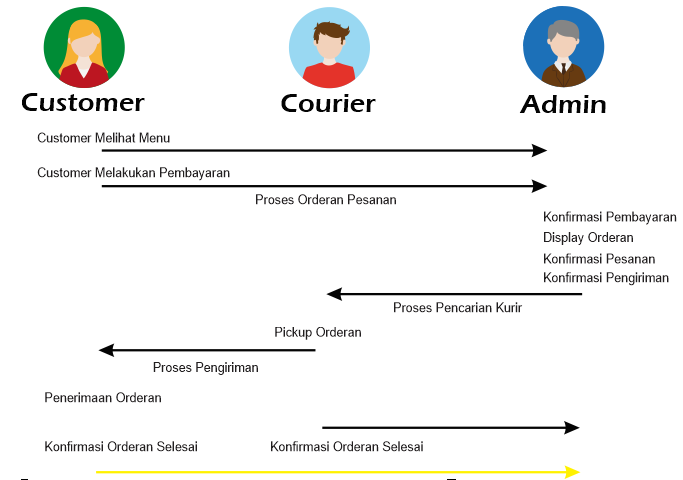
1. **Design *stack backend* dan *micro service* untuk *mobile apps* layanan antar makanan.**



Bisnis Proses Aplikasi Antar Makanan

1. Customer melihat menu yang ada di Admin(Restaurant)

2. Customer melakukan pembayaran via online

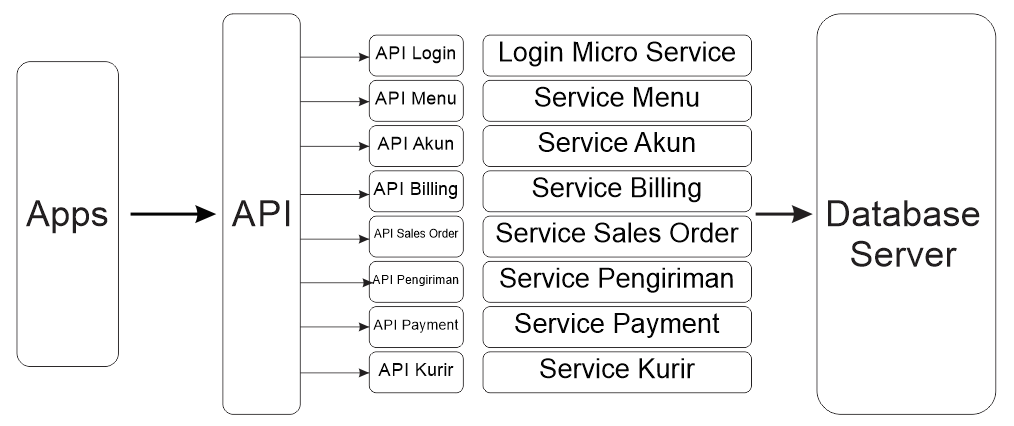
3. Admin melakukan konfirmasi pembayaran dari customer

4. Admin melakukan konfirmasi pesanan

5. Admin melakukan konfirmasi pengiriman

6. Kurir mengambil pesanan dan mengirimkan ke customer

7. Konfirmasi pesanan selesai dapat dari pihak customer ataupun dari kurir



Tools yang digunakan dalam pembuatan API tersebut adalah *Notepad++* dan *Postman. Notepad++* digunakan karena aplikasi tersebut dapat membuat file api yang dibutuhkan dan tidak terlalu ribet dalam penggunaannya. *Postman* digunakan untuk testing file api yang telah dibuat, apakah api yang dibuat jalan sesuai dengan kebutuhan ataupun tidak.

Web Server APACHE

BACKEND LANGUAGE PYTHON

FRAMEWORK DJANGO

DATABASE MYSQL / PHPMyAdmin

* 1. Web Server – Apache

*Application server* untuk *mobile apps* akan menangani konten yang dinamis seperti dalam logika bisnis proses. Dalam bisnis proses, aliran data antara *client side* dan *server side* berupa data, informasi dan pemanggilan *method*/*function*. Case mengenai *mobile apps* layanan antar makanan merupakan bisnis proses yang kompleks. Apache memiliki kemampuan dan kecepatan yang baik dalam menangani konten dinamis. Terdapat banyak modul Apache yang dikembangkan untuk menangani konten dinamis, salah satunya *multi-processing modules*(MPM). Kemampuan Apache untuk mempertahankan koneksi selalu terhubung antar *client side* dan *server side* membuat logika bisnis proses dapat berjalan tanpa terganggu atau terputus di tengah proses.

