

Crea la aplicación y la tabla si no existiera

The screenshot shows a code editor interface with several tabs at the top: models.py, main.py (selected), usuarios.py, insertar.py, and database.py. The left sidebar shows a project structure under 'CODE' with files like \_\_pycache\_\_, main.cpython-314.pyc, app, routers, usuarios.py, \_\_init\_\_.py, database.py, insertar.py, main.py, and models.py. The main editor area displays the following Python code:

```
app > main.py > ...
1  from fastapi import FastAPI
2  from .database import engine, metadata
3  from .routers import usuarios
4
5  # Crea la aplicación FastAPI
6  app = FastAPI()
7
8  # Crea las tablas si no existe
9  metadata.create_all(engine)
10
11 # Registra las rutas del ruter de usuarios
12 app.include_router(usuarios.router)
```

Defino la tabla crear

The screenshot shows a code editor interface with tabs for models.py (selected), main.py, usuarios.py, insertar.py, and database.py. The main editor area displays the following Python code:

```
app > models.py > ...
1  from sqlalchemy import Table, Column, Integer, String
2  from .database import metadata
3
4  # Definición de la tabla
5  usuarios = Table(
6      'usuarios', # Nombre de la tabla
7      metadata, # Metadata donde se registrará la tabla
8      # El ID se incrementara con cada nuevo usuario
9      Column('id_usuario', Integer, primary_key=True, autoincrement=True),
10     # "nullable=False" = obligatorio rellenar campos
11     Column('nombre', String(20), nullable=False),
12     Column('apellidos', String(50), nullable=False),
13     Column('año_nacimiento', Integer, nullable=False)
14 )
```

Creo un router para agrupar las rutas relacionadas con los usuarios.

