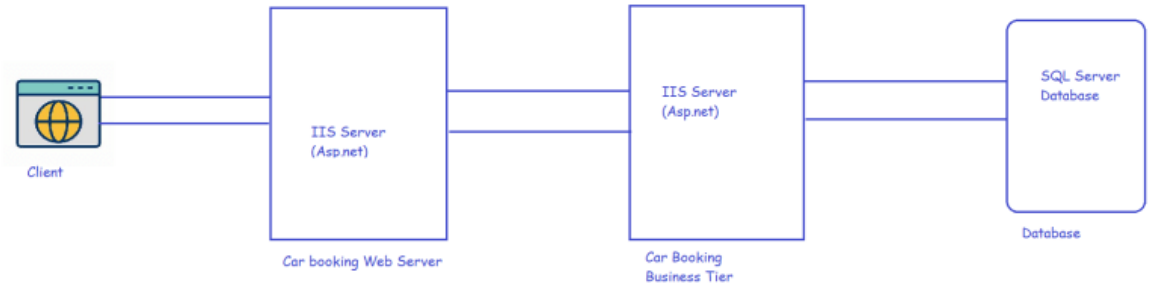
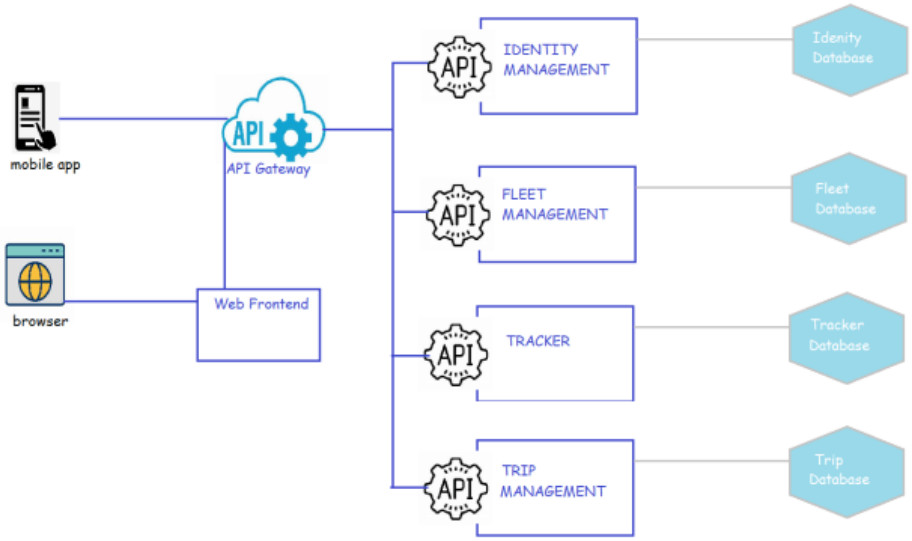
Case Study: Car booking Service

- The current implementation of Car booking Service is as follows

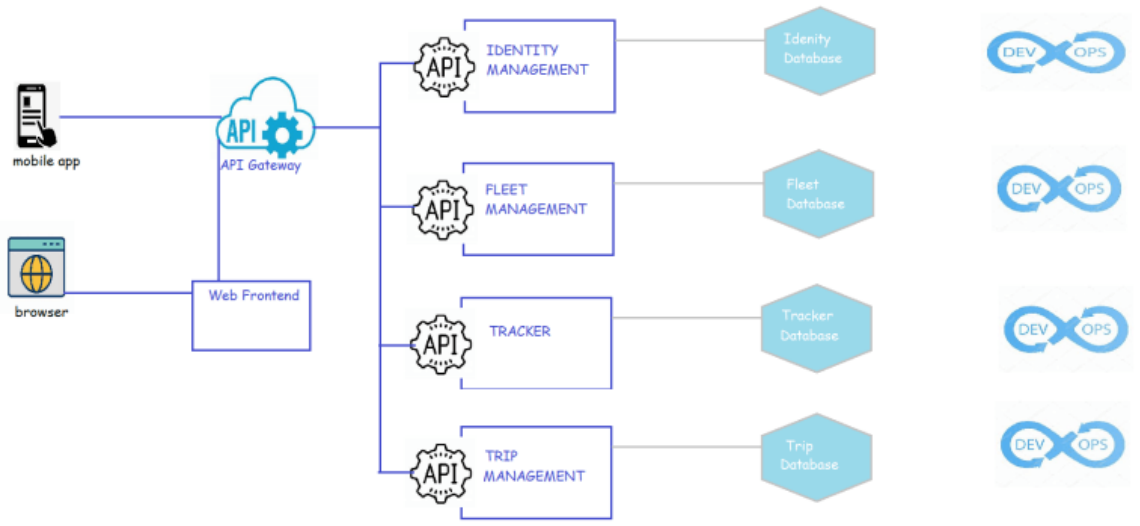


- The current architecture in n-tier application architecture
- Now the Business goals of the Car booking service are
  - Support mobile apps
  - Support Realtime booking of Cars
  - Planning to Serve other developing countries.
- The major components of this application as of now are
  - Web Server
  - App Server
  - Db Server
- As a first step, Architect categorizes the application according to business domains
  - Fleet Management: Here end-user can register their car like OLA/UBER
  - Tracker: Where is my car?
  - Trip Management: Bill calculation
  - Identity & Access Management: Authentication and authorization
- The architecture of the refactored application looks as shown below



- Now, each application is owned by a different team. Each microservice will have its own database.

- Each microservice will have automated CI/CD pipeline.



- To realize this application
  - Each microservice has a docker image
  - This docker image is used to create a Kubernetes deployment

- To understand communication b/w microservices, we need to understand
    - How container in a Pod gets a network?
    - How can two containers in the same pod communicate with each other?
    - How Pod to Pod networking works?
    - How service to Pod networking works?
- How virtual networking works?
- What is bridge?
- Container Networking?
- Pod Networking?

Note: Strangler Pattern is used to switch to microservices from monoliths