**Panduann Setup FastAPI & SQLAlchemy: MySQL + Hosting**

**Oleh Ary Budi Warsito**

1. Pendahuluan

Dibawah ini adalah panduan bagaimana mengimplementasikan FastAPI dengan database MYSQL setalah itu dapat di develop ke hosting. Untuk koneksi ke mysql saya menggunakan SQL toolkit yaitu mysql-connector-python. Untuk server mysql bisa gunakan xampp atau laragon. Dalam hal ini saya menggunakan Laragon.

1. Step 1 : Instalasi FastAPI di python

Silakan lakukan instalasi dengan melakukan ketik command berikut:

|  |
| --- |
| 1. pip install fastapi uvicorn sqlalchemy mysqlclient pymysql pydantic |

1. Folder setup:

Pada folder ini tergantung dari tabel yang akan dibuat. Jika ada banyak tabel silakan perhatikan bagian file pada folder controller dan file pada folder model. Dalam hal pembuatan ini saya membuat 2 tabel yaitu : customer dan user.

A screenshot of a computer program

AI-generated content may be incorrect.

1. Buka database configuration (db/database.py)

Pada file berkas ini digunakan untuk melakukan koneksi ke dalam database. Silakan copy kan. Untuk user dan pass disesuaikan dengan yang ada di dalam user database. Default laragon adalah user : root dan password kosong.

|  |
| --- |
| 1. import mysql.connector  2.  3. # Connect to the database  4. mydb = mysql.connector.connect(  5.     host="localhost",  6.     user="root",  7.     password="",  8.     database="latihan\_fastapi"  9. )  10.  11. # Create a cursor object  12. cursor = mydb.cursor()  13. |

1. Buka model configuration ( model/customerModel.py)

Mendefiniskan model dengan yang ada tabel. Dalam hal ini adalah menggunakan **tabel** **customer** Untuk referensi types data bisa pada pydantic silakan merefer : <https://docs.pydantic.dev/1.10/usage/types/>

|  |
| --- |
| 1. from pydantic import BaseModel  2.  3. class TBCustomer(BaseModel):  4.     name: str  5.     email: str  6.     address: str  7. |

1. Buka Rest API respon atau rekap berkas ( utils/response.py)

Digunakan untuk mengkonversi data ke dalam bentuk json. Cukup buat sekali untuk digunakan berbagai model data.

|  |
| --- |
| 1. # utils/response.py  2. def api\_response(data=None, message="Success", status=True, error=None):  3. return {  4. "status": status,  5. "message": message,  6. "data": data,  7. "error": error,  8. } |

1. Buat controller configuration (controllers/customer\_controller.py)

Ini digunakans sebagai router. Jika ada yang lebih pada tabel silakan tambahkan.