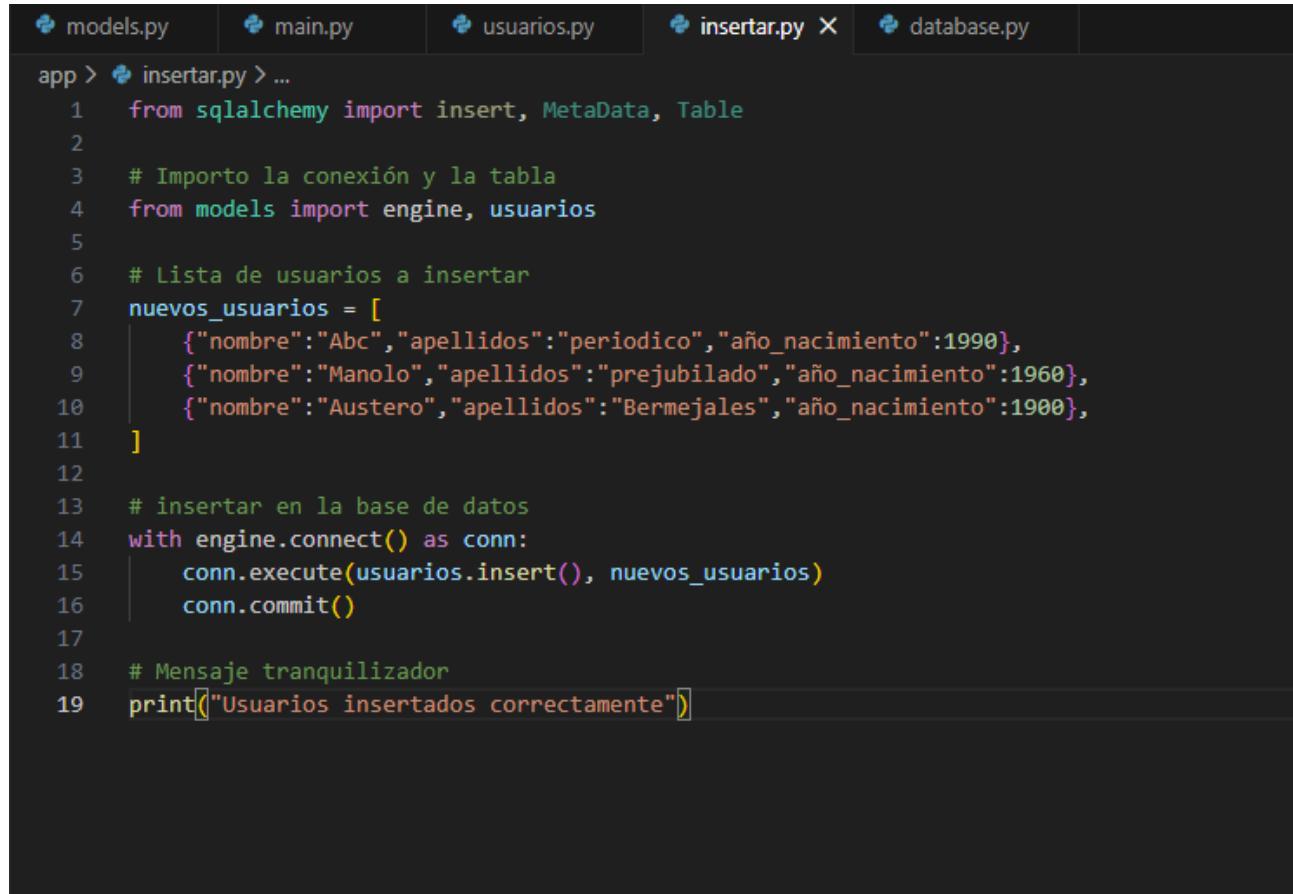
Abro las dependencias de sesión

Llamo a la clase de insertar usuarios

+ el de listar usuarios en -JSON

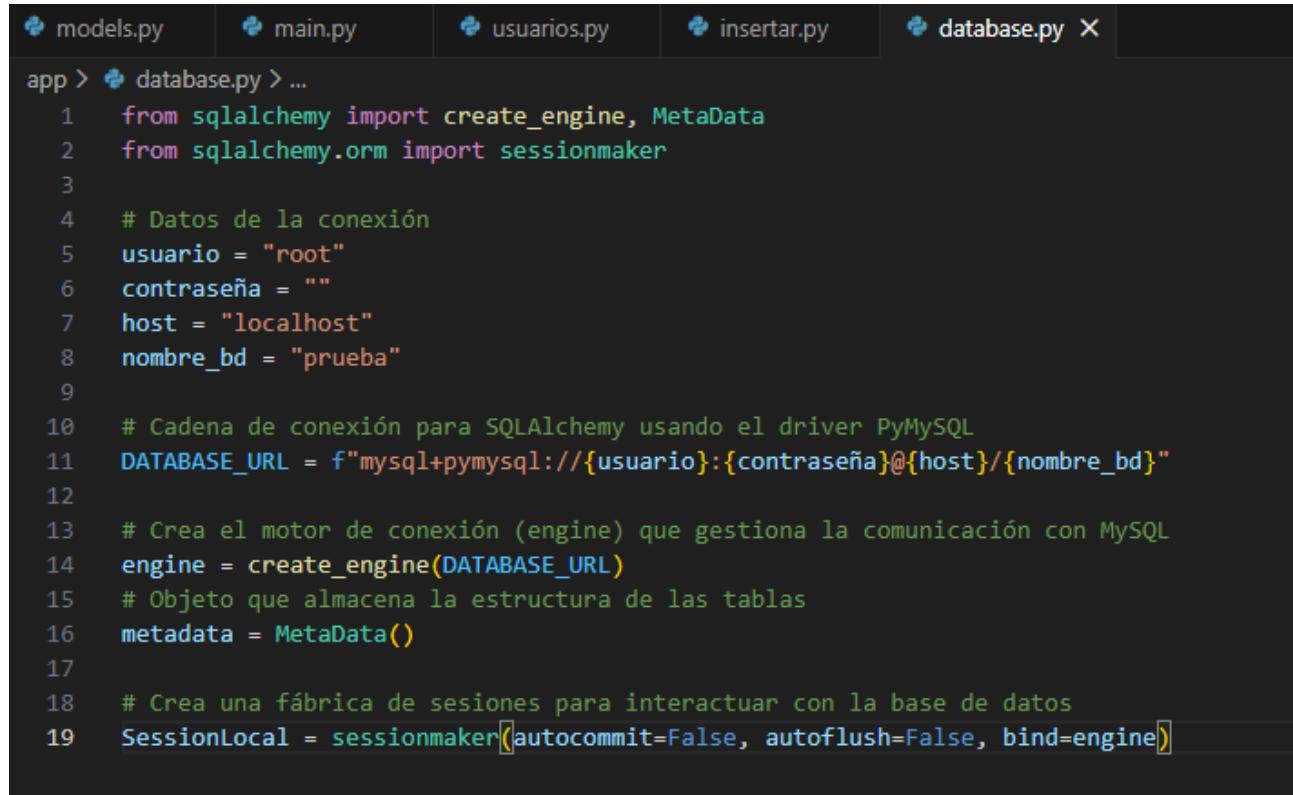
```
models.py main.py usuarios.py X insertar.py database.py
app > routers > usuarios.py > listar_usuarios
1  from fastapi import APIRouter, Depends
2  from sqlalchemy.orm import Session
3  from ..database import SessionLocal
4  from ..models import usuarios
5
6  # Crea un router para agrupar las rutas relacionadas con usuarios
7  router = APIRouter(prefix="/usuarios", tags=["Usuarios"])
8
9  # Dependencia que crea y cierra una sesión en la base de datos por petición
10 def get_db():
11     db = SessionLocal()      #Abre la sesion
12     try:
13         yield db            #Devuelve la ruta solicitada
14     finally:
15         db.close()          #Se cierra al terminar la petición
16
17 #Creación de usuario
18 @router.post("/")
19 def crear_usuario(data: dict, db: Session= Depends(get_db)):
20     # Construye una sentencia SQL INSERT con los datos recibidos
21     query = usuarios.insert().values(data)
22     # Ejecuta la sentencia en la base de datos
23     db.execute(query)
24     # Guarda los cambios
25     db.commit()
26     # Mensaje tranquilizador de todo OK
27     return{"mensaje":"Usuario creado correctamente"}
28
29 # Listar usuarios
30 @router.get("/")
31 def listar_usuarios (db: Session= Depends(get_db)):
32     # Construye una sentencia SELECT * FROM usuarios (obtiene todo los datos de la tabla 'usuarios')
33     query = usuarios.select()
34     # Ejecuta la consulta y obtiene todos los registros
35     resultado = db.execute(query).fetchall()
36     # Convierto el resultado que esta en ROW a JSON
37     usuarios_lista = [dict(row._mapping) for row in resultado]
38     # Devuelve los datos ahora en JSON
39     return usuarios_lista
```