* 1. Backend Language – Python dan Framework – Django

Django merupakan *framework* berbasis Python. Python memiliki kelebihan yaitu *object-oriented programming* (OOP), *flexible*, *high security*, *good structure*, *efficient code*, integrasi yang baik dengan Apache dan PostgreSQL, serta optimal dalam mengembangkan aplikasi kompleks.

* 1. Database – PostgreSQL

Case mengenai *mobile apps* layanan antar makanan merupakan bisnis proses yang kompleks. Model *database* yang dibutuhkan adalah *relational database management system* (RDBMS). Hal tersebut dikarenakan dalam transaksi yang kompleks akan terdapat keterkaitan antar tabel yang tidak dapat dipisahkan. PostgreSQL merupakan *object-oriented* RDBMS yang sesuai digunakan dalam level enterprise. PostgreSQL memiliki beberapa karakter yang sesuai untuk membuat bisnis proses yang kompleks yaitu *highly programmable*, mampu menangani desain *database* yang kompleks, mendukung *custom* SQL *procedures*, *data integrity and reliability*, dan kemudahan integrasi antar *database systems* lainnya. PostgreSQL sendiri memiliki keamanan yang baik dengan implementasi *object level privileges using roles & user*, *kerberos*, *SSL communication*, dan *data level encryption*.

* 1. Database – MySQL

MySQL merupakan RDBMS yang mudah digunakan dan tidak kompleks. Walaupun MySQL tidak kompleks, namun memiliki banyak fitur yang dapat digunakan untuk mengembangkan sistem. MySQL sendiri memiliki sekuritas yang baik dan performa pemrosesan yang cepat. Hal tersebut memberikan keuntungan dalam merancang *catalog database* yang memilki desain bisnis proses sederhana dan perlu kecepatan proses.

* 1. Redis Cache

Redis merupakan penyimpanan *in-memory data structure*, yang dapat digunakan sebagai *database*, *cache* dan *message broker*. Redis mendukung penggunaan *data structure* seperti *strings*, *hashes*, *lists*, *sets*, *sorted sets with range queries*, *bitmaps*, *hyperloglogs* dan *geospatial indexes with radius queries*. Dari deskripsi tersebut dapat disimpulkan bahwa Redis Cache cocok untuk digunakan memproses *notification microservice*. Karena *notification* merupakan suatu proses bisnis sederhana untuk memberikan informasi ke user dan harus dilakukan dengan sangat cepat. *Data structure* dari Redis Cache sudah mendukung dan kecepatan pemrosesan *cache memory* sangat tinggi.

1. **Solusi keamanan dalam pengiriman data (*backend* dan *mobile apps*)**
   1. Login username dan password

Login standar dengan menggunakan username dan password dari user/customer yang mengakses aplikasi.

* 1. Generate Token tiap service dan enkripsi data

Digunakan pengamanan di tiap pengaksesan service yang ada di aplikasi dengan metode enkrip as256. Generate token yang didapatkan dari proses login user akan digunakan untuk tiap service yang ada di aplikasi, berguna juga untuk *session* user untuk mengakses service tersebut. *Token* yang di generate dari proses login akan disimpan di database dan akan tereset saat user melakukan *logout* atupun saat *session* user tersbeut telah habis. Semua *flow* aplikasi hanya dapat diakses oleh user yang memiliki ­*token login* yang sesuai. Proses keamanan berikut diperlukan untuk penjagaan user / data akses yang melakukan proses data dan untuk menghindari pembajakan data dari *session data*. Hanya user yang memiliki akses yang sesuai dengan token yang tergenerate waktu login.

* 1. Enkripsi Stream Data

Data yang dikirimkan dari aplikasi user akan dienkripsi sebelum data sampai ke ranah *backend*. Data yang dikirimkan akan dicek di *MAIN API* yang selanjutnya data yang di enkripsi pada ranah masing – masing API di tiap *service* aplikasi.