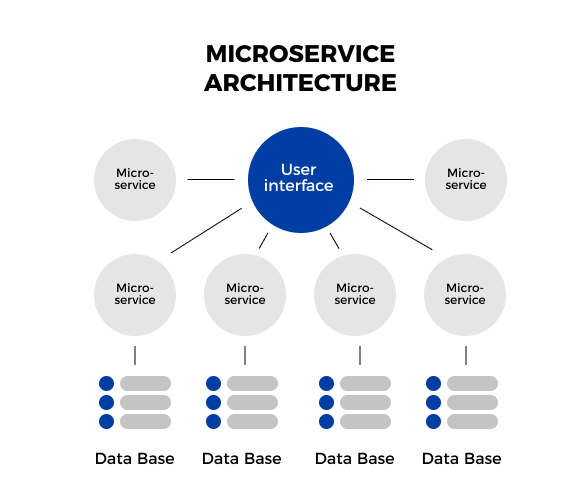
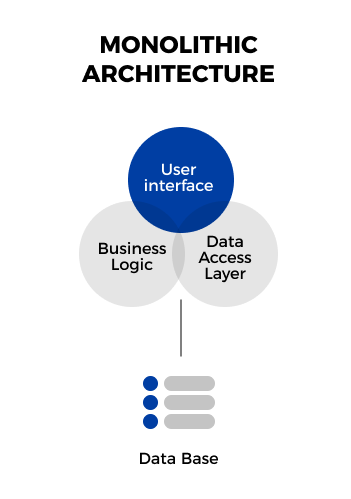
**Backend Microservices Python Documentation**

**Monolithic vs Microservice**



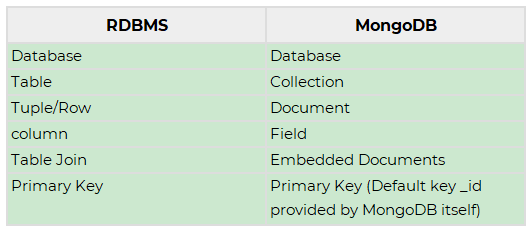
Arsitektur monolitik biasanya digunakan saat sebuah perusahaan atau aplikasi pertama kali mulai karena kemudahan dalam pembuatan dan manajemen. Pada arsitektur monolitik, jika ada komponen program yang harus diperbarui, seluruh aplikasi harus ditulis ulang.

Berbeda dengan arsitektur monolitik, arsitektur microservices dapat memiliki basisdata terpisah yang independen dari microservice lainnya. Sehingga, pengembangan pada basisdata pun bisa dilakukan dengan lebih aman sesuai kebutuhan. Pada aplikasi microservices, setiap modul terpisah dan dapat diubah tanpa mempengaruhi bagian lain dari program.

Dapat disimpulkan, beberapa kelebihan dari arsitektur microservices adalah:

1. Continuous Integration and Deployment (CI/CD): Microservices dapat dikelola oleh beberapa tim kecil yang memberikan kebebasan mereka untuk mengubah, menghapus, atau menambahkan kode tanpa mengganggu service lainnya.
2. Bebas menggunakan bahasa pemograman, Framework, dan basisdata apapun. Bahkan berbeda satu dan lainnya diperbolehkan
3. Kontainerisasi (misal menggunakan docker)
4. Scalibility, ketersediaan, dan lebih tahan pada perubahan. Scaling pada arsitektur monolitik tidak efisien karena programmer harus melakukan scaling seluruh aplikasi alih-alih hanya melakukan scaling pada fungsi atau layanan individual.

**Seputar NoSQL, pada project ini menggunakan MongoDB**



setiap kolom dijelaskan oleh field, lalu dokumen merepresentasikan baris, dan collection merupakan tabel dimana dokumen dikumpulkan.

Misalkan kita coba mengacu pada saat kita belajar mysql maka database kita akan menjadi,

database : rentalfilm

collection : customers

field : id, username(namaid), namalengkap, email

**CRUD pada MySQL**

1. Create data:

INSERT INTO table(c1,c2,...)

VALUES (v1,v2,...);

Contoh:

INSERT INTO customers (NAMAID, NAMALENGKAP, EMAIL)

VALUES ("Astrid", "Astrid Gruber", "astrid.gruber@apple.at");

Insert banyak data:

INSERT INTO table(kolom1,kolom2,...)

