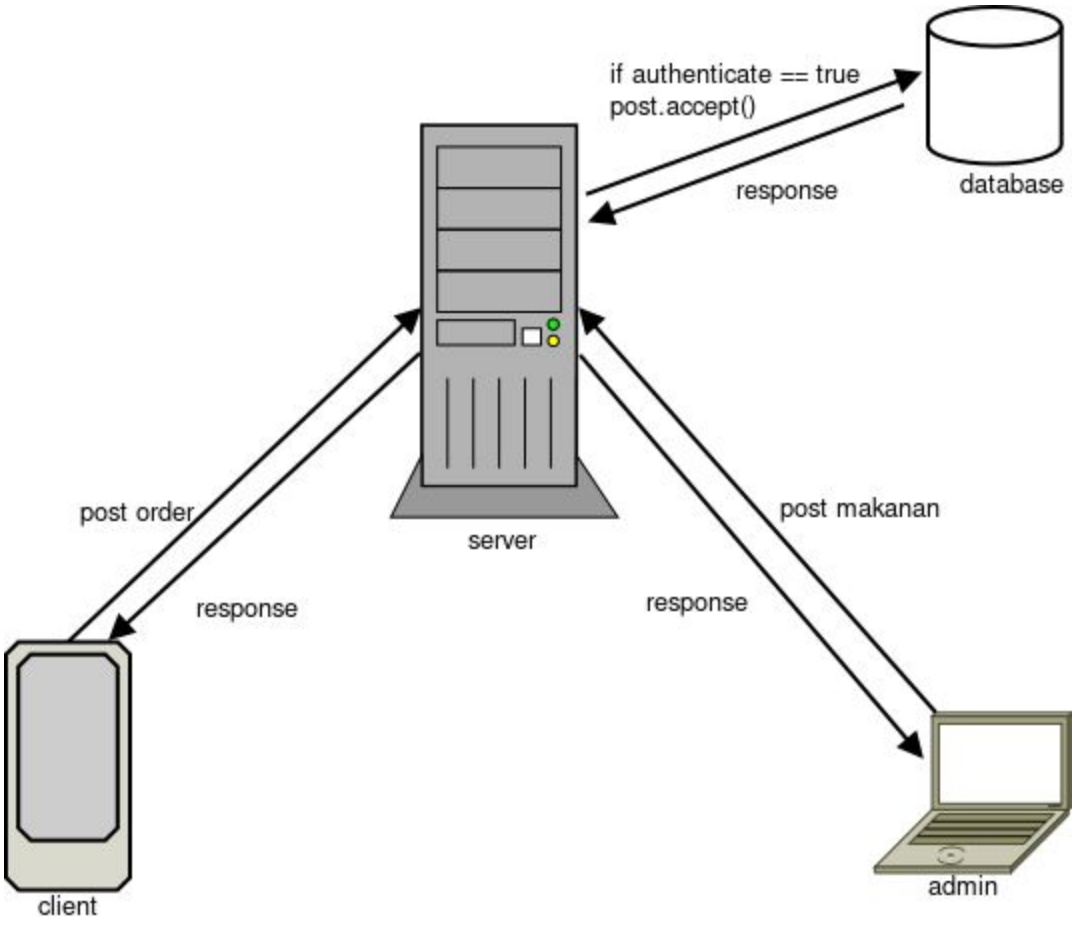
desain aplikasi antar makanan

dalam hal ini menggunakan design pattern oop
ada 3 objek penting yang berperan dalam sistem ini
1. makanan
2. consumen
3. order

untuk flow



admin menginput data makanan, lalu consumen bisa memesan makanan dengan login terlebih dahulu/register bila belum punya akun (makanan yang bisa dipesan adalah makanan yang tersedia stocknya), pesanan disimpan dalam order(bila order terkirim maka sukses dan penggunan bisa membatalkan dengan catatan <= 10 menit dari waktu order), untuk makanan disimpan dalam data makanan, dan consumen disimpan dalam data consumen serta order disimpan dalam data order lalu ada data admin yang bertugas menginput data makanan serta dapat melakukan edit pada data consumen dan order

*stack backend
- sistem operasi linux(dalam hal ini dengan arch linux), desain api serta websocket dengan nodejs, web server nginx(sebagai reverse proxy) dan untuk action
1. makanan(crud)admin
2. consumen(c)client, (crud)admin
3. order(c)client, (crud)admin

* data
a. tabel makanan

1. kode(char, primary key)
2. nama(varchar, notnull)
3. harga(int, notnull)
4. quantity(int)

b. tabel consumen
1. kode(char, primary key)
2. nama(varchar, notnull)
3. telepon(int, notnull)
4. alamat(text)

c. tabel order
1. id(int, primary key, autoincrement)
2. kode_m(fk tabel makanan)
3. kode_c(fk tabel consumen)
4. status(enum('terkirim','batal'))

* micro service
bakcend(nodejs dengan web socket)
database(postgresql)
frontend mobile(bisa android atau ios)

*alasan memilih design pattern oop karena ini bisa diklasifikasikan sebagai objek(maksudnya yang bertinteraksi dalam sistem ini) serta untuk pengembangan lebih mudah karena sudah dibagi menjadi objek serta menggunakan web socket sebagai komunikasi antara client dan server karena data bersifat realtime

* untuk tools yang digunakan(vscode untuk coding, httpie/postman untuk untuk test api, pgadmin4 untuk manajemen database, browser untuk test websocket, apiary untuk testing api awal)

penanganan keamanan
menurut saya bisa menggunakan token, karena token adalah salah satu otentikasi yang digunakan dalam arsitektur micro service dan dalam aplikasi ini yang bisa melakukan order adalah consumen yang sudah teregistrasi

* restful api(source code)

* menurut saya perlu ditambahkan action logout dan blacklist token(ketika user logout tetapi token masih aktif maka action ini akan bertindak untuk menghanguskan token karena token menggunakan waktu sebagai indikator aktif)