro
Cerón Samperio Lizeth Montserrat
Higuera Pineda Angel Abraham
Lorenzo Silva Abad Rey
4BV1.

ESCUELA SUPERIOR DE CÓMPUTO

TECNOLOGÍAS PARA EL DESARROLLO DE APLICACIONES WEB

13/10/2025

Índice

1. Introducción **2**

2. Fundamentos Teóricos **2**

3. Diagrama UML **5**

4. Implementación **5**

Referencias **9**

1. Introducción

Planteamiento del problema

Se busca crear una API web cuyo propósito sea gestionar un inventario y optimizar envíos con ayuda del algoritmo genético simple. Para la gestión del inventario, se debe garantizar lo siguiente:

- Permitir crear, leer, actualizar y borrar items, categorías y envíos en una base de datos.

Para la optimización de envíos, se debe garantizar lo siguiente:

- Ofrecer un endpoint que utilice el AGS para resolver el problema de la mochila. Que calcule la combinación de items que maximiza la ganancia total de un envío sin exceder una capacidad de peso determinada.

Propuesta de solución

La solución propuesta es el desarrollo de una API con FastAPI para la lógica de negocio y SQLModel para la gestión de la base de datos. Componentes clave de la solución:

- **SQLModel:** Para definir la estructura de la base de datos estableciendo categoría, item y envío como entidades principales.
- **endpoints (URLs):** Para crear, leer, actualizar y borrar items, categorías y envíos.
- **Algoritmo Génetico:** Realizará el calculo. Simulando un proceso de evolución para encontrar la mejor combinación de items que da la mayor ganancia total sin superar la capacidad.

2. Fundamentos Teóricos

Para la correcta creacion de nuestra práctica, es importante que entendamos las relaciones entre las tecnologías y los conceptos que ya revisados para la creación de una API, esta vez implementando persistencia de datos.

Entorno de Trabajo y Herramientas Principales

- **Python:** Es el lenguaje de programación sobre el que se construye toda la lógica de la aplicación. Su sencilla sintaxis y su amplia cantidad de librerías lo hacen ideal para el desarrollo web.[2]
- **FastAPI:** Framework web moderno para construir APIs con Python. Sus características más destacadas son la rapidez, la validación de datos automática mediante Pydantic y la generación de documentación interactiva , que fue crucial para probar los endpoints de nuestra API.
- **Uvicorn:** Es un servidor ASGI ultrarrápido, utilizado para ejecutar la aplicación FastAPI. Permite que la API maneje múltiples peticiones de forma asíncrona, mejorando el rendimiento.[3]

Persistencia de Datos

La persistencia de datos es la capacidad de un sistema para poder conservar la información posterior de la duración de una sola ejecución. En nuestra practica pasada, los datos eran volátiles, ya que se almacenaban en una lista en memoria. En esta práctica, se implementa la persistencia a través de los siguientes componentes:

- **SQLite:** Es un motor de base de datos relacional, autocontenido y que no requiere un servidor. Almacena toda la base de datos en un único archivo (en nuestro caso, `database.db`). Es ideal para desarrollo y aplicaciones de pequeña a mediana escala por su simplicidad y portabilidad.
- **SQLAlchemy:** Es una librería que funciona como un Mapeador Objeto-Relacional (ORM). Un ORM es una técnica que actúa como un "traductor.^entre el código orientado a objetos y las tablas de una base de datos relacional. Permite manipular la base de datos utilizando código Python en lugar de escribir consultas SQL directamente.
- **SQLModel:** Es una librería que combina **SQLAlchemy** y **Pydantic**, creada por el mismo autor de FastAPI. Permite definir la estructura de los datos, las validaciones y el esquema de la base de datos en una sola clase, reduciendo la duplicación de código y simplificando el desarrollo. En nuestra práctica, las clases como `Item` son modelos de SQLModel que representan tanto la tabla en la base de datos como los datos que la API recibe y envía.[4]

