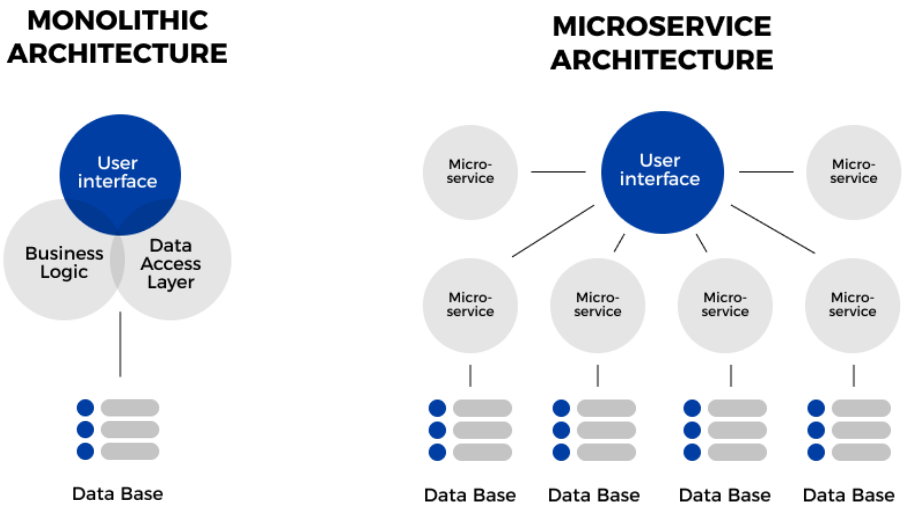
Backend Microservices Python Documentation

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

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)

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");

2. 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;
```

3. Delete data:

```
DELETE FROM table_name
WHERE condition;
```

Contoh:

```
DELETE FROM customers
WHERE ID = 4;
```

CRUD pada MongoDB

Perintah Dasar

mongo	masuk ke dalam terminal mongodb
show dbs	menampilkan semua basisdata pada server
show collections	menampilkan semua collection pada basisdata
use <namabasisdata>	memilih basisdata yang ingin digunakan
db	menampilkan basisdata yang sedang digunakan
db.dropDatabase()	menghapus basisdata yang sedang digunakan
db.createCollection(<namacollection>)	membuat collection pada basisdata yang digunakan

4. Insert data:
   db.<namacollection>.insert(<document>)
   Contoh:

```
db.customers.insert({
    id : "Astrid",
    fullname : "Astrid Gruber",
    email : "astrid.gruber@apple.at"
})
```

Insert Data

insertOne	Memasukan satu dokumen ke dalam collection
db.customers.insertOne({ id : "Astrid" })	memasukkan data dengan id Astrid
insertMany	Memasukan banyak dokumen ke dalam collection
db.customers.insertMany({ id : "Astrid" } , { id : "Mark" })	Memasukkan data dengan id Astrid dan Mark
insert	Memasukan satu/banyak dokumen ke dalam collection
db.customers.insert([{ id : "Fernanda"},{id : "Mark"}])	Memasukkan data dengan id Fernanda dan Mark

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