VALUES

(value1\_1,value1\_2,...),

...

(valuen\_1,valuen\_2,...);

Contoh:

INSERT INTO customers (NAMAID, NAMALENGKAP, EMAIL)

values

("Fernanda","Fernanda Ramos","fernadaramos4@uol.com.br"),

("Mark","Mark Philips","mphilips12@shaw.ca"),

("Jennifer","Jennifer Peterson","jenniferp@rogers.ca");

1. Update data:

UPDATE [LOW\_PRIORITY] [IGNORE] table\_name

SET

column\_name1 = expr1,

column\_name2 = expr2,

...

WHERE

condition;

Contoh:

UPDATE customers

SET

NAMALENGKAP = "Jennifer Lauren",

EMAIL = "jenniferl@rogers.ca"

WHERE

ID = 4;

1. Delete data:

DELETE FROM table\_name

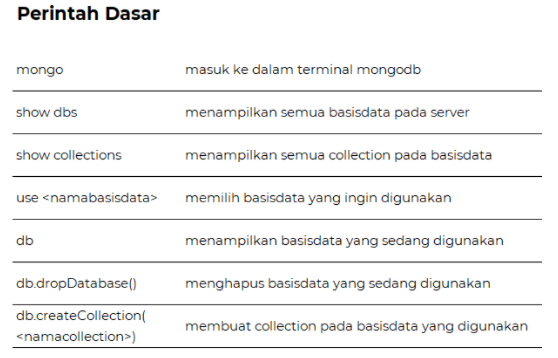
WHERE condition;

Contoh:

DELETE FROM customers

WHERE ID = 4;

**CRUD pada MongoDB**



1. Insert data:

db.<namacollection>.insert(<document>)

Contoh:

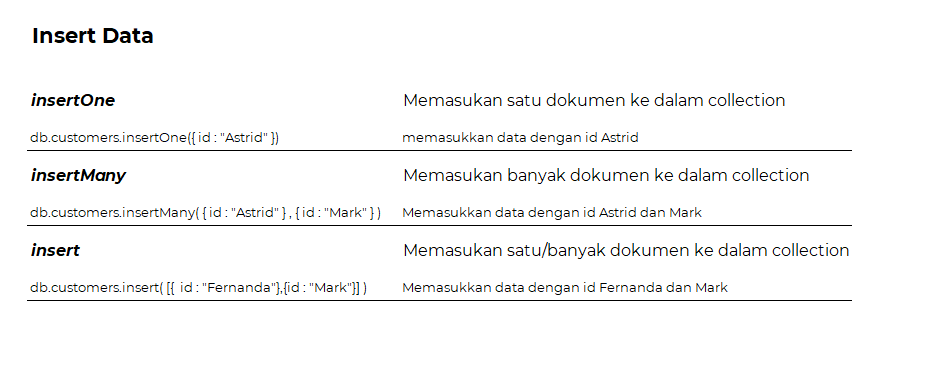
db.customers.insert({

id : "Astrid",

fullname : "Astrid Gruber",

email : "astrid.gruber@apple.at"

})



1. Update data:

db.<namacollection>.update({<kriteria>}, {$set:{<nilai baru>}})

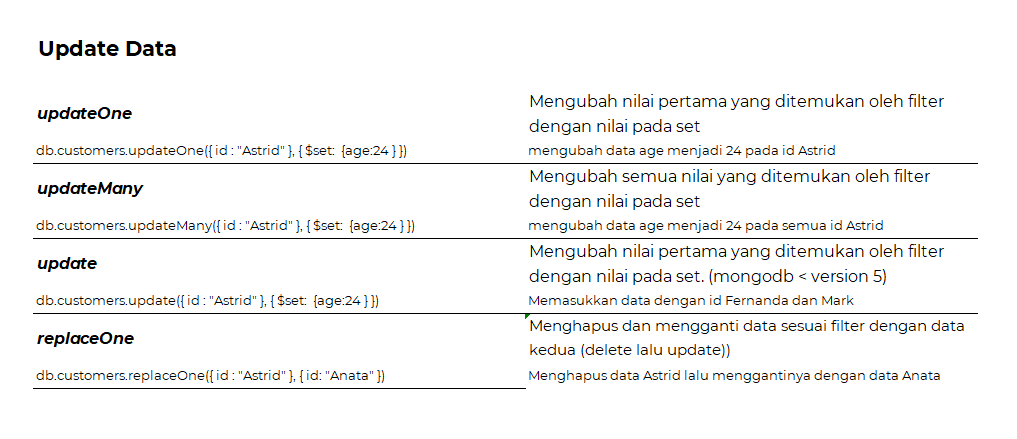
Contoh:

db.customers.update(

{id:"Jennifer"},

{$set:{fullname:'Jennifer Aniston'}}

)



1. Delete data:

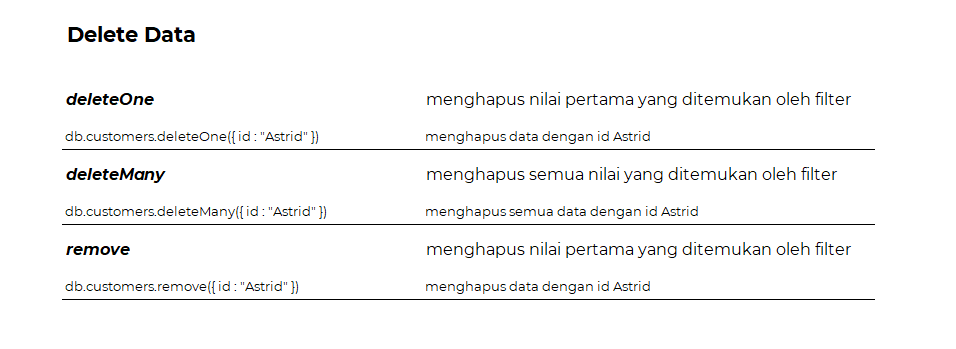
db.<nama collection>.remove(<query>)

Contoh:

db.customers.remove(

{id:"Jennifer"}

)