Relaciones en Bases de Datos

- **Relación Unívoca:** Cada valor de clave primaria se relaciona con sólo un registro en la tabla relacionada.
- **Uno a varios:** La tabla de claves primaria sólo contiene un registro que se relaciona con ninguno, uno o varios registros en la tabla relacionada.
- **Varios a varios:** Cada registro en ambas tablas puede estar relacionado con varios registros en la otra tabla. Este tipo de relaciones requieren una tercera tabla, denominada tabla de enlace o asociación, porque los sistemas relacionales no pueden alojar directamente la relación. [5]

API CRUD

Una API CRUD es una interfaz de programación que permite Crear, Leer, Actualizar y Borrar datos. [6]

Algoritmo Genético Simple (AGS)

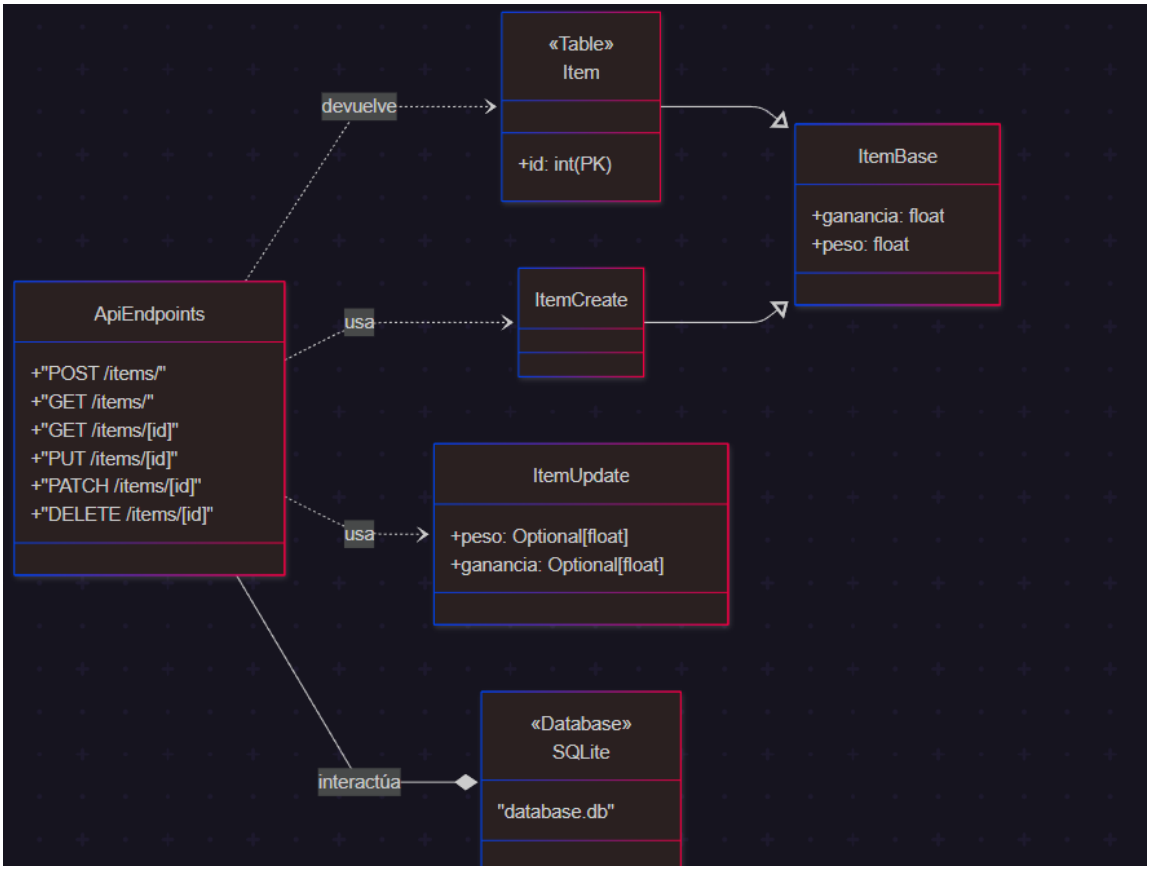
Está basado en el proceso genetico de los organismos vivos. Los Algoritmos Geneticos usan una analogía directa con el comportamiento natural.