|  |
| --- |
| 1. # controllers/user\_controller.py  2. from fastapi import APIRouter, HTTPException, status  3. import mysql.connector  4.  5. # koneksi mysql cek folder dv  6. from ..db.database import mydb, cursor  7. # model Tabel cek folder model  8. from ..model.CustomerModel import TBCustomer  9. # respon JSON cek folder utils  10. from ..utils.response import api\_response  11.  12. router = APIRouter()  13.  14. # CREATE  15. @router.post("/customer",status\_code=status.HTTP\_201\_CREATED)  16.  17. # Definisikan Customer model  18. def insert\_customer(customer: TBCustomer):  19.  20. insert\_query = """  21. INSERT INTO customers (name, email, address)  22. VALUES (%s, %s, %s)  23. """  24. values = (customer.name, customer.email, customer.address)  25.  26. try:  27. cursor.execute(insert\_query, values)  28. mydb.commit()  29. except mysql.connector.Error as err:  30. raise HTTPException(status\_code=400, detail=f"Error: {err}")  31.  32. return api\_response(message="customer created successfully")  33.  34. # READ All  35. @router.get("/customer",status\_code=status.HTTP\_302\_FOUND)  36. def select\_customer():  37. select\_query = "SELECT \* FROM customers"  38. cursor.execute(select\_query)  39. results = cursor.fetchall()  40. return api\_response(data=results, message="All customer retrieved")  41.  42. # READ Single  43. @router.get("/customer/{customer\_id}",status\_code=status.HTTP\_200\_OK)  44. def get\_customer\_by\_id(customer\_id: int):  45. select\_query = "SELECT \* FROM customers WHERE id = %s"  46. cursor.execute(select\_query, (customer\_id,))  47. result = cursor.fetchone()  48. if result is None:  49. raise HTTPException(status\_code=404, detail="customer not found")  50. return api\_response(data=result, message="customer retrieved successfully")  51.  52. # UPDATE  53. @router.put("/customer/{customer\_id}",status\_code=status.HTTP\_200\_OK)  54. def update\_customer(customer\_id: int, customer: TBCustomer):  55.  56. update\_query = """  57. UPDATE customers  58. SET name = %s, email = %s, address = %s  59. WHERE id = %s  60. """  61. values = (customer.name,customer.email, customer.address, customer\_id)  62.  63. cursor.execute(update\_query, values)  64. mydb.commit()  65. if cursor.rowcount == 0:  66. raise HTTPException(status\_code=404, detail="customer not found")  67. return api\_response(message="customer updated successfully")  68.  69. # DELETE user  70. @router.delete("/customer/{customer\_id}",status\_code=status.HTTP\_200\_OK)  71. def delete\_user(customer\_id: int):  72. delete\_query = "DELETE FROM users WHERE id = %s"  73. cursor.execute(delete\_query, (customer\_id,))  74. mydb.commit()  75. if cursor.rowcount == 0:  76. raise HTTPException(status\_code=404, detail="customer not found")  77. return api\_response(message="customer deleted successfully")  78. |

1. Registerkan controller di dalam main.py

Dalam hal ini saya registerkan customer didalam main.py

|  |
| --- |
| 1. # main.py  2. from fastapi import FastAPI  3. # controller cek folder controller  4. from .controllers.user\_controller import router as user\_router  5. from .controllers.customer\_controller import router as customer\_router  6.  7. tags\_metadata = [  8. {"name": "Pengguna","description": "Crud untuk pengguna / user. "},  9. {"name":"Customers","description": "Crud untuk pengguna Customers. "},  10. {"name": "Mahasiswa","description": "CRUD data mahasiswa."}  11. ]  12. app = FastAPI(  13. title="ABWarsito API",  14. version="0.0.1",  15. contact={  16. "name": "AbWarsito",  17. "url": "http://abwarsito.my.id/",  18. "email": "ariebhewhe@gmail.com",  19. },  20. license\_info={  21. "name": "Apache 2.0",  22. "identifier": "MIT",  23. },  24. swagger\_ui\_parameters={"defaultModelsExpandDepth": -1}  25. )  26. # contoh : app.include\_router(customer\_router, prefix="/api", tags=["Customers"])  28. app.include\_router(customer\_router, tags=["Customers"])  29.  30. @app.get("/")  31. def root():  32. return {"message": "Selamat data di latihan webservice Mobile Prograaming"}  33. |

1. Buat database dengan nama latihan\_fastapi

A screenshot of a computer

AI-generated content may be incorrect.

Dengan tabel customer

A screenshot of a computer

AI-generated content may be incorrect.

1. Jalankan server laragon dan fast api application uvicorn

Menjalan server laragon pilih start, maka akan mengaktifkan server mysql.

A screenshot of a computer

AI-generated content may be incorrect.

Jalankan python dengan posisi sperti pada di gambar berikut :

python -m uvicorn app.main:app –reload

A screenshot of a computer screen

AI-generated content may be incorrect.

Keterangan :

**app**.main: app dimaksudkan untuk app adalah nama folder app, sedangkan main untuk nama main.py yang dijalankan adalah function app

A screen shot of a computer

AI-generated content may be incorrect.

1. Testing API

<http://127.0.0.1:8000/docs>

A screenshot of a computer

AI-generated content may be incorrect.

1. Silakaan lakukan uji coba