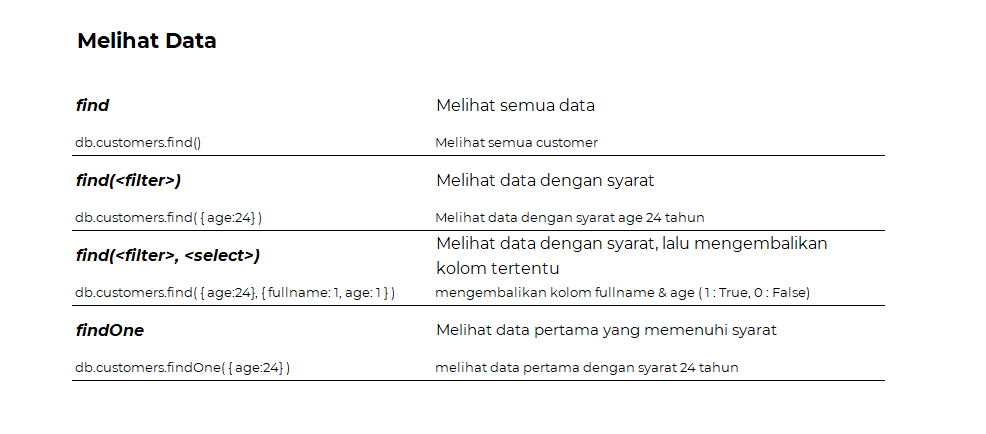


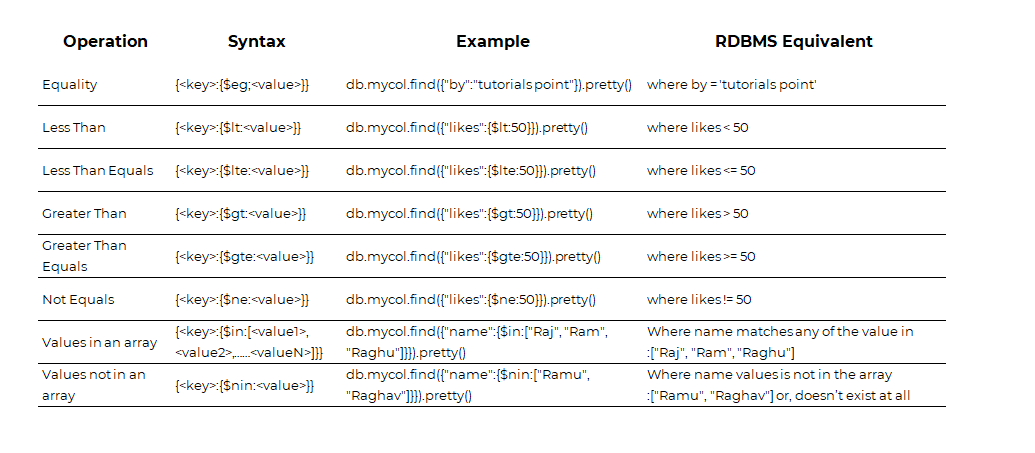
1. Query data: find

db.<nama collection>.find()

Contoh:

db.customers.find( {age:24} ).pretty()





Kita juga bisa mencari menggunakan regex misalnya kita ingin mencari orang yang mempunyai nama award yang di awali "Rosing".

db.bios.find({"awards.award" : {$regex: /^Rosing/}})

**Membuat python virtual environment**

Agar setiap project dapat memiliki library khusus

1. Buka cmd, ketik:

pip install virtualenv

1. Untuk membuat virtual environment baru kita perlu menuliskan, pastikan kita berada di dalam folder dimana kita ingin mengaktifkan virtual environment tersebut:

virtualenv env

1. Jika kita cek akan ada folder baru yang menyimpan virtual environment kita yang diberi nama env. Isi dari folder tersebut adalah:

Lib/ pyvenv.cfg Scripts/

1. Menggunakan VirtualEnvWrapper, agar lebih mudah. Install dulu:

pip to install virtualenvwrapper-win

1. Buat virtual environment kita:

‘mkvirtualenv HelloWold’

menariknya adalah saat menggunakan tambahan virtualEnvWrapper, virtualenv yang kita buat tidak berada di folder kerja kita tapi berada di folder user kita.

1. sambungkan folder kerja kita dengan virtualenv yang kita buat. Gunanya adalah agar saat kita mengaktifkan virtualenv maka kita akan otomatis berpindah ke folder kerja kita:

setprojectdir .

1. Untuk menyalakan kembali virtualenv kita kita dapat menggunakan perintah workon:

workon 'nama\_env'

**Python-MySQL tanpa ORM**

CRUD MySQL database dengan python tanpa ORM

1. Buka cmd, aktifkan python virtual environment:

workon ms\_python

‘ms\_python’ adalah nama virtual environmentnya

Pada visual studi, ketik ctrl+shift+p dan arahkan interpreter ke ‘ms\_python’

1. Install pada virtual env:

pip install mysql-connector-python

1. Arahkan cmd pada folder projek pada terminal:

cd C:\Users\ASUS\Desktop\github\microservices-python\Without ORM\python mysql

1. Jalankan MySQL server pada XAMPP, sesuaikan host, user, passwordnya

Buat database pada MySQL dengan nama table dan kolom2 yang sesuai

1. Buat projek CRUD python MySQL
2. Jalankan file app.py pada terminal:

app.py

**Python-MySQL dengan ORM**

CRUD MySQL database dengan python dengan ORM

1. Buka cmd, aktifkan python virtual environment:

workon ms\_python

‘ms\_python’ adalah nama virtual environmentnya

Pada visual studi, ketik ctrl+shift+p dan arahkan interpreter ke ‘ms\_python’

1. Install pada virtual env:

pip install mysql-connector-python

1. Arahkan cmd pada folder projek pada terminal:

cd C:\Users\ASUS\Desktop\github\microservices-python\With ORM\python mysql

1. Jalankan MySQL server pada XAMPP, sesuaikan host, user, passwordnya

Buat database pada MySQL dengan nama table dan kolom2 yang sesuai

1. Buat projek CRUD python MySQL
2. Jalankan file app.py pada terminal:

app.py

**Python-MongoDB tanpa ODM**

CRUD MongoDB database dengan python tanpa ODM

1. Buka cmd, aktifkan python virtual environment:

workon ms\_python

‘ms\_python’ adalah nama virtual environmentnya

Pada visual studi, ketik ctrl+shift+p dan arahkan interpreter ke ‘ms\_python’

