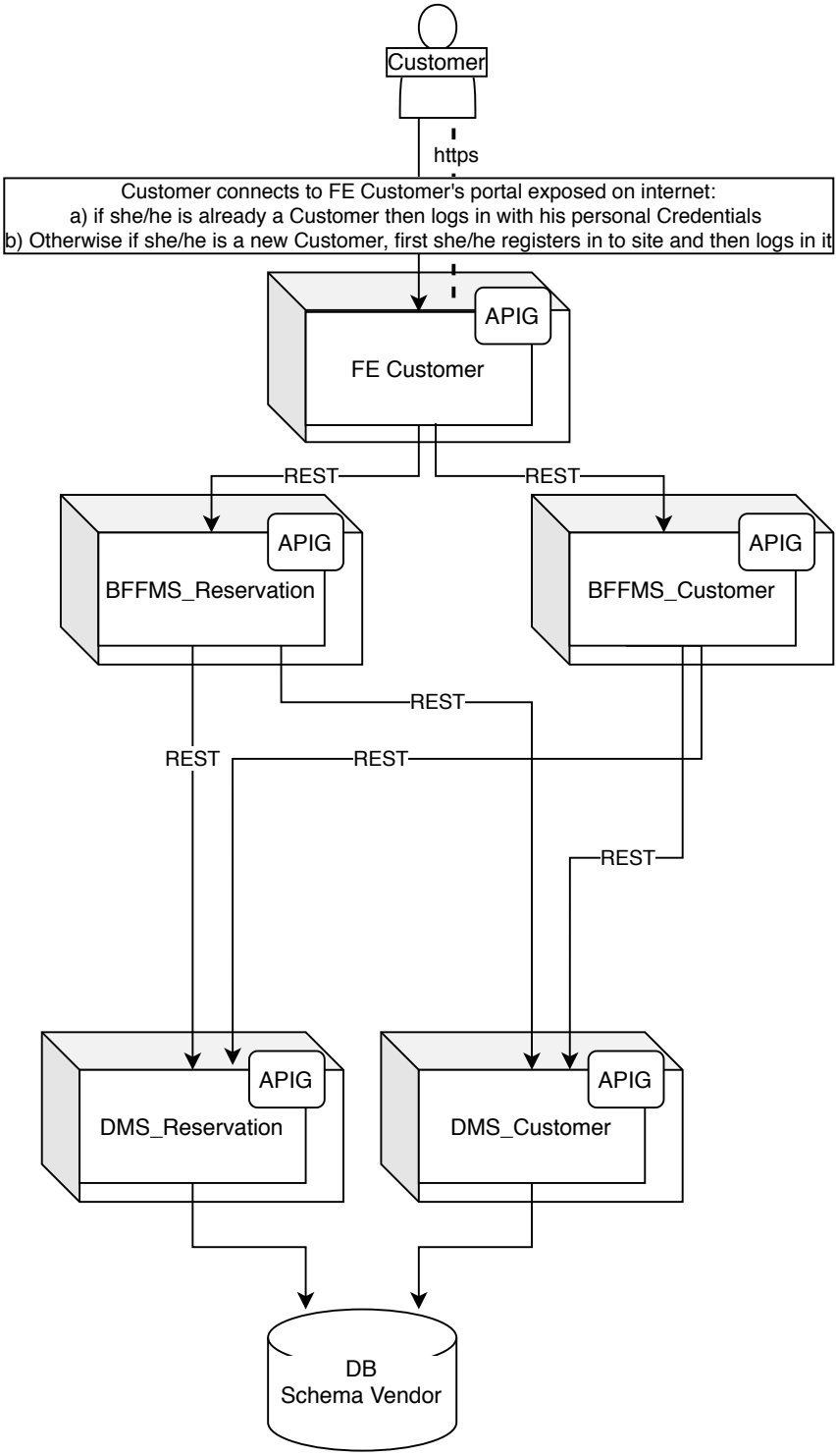
Exercise 2

Describe in words (or with a diagram) the architecture would you use to allow Oscar to store wine reservations and retrieve them quickly and efficiently.

Customers' architecture to register and make reservations



Diagrams Explanation and Notes - Customer services

The FE Customers' sales portal is expose on internet and reachable only under https endpoint (ex: https://www.oscarwines.com/)

As the login all the communication between component are under VPN and strictly via https

BFFMS stay for BackEnd For Front End Microservices

DMS stay for Data Microservices

APIG stay for API Gateway

By design both BFFMSs and DMSs are Docker-ready and could be hosted on cloud

API Gateway authenticates the request and passes an access token (e.g. JSON Web Token) that securely identifies the requestor in each request to the services. A service can include the access token in requests it makes to other services (ex: BFFMS to DMS).