- **Población:** Trabajan con una poblacion de individuos, cada uno de los cuales representa una solucion factible a un problema dado. A cada individuo se le asigna un valor o puntuacion, relacionado con la bondad de dicha solucion.
- **Fitness:** Cuanto mayor sea la adaptacion de un individuo al problema, mayor sera la probabilidad de que el mismo sea seleccionado.
- **Cruzamiento:** Será seleccionado para reproducirse, cruzando su material genetico con otro individuo seleccionado de igual forma. Este cruce producira nuevos individuos descendientes de los anteriores los cuales comparten algunas de las caractersticas de sus padres. Cuanto menor sea la adaptacion de un individuo, menor sera la probabilidad de que dicho individuo sea seleccionado para la reproduccion, y por tanto de que su material genetico se propague en sucesivas generaciones.[7]

De esta manera se produce una nueva poblacion de posibles soluciones, la cual reemplaza a la anterior y verifica la interesante propiedad de que contiene una mayor proporcion de buenas caractersticas en comparacion con la poblacion anterior. Así a lo

largo de las generaciones las buenas caractersticas se propagan a travees de la poblacion. Favoreciendo el cruce de los individuos mejor adaptados, van siendo explo radas las areas mas prometedoras del espacio de busqueda. Si el Algoritmo Genetico ha sido bien disenado, la poblacion convergera hacia una solucion optima del proble ma.

3. Diagrama UML



4. Implementación

Código:

En esta seccion tendremos los nuevos modulos que usamos para el funcionamiento de la practica.

Debemos entender la diferencia entre esta practica y la anterior, ya que pueden parecer iguales, pero tienen una diferencia crucial. En la practica anterior, nuestra API gestionaba los datos de los items en una lista en memoria, lo cual tenia una principal desventaja, que al reiniciar nuestro servidor, todos los datos se perdian. En cambio, esta vez integramos una base de datos para asegurar la permanencia de los datos.

```python
#FastApi es el modulo principal, HTTPException es un manejador de errores, como el "no encontrado"
#Status es un modulo de indentificacion de codigos de estado HTTP, como el clasico 404
#Depends es un mecanismo para poder inyectar dependencias
from fastapi import FastAPI, HTTPException, status, Depends
#Field es usado para definir las propieddes de los campos del modelo
#create_engine se usa para crear un motor que conectara la base de datos con la aplicacion
#Session es el enncargado de manejar la comunicacion con la base de datos
#SQLModel sera la base para los modelos de datos
from sqlmodel import Field, create_engine, Session, SQLModel
#Optional y List son tipos de datos que permiten definir campos opcionales y listas
#Se usan para definir los modelos de datos y las respuestas de la API
from typing import Optional, List
#Annotated es una forma para escribir tipos con metadatos
from typing_extensions import Annotated
```

El principal cambio que podemos ver aqui es la adicion de la libreria SQLModel, la cual nos permitira interactuar con la base de datos, gracias a sus capacidades provenientes de SQLAlchemy y Pydantic.

```python
#Item contiene el indentificador unico (id) y hereda los campos de ItemBase
#y con table=True se marca como una tabla
class Item(ItemBase, table=True):
    id: Optional[int]=Field(default=None, primary_key=True)
#Los campos son opcionales, ya que podemos querer actualizar solo uno de ellos
#El valor por defecto es None
#primary_key marca el campo como la clave primaria(ID)
```

En esta parte, podemos ver que los modelos de datos fueron redefinidos usando SQLModel y asi poder mapearlos en una tabla en nuestra base de datos. Se agrego 'table=true', que le indica a SQLModel que la clase esta representando una tabla en la base. Otro cambio es en id, que ahora lo definimos como la clave primaria de nuestra tabla. Esto hara que la base de datos se encargue de ir generando los id's de forma automatica.

```python
#ItemOut nos ayudara a colocar en el orden deseado nuestras respuestas
class ItemOut(SQLModel):
    peso: float
    ganancia: float
    id: int
```