1. Install pada virtual env:

pip install pymongo

1. Arahkan cmd pada folder projek pada terminal:

cd C:\Users\ASUS\Desktop\github\microservices-python\Without ORM\python mongodb

1. Jalankan MongoDB server pada MongoDB Compass, sesuaikan host dan port

Buat database pada MongoDB dengan nama collection (table) yang sesuai

1. Buat projek CRUD python MongoDB
2. Jalankan file app.py pada terminal:

app.py

**Python-MongoDB dengan ODM**

CRUD MongoDB database dengan python dengan ODM

1. Buka cmd, aktifkan python virtual environment:

workon ms\_python

‘ms\_python’ adalah nama virtual environmentnya

Pada visual studi, ketik ctrl+shift+p dan arahkan interpreter ke ‘ms\_python’

1. Install pada virtual env:

pip install mongoengine

1. Arahkan cmd pada folder projek pada terminal:

cd C:\Users\ASUS\Desktop\github\microservices-python\With ORM\python mongodb

1. Jalankan MongoDB server pada MongoDB Compass, sesuaikan host dan port

Buat database pada MongoDB dengan nama collection (table) yang sesuai

1. Buat projek CRUD python MongoDB
2. Jalankan file app.py pada terminal:

app.py

**Python web development dengan Flask**

CRUD MySQL database dengan python dengan Flask web development

1. Buka cmd, aktifkan python virtual environment:

workon ms\_python

‘ms\_python’ adalah nama virtual environmentnya

Pada visual studi, ketik ctrl+shift+p dan arahkan interpreter ke ‘ms\_python’

1. Install pada virtual env:

pip install flask

1. Buat projek CRUD python MySQL berdasarkan REST API

Buat routes

Gunakan POSTMAN

url default: localhost:5000

1. Jalankan file app.py pada terminal:

app.py

**Python web development dengan FastAPI**

CRUD MongoDB database dengan python dengan FastAPI web development. Jika Flask sendiri sudah merupakan microframework yang lebih ringan dibanding Django, maka FastAPI lebih dibandingkan Flask. FastAPI adalah web framework yang dikhususkan untuk pembuatan API yang diklaim memiliki kecepatan dan penulisan kode yang modern.

1. Buka cmd, aktifkan python virtual environment:

workon ms\_python

‘ms\_python’ adalah nama virtual environmentnya

Pada visual studi, ketik ctrl+shift+p dan arahkan interpreter ke ‘ms\_python’

1. Install pada virtual env:

pip install fastapi

pip install uvicorn 🡪 untuk server

1. Buat projek CRUD python MySQL berdasarkan FastAPI

Buat routes

Gunakan POSTMAN

url default: localhost:8000

1. Jalankan file app.py pada terminal:

app.py

1. Start server, dan agar terus run ketika ada perubahan, pada terminal jalankan:

uvicorn main:app --reload

**Keamanan Endpoint API dengan JWT**

JWT adalah kependekan dari JSON Web Token. JWT adalah sebuah token berbentuk string panjang yang digunakan sebagai sarana auth. JWT yang disimpan pada cookies browser. JWT disusun berdasarkan: Header (terdiri dari Algoritma dan type jwt yang dipakai), Payload (data yang ingin kita kirimkan atau sebagai acuan saat aplikasi server melakukan encode), dan Verify Signature (hasil dari Hash atau gabungan dari isi encode Header dan Payloadnya lalu ditambahkan kode secretnya).

1. Buka cmd, aktifkan python virtual environment:

workon ms\_python

‘ms\_python’ adalah nama virtual environmentnya

Pada visual studi, ketik ctrl+shift+p dan arahkan interpreter ke ‘ms\_python’

1. Install pada virtual env:

pip install flask-jwt-extended

1. Buat controller yang berkaitan dengan token, isi dengan fungsi create\_access\_token
2. Request token:

localhost:5000/requesttoken (POST)

body:

{

"email": “rudi.pertest@gmail.com”

}

Copy tokennya

localhost:5000/users (POST)

Auth bearer: paste tokennya

localhost:5000/user (POST)

body:

{

"userid": 1

}

Auth bearer: paste tokennya