BFFMS_Customer expose all the Rest resources usefully to execute Customer's registrations, logs in, cancelation and update of Customer's data like email, phone and shipping address. Each Customer has access only to her/his personal data

PUT /addCustomer(Customer customer)
GET /getCustomer(Customer customer)
POST /updateCustomer(Customer customer)
DELETE /deleteCustomer(Customer customer)

BFFMS_Reservation expose all the Rest resources usefully to create, visualize and update Customers' reservation, each Customer as the visibility only of her/his personal reservation:

GET /showReservations(ReservationFilter filter)
GET /extractReservation(ReservationFilter filter)
POST /editReservation(Reservation reservation)
PUT /addReservation(Reservation reservation)
DELETE /deleteReservation(Reservation reservation)

DMS_Reservation expose all the Rest resources needed from other BFFMS to manage querying on Reservation data store in the DB

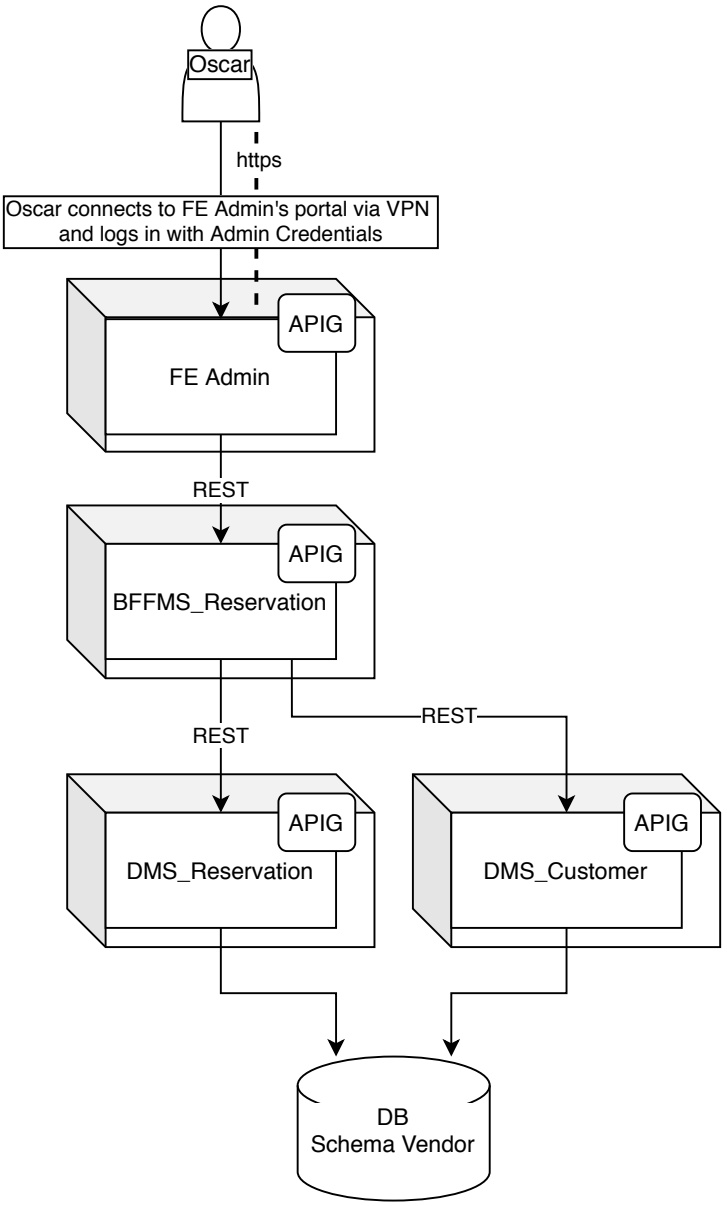
GET /retrieveReservations(ReservationFilter filter)
POST /updateReservation(Reservation reservation)
PUT /addReservation(Reservation reservation)
DELETE /deleteReservation(Reservation reservation)

DMS_Customer expose all the Rest resources needed from other BFFMS to manage querying on Customers' data stored in the DB

POST /editCustomer(Customer customer)
PUT /addReservation(Customer customer)
DELETE /deleteReservation(Customer customer)

DB Schema Vendor contains all the information gathered by the sign in of the Customer and their wines reservations

Admin's architecture to manage reservations



Exercise 2

Describe in words (or with a diagram) the architecture would you use to allow Oscar to store wine reservations and retrieve them quickly and efficiently.

Diagrams Explanation and Notes - Admin services

The FE Admin is reachable only via VPN under https endpoint (ex: https://www.oscarwines.com/admin_entrance)

As the login all the communication between component are under VPN and strictly via https

BFFMS stay for BackEnd For Front End Microservices

DMS stay for Data Microservices

APIG stay for API Gateway

By design both BFFMSs and DMSs are Docker-ready and could be hosted on cloud

API Gateway authenticates the request and passes an access token (e.g. JSON Web Token) that securely identifies the requestor in each request to the services. A service can include the access token in requests it makes to other services (ex: BFFMS to DMS).

BFFMS_Reservation expose all the Rest resources usefully to maintain and query the wine reservations state, retrieves and manages them:

GET /showReservations(ReservationFilter filter)
GET /extractReservation(ReservationFilter filter)
POST /duplicateReservation(Reservation reservation)
POST /updateReservation(Reservation reservation)
PUT /addReservation(Reservation reservation)
DELETE /deleteReservation(Reservation reservation)

DMS_Reservation expose all the Rest resources needed from other BFFMS to manage querying on Reservation data store in the DB

GET /retrieveReservations(ReservationFilter filter)
POST /updateReservation(Reservation reservation)
PUT /addReservation(Reservation reservation)
DELETE /deleteReservation(Reservation reservation)

DMS_Customer expose all the Rest resources needed from other BFFMS to manage querying on Customers' data stored in the DB

POST /editCustomer(Customer customer)
PUT /addReservation(Customer customer)
DELETE /deleteReservation(Customer customer)

DB Schema Vendor contains all the information gathered by the sign in of the Customer and their wines reservations