```
db.customers.update(
    {id:"Jennifer"},
    {$set:{fullname:'Jennifer Aniston'}}
```

)

Update Data

updateOne	Mengubah nilai pertama yang ditemukan oleh filter dengan nilai pada set
db.customers.updateOne({ id : "Astrid" }, { $set: {age:24 } })	mengubah data age menjadi 24 pada id Astrid
updateMany	Mengubah semua nilai yang ditemukan oleh filter dengan nilai pada set
db.customers.updateMany({ id : "Astrid" }, { $set: {age:24 } })	mengubah data age menjadi 24 pada semua id Astrid
update	Mengubah nilai pertama yang ditemukan oleh filter dengan nilai pada set. (mongodb < version 5)
db.customers.update({ id : "Astrid" }, { $set: {age:24 } })	Memasukkan data dengan id Fernanda dan Mark
replaceOne	Menghapus dan mengganti data sesuai filter dengan data kedua (delete lalu update))
db.customers.replaceOne({ id : "Astrid" }, { id: "Anata" })	Menghapus data Astrid lalu menggantinya dengan data Anata

6. Delete data:
   db.<nama collection>.remove(<query>)
   Contoh:
   db.customers.remove(
   {id:"Jennifer"}
)

Delete Data

deleteOne	menghapus nilai pertama yang ditemukan oleh filter
db.customers.deleteOne({ id : "Astrid" })	menghapus data dengan id Astrid
deleteMany	menghapus semua nilai yang ditemukan oleh filter
db.customers.deleteMany({ id : "Astrid" })	menghapus semua data dengan id Astrid
remove	menghapus nilai pertama yang ditemukan oleh filter
db.customers.remove({ id : "Astrid" })	menghapus data dengan id Astrid

7. Query data: find
   db.<nama collection>.find()
   Contoh:
   db.customers.find({age:24}).pretty()

Melihat Data

find	Melihat semua data
db.customers.find()	Melihat semua customer
find(<filter>)	Melihat data dengan syarat
db.customers.find({ age:24})	Melihat data dengan syarat age 24 tahun
find(<filter>, <select>)	Melihat data dengan syarat, lalu mengembalikan kolom tertentu
db.customers.find({ age:24}, { fullname: 1, age: 1 })	mengembalikan kolom fullname & age (1 : True, 0 : False)
findOne	Melihat data pertama yang memenuhi syarat
db.customers.findOne({ age:24})	melihat data pertama dengan syarat 24 tahun

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:{$eg;<value>}}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{$lt:<value>}}	db.mycol.find({"likes":{$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{$lte:<value>}}	db.mycol.find({"likes":{$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{$gt:<value>}}	db.mycol.find({"likes":{$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{$gte:<value>}}	db.mycol.find({"likes":{$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{$ne:<value>}}	db.mycol.find({"likes":{$ne:50}}).pretty()	where likes != 50
Values in an array	{<key>:{$in:[<value1>, <value2>,.....<valueN>]}}	db.mycol.find({"name":{$in:["Raj", "Ram", "Raghu"]}}).pretty()	Where name matches any of the value in :["Raj", "Ram", "Raghu"]
Values not in an array	{<key>:{$nin:<value>}}	db.mycol.find({"name":{$nin:["Ramu", "Raghav"]}}).pretty()	Where name values is not in the array :["Ramu", "Raghav"] or, doesn't exist at all

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

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

3. 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/

4. Menggunakan VirtualEnvWrapper, agar lebih mudah. Install dulu:
   pip to install virtualenvwrapper-win

5. 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.

6. 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 .

7. 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'

2. Install pada virtual env:
   pip install mysql-connector-python

3. Arahkan cmd pada folder projek pada terminal:
   cd C:\Users\ASUS\Desktop\github\microservices-python\Without ORM\python mysql

4. Jalankan MySQL server pada XAMPP, sesuaikan host, user, passwordnya
   Buat database pada MySQL dengan nama table dan kolom2 yang sesuai

5. Buat projek CRUD python MySQL

6. 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'

2. Install pada virtual env:
   pip install mysql-connector-python

3. Arahkan cmd pada folder projek pada terminal:
   cd C:\Users\ASUS\Desktop\github\microservices-python\With ORM\python mysql

4. Jalankan MySQL server pada XAMPP, sesuaikan host, user, passwordnya
   Buat database pada MySQL dengan nama table dan kolom2 yang sesuai

5. Buat projek CRUD python MySQL

6. 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'

2. Install pada virtual env:
   pip install pymongo

3. Arahkan cmd pada folder projek pada terminal:
   cd C:\Users\ASUS\Desktop\github\microservices-python\Without ORM\python mongodb

4. Jalankan MongoDB server pada MongoDB Compass, sesuaikan host dan port
   Buat database pada MongoDB dengan nama collection (table) yang sesuai

5. Buat projek CRUD python MongoDB

6. 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'

2. Install pada virtual env:
   pip install mongoengine

3. Arahkan cmd pada folder projek pada terminal:
   cd C:\Users\ASUS\Desktop\github\microservices-python\With ORM\python mongodb

4. Jalankan MongoDB server pada MongoDB Compass, sesuaikan host dan port
   Buat database pada MongoDB dengan nama collection (table) yang sesuai

5. Buat projek CRUD python MongoDB

6. 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'

2. Install pada virtual env:
   pip install flask

3. Buat projek CRUD python MySQL berdasarkan REST API
   Buat routes
   Gunakan POSTMAN
   url default: localhost:5000

4. 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'

2. Install pada virtual env:
   pip install fastapi
   pip install uvicorn → untuk server

3. Buat projek CRUD python MySQL berdasarkan FastAPI
   Buat routes
   Gunakan POSTMAN
   url default: localhost:8000

4. Jalankan file app.py pada terminal:
   app.py

5. 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'

2. Install pada virtual env:
   pip install flask-jwt-extended

3. Buat controller yang berkaitan dengan token, isi dengan fungsi create_access_token

4. 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