## La clase del Insert



```
models.py main.py usuarios.py insertar.py database.py
app > insertar.py > ...
1  from sqlalchemy import insert, MetaData, Table
2
3  # Importo la conexión y la tabla
4  from models import engine, usuarios
5
6  # Lista de usuarios a insertar
7  nuevos_usuarios = [
8      {"nombre": "Abc", "apellidos": "periodico", "año_nacimiento": 1990},
9      {"nombre": "Manolo", "apellidos": "prejubilado", "año_nacimiento": 1960},
10     {"nombre": "Austero", "apellidos": "Bermejales", "año_nacimiento": 1900},
11 ]
12
13 # insertar en la base de datos
14 with engine.connect() as conn:
15     conn.execute(usuarios.insert(), nuevos_usuarios)
16     conn.commit()
17
18 # Mensaje tranquilizador
19 print("Usuarios insertados correctamente")
```

## Conexión con la base de datos



```
models.py main.py usuarios.py insertar.py database.py
app > database.py > ...
1  from sqlalchemy import create_engine, MetaData
2  from sqlalchemy.orm import sessionmaker
3
4  # Datos de la conexión
5  usuario = "root"
6  contraseña = ""
7  host = "localhost"
8  nombre_bd = "prueba"
9
10 # Cadena de conexión para SQLAlchemy usando el driver PyMySQL
11 DATABASE_URL = f"mysql+pymysql://{usuario}:{contraseña}@{host}/{nombre_bd}"
12
13 # Crea el motor de conexión (engine) que gestiona la comunicación con MySQL
14 engine = create_engine(DATABASE_URL)
15 # Objeto que almacena la estructura de las tablas
16 metadata = MetaData()
17
18 # Crea una fábrica de sesiones para interactuar con la base de datos
19 SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
```

Para poder iniciar la app se necesita la clase `-init-.py` obligatoriamente aunque este vacía.

routers	03/02/2026 13:30	Carpeta de archivos
__init__.py	03/02/2026 13:33	Python File

Ejecuto en la consola de comando desde la carpeta del proyecto.

```
uvicorn app.main:app --reload
```