Esta funcion tiene el unico proposito de mostrarnos los datos en el orden deseado, el cual es peso/ganancia/id, ya que originalmente los mostraba en desorden.

```python
# Conexion a la base de datos

#La cadena sqlite:///database.db indica que vamos a usar un archivo local llamda 'database.db'
#como nuestra base de datos
sql_url="sqlite:///databse.db"
#Crea el motor de la base de datos, que sera el encargado de gestionar la conexion
engine=create_engine(sql_url)
```

Aqui tenemos la nueva adicion mas importante, la cual se encargara de la configuracion y manejo de la conexion con la base de datos (que estara en SQLite). Esta sera almacenada en un archivo bajo el nombre de database.db. Tambien tenemos lo que vendria a ser nuestro 'motor', por decirlo de una forma, que funcionara como el centro de comunicaciones entre la API y nuestro database.db

```python
#Funcion encarga de crear la tabla 'item' en la base de datos
def create_db_and_tables():
    #SQLModel.metadata... sera el que checara que todos los modelos que heredan de SQLModel,
    #y que ademas esten marcado con table=True, y les creara sus tablas en la base de datos
    SQLModel.metadata.create_all(engine)
```

Esta funcion sera la encargada de la creacion de las tablas y solo se ejecutara una vez al iniciar la aplicacion. Crea las tablas correspondientes en la base si es que no existen.

```python
#Funcion encargada de crear y proporcionar una sesion de base de datos por cada peticion
def get_session():
    #Abre una nueva sesion con el motor(engine) de la base de datos
    with Session(engine) as session:
        #Yield dara la sesion al endpoint de nuestra API
        yield session
#SessionDep es una anotacion que usa a Depends para inyectar una sesion de base de datos
#en nuestras funciones
SessionDep=Annotated[Session, Depends(get_session)]
```

Aqui implementamos un sistema de sesiones por peticion. Esto se refiere a que cada vez que la API recibe una solicitud, una nueva sesion de comunicacion con la base de datos se abre, y esta misma se cierra en automatico al terminar la solicitud, lo cual nos garantiza un manejo seguro de las conexiones.

```python
#On_event le dice a FastAPI que ejecute lo siguiente una sola vez cuando iniciamos la aplicacion
@app.on_event("startup")
def on_startup():
    #Llamamos a create_db... para tener la seguridad de que la base de datos y sus tablas ya fueron
    #creadas antes de que la aplicacion reciba peticiones
    create_db_and_tables()
```

Este evento se ejecuta en el instante en el momento en que la API se termine de iniciar,antes de que cualquier usuario mande una peticion y solo se ejecuta una vez. Y esto no garantiza que la base de datos y sus tablas ya existan antes de que los usuarios empiezen a mandar peticiones.

Referencias

[1] IBM. (s.f.). ¿Qué es una API REST? Recuperado el 14 de octubre de 2025, de https://www.ibm.com/mx-es/topics/rest-apis

[2] Python Software Foundation. (s.f.). Acerca de Python™. Resumen Ejecutivo. Recuperado el 14 de octubre de 2025, de https://www.python.org/doc/essays/blurb-es/

[3] Ramírez, S. (s.f.). FastAPI. Recuperado el 14 de octubre de 2025, de https://fastapi.tiangolo.com/es/

[4] Ramírez, S. (s.f.). SQLModel. Recuperado el 14 de octubre de 2025, de https://sqlmodel.tiangolo.com/es/

[5] IBM Control Desk. (s. f.). Relaciones en Bases de Datos. Recuperado el 31 de octubre de 2025, de https://www.ibm.com/docs/es/control-desk/7.6.1?topic=structure-database-relationships

[6] Jain, A. (2024, 8 septiembre). What is CRUD API? DEV Community. Recuperado el 31 de octubre de 2025, de https://dev.to/ankitjaininfo/what-is-crud-api-502i

[7] Tema 2. Algoritmos genéticos. (s. f.). Departamento de Ciencias de la Computación E Inteligencia Artificial. Recuperado el 31 de octubre de 2025, de http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t2geneticos