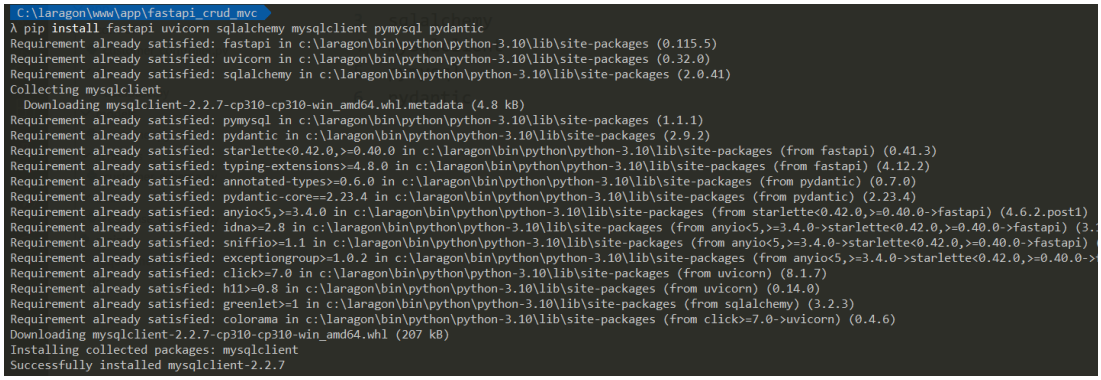
Panduann Setup FastAPI & SQLAlchemy: MySQL + Hosting

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.
2. Step 1 : Instalasi FastAPI di python
   Silakan lakukan instalasi dengan melakukan ketik command berikut:

```
1. pip install fastapi uvicorn sqlalchemy mysqlclient pymysql pydantic
```



```
C:\laragon\www\app\fastapi_crud_mvc
λ pip install fastapi uvicorn sqlalchemy mysqlclient pymysql pydantic
Requirement already satisfied: fastapi in c:\laragon\bin\python\python-3.10\lib\site-packages (0.115.5)
Requirement already satisfied: uvicorn in c:\laragon\bin\python\python-3.10\lib\site-packages (0.32.0)
Requirement already satisfied: sqlalchemy in c:\laragon\bin\python\python-3.10\lib\site-packages (2.0.41)
Collecting mysqlclient
  Downloading mysqlclient-2.2.7-cp310-cp310-win_amd64.whl.metadata (4.8 kB)
Requirement already satisfied: pymysql in c:\laragon\bin\python\python-3.10\lib\site-packages (1.1.1)
Requirement already satisfied: pydantic in c:\laragon\bin\python\python-3.10\lib\site-packages (2.9.2)
Requirement already satisfied: starlette<0.42.0,>=0.40.0 in c:\laragon\bin\python\python-3.10\lib\site-packages (from fastapi) (0.41.3)
Requirement already satisfied: typing-extensions>=4.8.0 in c:\laragon\bin\python\python-3.10\lib\site-packages (from fastapi) (4.12.2)
Requirement already satisfied: annotated-types>=0.6.0 in c:\laragon\bin\python\python-3.10\lib\site-packages (from pydantic) (0.7.0)
Requirement already satisfied: pydantic-core==2.23.4 in c:\laragon\bin\python\python-3.10\lib\site-packages (from pydantic) (2.23.4)
Requirement already satisfied: anyio<5,>=3.4.0 in c:\laragon\bin\python\python-3.10\lib\site-packages (from starlette<0.42.0,>=0.40.0->fastapi) (4.6.2.post1)
Requirement already satisfied: idna>=2.8 in c:\laragon\bin\python\python-3.10\lib\site-packages (from anyio<5,>=3.4.0->starlette<0.42.0,>=0.40.0->fastapi) (3.
Requirement already satisfied: sniffio>=1.1 in c:\laragon\bin\python\python-3.10\lib\site-packages (from anyio<5,>=3.4.0->starlette<0.42.0,>=0.40.0->fastapi)
Requirement already satisfied: exceptiongroup>=1.0.2 in c:\laragon\bin\python\python-3.10\lib\site-packages (from anyio<5,>=3.4.0->starlette<0.42.0,>=0.40.0->
Requirement already satisfied: click>=7.0 in c:\laragon\bin\python\python-3.10\lib\site-packages (from uvicorn) (8.1.7)
Requirement already satisfied: h11>=0.8 in c:\laragon\bin\python\python-3.10\lib\site-packages (from uvicorn) (0.14.0)
Requirement already satisfied: greenlet>=1 in c:\laragon\bin\python\python-3.10\lib\site-packages (from sqlalchemy) (3.2.3)
Requirement already satisfied: colorama in c:\laragon\bin\python\python-3.10\lib\site-packages (from click>=7.0->uvicorn) (0.4.6)
Downloading mysqlclient-2.2.7-cp310-cp310-win_amd64.whl (207 kB)
Installing collected packages: mysqlclient
Successfully installed mysqlclient-2.2.7
```

3. 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.

```
.
├── app
│   ├── controllers
│   │   ├── customer_controller.py
│   │   └── user_controller.py
│   ├── db
│   │   └── database.py
│   ├── model
│   │   ├── CustomerModel.py
│   │   └── UserModel.py
│   ├── utils
│   │   └── response.py
│   └── main.py
├── doc
│   ├── berkas_pip
│   │   └── get-pip.py
│   └── FastApi_Mysql_Hosting.docx
└── requirements.txt
```

4. 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.

```python
import mysql.connector

# Connect to the database
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="latihan_fastapi"
)

# Create a cursor object
cursor = mydb.cursor()

```

5. 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/

```python
from pydantic import BaseModel

class TBCustomer(BaseModel):
    name: str
    email: str
    address: str