Y se inicia.

```
Microsoft Windows [Versión 10.0.26200.7623]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\code>uvicorn app.main:app --reload
INFO:     Will watch for changes in these directories: ['C:\\\\code']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [15256] using WatchFiles
INFO:     Started server process [15104]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Ahora entro en “<http://127.0.0.1:8000/docs>”

The screenshot shows a browser window displaying the FastAPI documentation at <http://127.0.0.1:8000/docs>. The page title is "FastAPI 0.1.0 OAS 3.1". Below the title, there is a link to `/openapi.json`.

The main content area is titled "Usuarios". It lists two endpoints:

- GET /usuarios/** Listar Usuarios
- POST /usuarios/** Crear Usuario

Below the endpoints, there is a section titled "Schemas" which contains two expanded definitions:

- HTTPValidationError >** Expand all `object`
- ValidationError >** Expand all `object`

Ejecuto desde Usuarios POST los usuarios a añadir

## Usuarios

The screenshot shows a REST API interface for creating a user. At the top, there's a GET method for listing users. Below it, a POST method for creating a user is selected. The 'Parameters' section shows 'No parameters'. The 'Request body' section is set to 'application/json' and contains the following JSON:

```
{  
    "nombre": "Osito",  
    "apellidos": "Volador O no",  
    "año_nacimiento": 2077  
}
```

Muestra el mensaje con el contenido añadido y mas abajo el de confirmación

The screenshot shows a terminal window with a curl command to create a user. The command includes the URL 'http://127.0.0.1:8000/usuarios/', headers for accept and content-type as application/json, and a JSON body with 'nombre', 'apellidos', and 'año\_nacimiento' fields. The response shows a success message: 'Usuario creado correctamente'.

```
Curl  
curl -X 'POST' \  
  'http://127.0.0.1:8000/usuarios/' \  
  -H 'accept: application/json' \  
  -H 'Content-Type: application/json' \  
  -d '{  
    "nombre": "Osito",  
    "apellidos": "Volador O no",  
    "año_nacimiento": 2077  
}'
```

Request URL  
`http://127.0.0.1:8000/usuarios/`

Server response

Code	Details
200	<p>Response body</p> <pre>{     "mensaje": "Usuario creado correctamente" }</pre> <p>Download</p>

Response headers

```
content-length: 42  
content-type: application/json  
date: Tue,03 Feb 2026 13:28:30 GMT  
server: uvicorn
```

Responses

Code	Description	Links
------	-------------	-------

Ahora en el Usuarios GET obtengo la lista de usuarios en la tabla

## Usuarios

GET /usuarios/ Listar Usuarios

Parameters

No parameters

Cancel

Execute Clear

### Responses

Curl

```
curl -X 'GET' \
'http://127.0.0.1:8000/usuarios/' \
-H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/usuarios/
```

Server response

Code Details

Code	Details
200	<p>Response body</p> <pre>"apellidos": "Pérez Gómez", " año_nacimiento": 1995 }, { "id_usuario": 4, "nombre": "Fran", "apellidos": "Tir Tor", " año_nacimiento": 1997 }, { "id_usuario": 5, "nombre": "Fran", "apellidos": "Tir Tor", " año_nacimiento": 0 }, { "id_usuario": 6, "nombre": "Osito", "apellidos": "Volador O no", " año_nacimiento": 2077 }, { "id_usuario": 7, "nombre": "Osito", "apellidos": "Volador O no", " año_nacimiento": 2077 } ]</pre> <p>Download</p> <p>Response headers</p> <pre>content-length: 406 content-type: application/json date: Tue,03 Feb 2026 13:30:02 GMT server: uvicorn</pre>

Los datos coinciden con la tabla vista desde MYSQL

Servidor: 127.0.0.1 » Base de datos: prueba » Tabla: usuarios

Examinar Estructura SQL Buscar Insertar Exportar Imprimir

Mostrando filas 0 - 3 (total de 4, La consulta tardó 0,0001 segundos.)

```
SELECT * FROM `usuarios`
```

Perfilando [ Editar en línea ] [ Editar ] [ Explicar SQL ] [ Crear código PHP ] [ Actualizar ]

Mostrar todo | Número de filas: 25 ▾ Filtrar filas: Buscar en esta tabla Order by

Opciones extra

		id_usuario	nombre	apellidos	año_nacimiento
<input type="checkbox"/>	Editar  Copiar  Borrar	1	Juan	Pérez Gómez	1995
<input type="checkbox"/>	Editar  Copiar  Borrar	4	Fran	Tir Tor	1997
<input type="checkbox"/>	Editar  Copiar  Borrar	5	Fran	Tir Tor	0
<input type="checkbox"/>	Editar  Copiar  Borrar	6	Osito	Volador O no	2077