```

6. 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.     }
```

7. 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.
```

8. 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.
```

9. Buat database dengan nama `latihan_fastapi`

Database: latihan_fastapi

Alter database Database schema Privileges

Tables and views

Search data in tables (1)

Search

	Table	Engine?	Collation?	Data Length?	Index Length?	Data Free?	Auto Increment?	Rows?	Comment?
☑	**customers**	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0	1	0	
☐	**users**	InnoDB	utf8mb4_0900_ai_ci	16,384	0	0	5	~ 3	
	2 in total	InnoDB	utf8mb4_0900_ai_ci	32,768	0	0			

Dengan tabel customer

Table: customers

Select data **Show structure** Alter table New item

Column	Type	Comment
id	int *Auto Increment*	
name	varchar(100)	
email	varchar(100) *NULL*	
address	varchar(255) *NULL*	

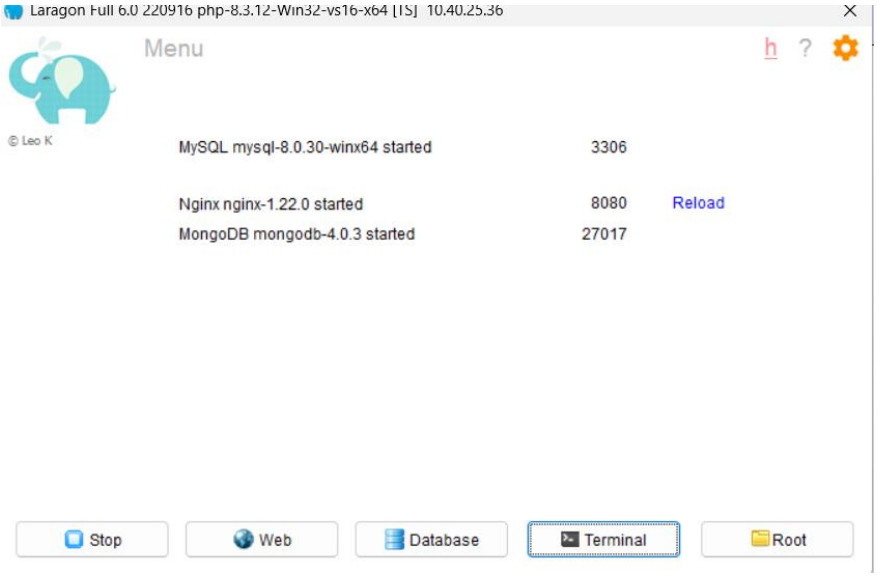
Indexes

PRIMARY	*id*

Alter indexes

10. Jalankan server laragon dan fast api application uvicorn

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



Jalankan python dengan posisi sperti pada di gambar berikut :

python -m uvicorn app.main:app –reload

```
 Directory of C:\laragon\www\app\fastapi

19/06/2025  13:58    <DIR>          .
18/06/2025  22:29    <DIR>          ..
19/06/2025  14:08    <DIR>          app
19/06/2025  14:10    <DIR>          doc
19/06/2025  14:00               227 requirements.txt
               1 File(s)            227 bytes
               4 Dir(s)  218.114.199.552 bytes free

 C:\laragon\www\app\fastapi
λ python -m uvicorn app.main:app --reload
```

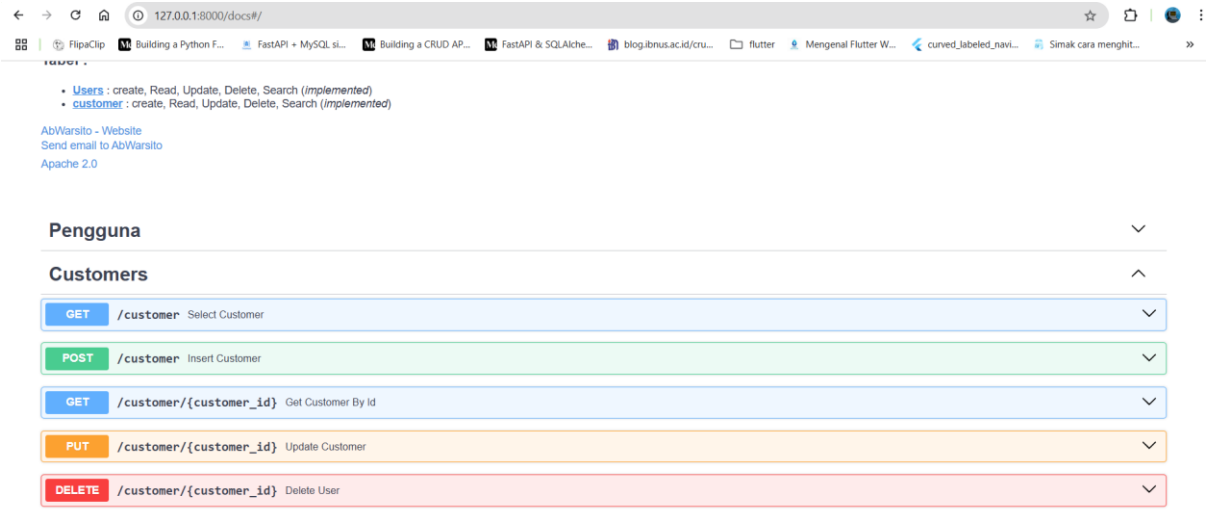
Keterangan :

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

```
 C:\laragon\www\app\fastapi
λ python -m uvicorn app.main:app --reload
INFO:     Will watch for changes in these directories: ['C:\\laragon\\www\\app\\fastapi']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [3132] using WatchFiles
INFO:     Started server process [372]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

11. Testing API

    http://127.0.0.1:8000/docs



12. Silakaan lakukan uji coba