High-Level Design Document - SafeRide

Team: #0000FF Thunder
Leon Chen
Frankie Htet
Andy Lee
An Ta
Orion Tang (Team Leader)

Date Issued:
October 6th, 2021

Front end - mvvm

For our front-end design, we will use the MVVM design because MVVM separates what the user views from the model. It allows the user interface to be free of application logic and gives a clean approach, meaning if we need to change the model in the future, the view does not need to be changed or vice-versa. To implement this, we have the view or user interface, which the user will interact with and viewmodel, which handles the UI configurations, validations and communication with the model.

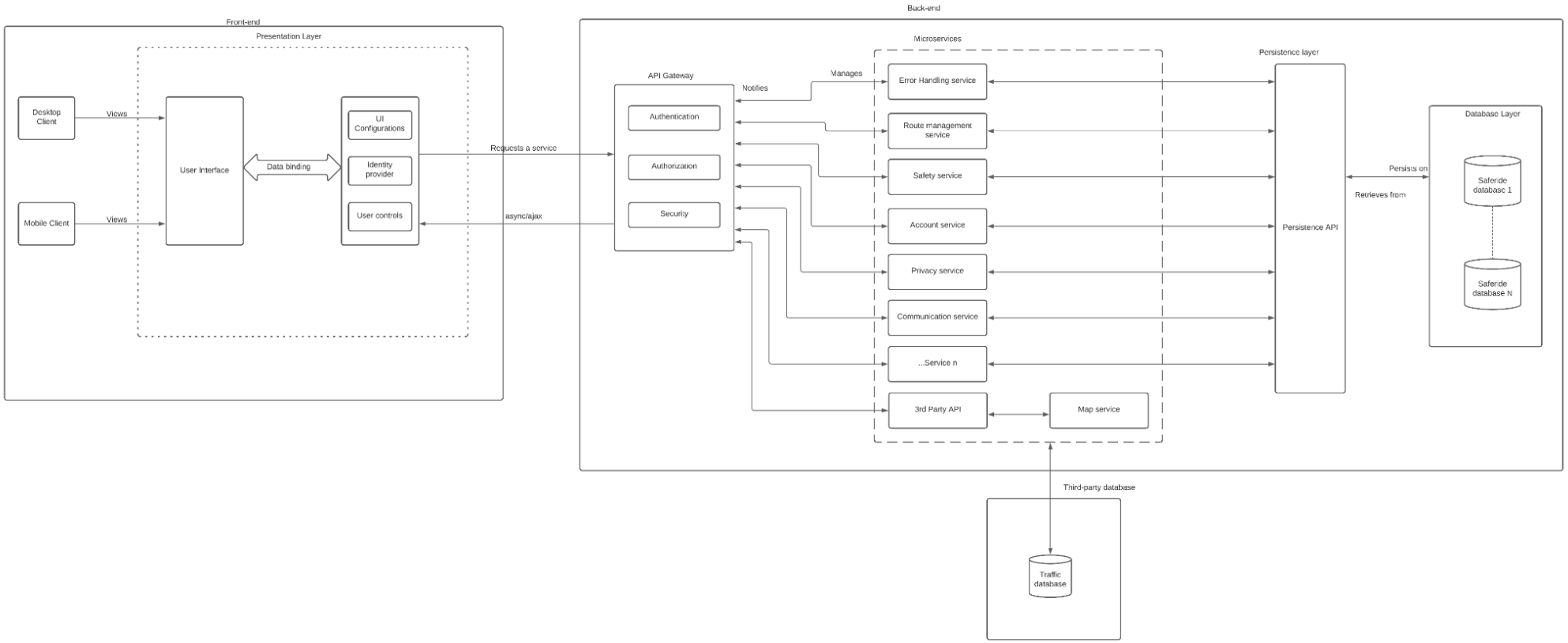
- Presentation layer
    - Includes the user interface and components that communicate with the backend that delivers information and gets rendered to the user.
- Asynchronous requests
    - SPA architecture means that the server sends asynchronous requests to the client, where each component of the client gets updated individually, such that the website does not need to be updated in its entirety.

Backend - SPA + microservices

For our backend design, we will use microservices because this approach allows scalability and optimized functionality as different services are managed independently, therefore fault in one service will not affect another. It is also easy to understand the functionality of individual services. To implement this design, we have API Gateway to return the results from appropriate services for each call. Then we have multiple databases for various services, allowing faster deployment. For third party services to implement functions such as maps, we will use 3rd party API so that services such as google maps can be used with our services.

- Gateway
    - Gateway routes requests from the client to the appropriate microservices to fulfill the request. The gateway is separated from the microservices because it does distinct actions and fulfills specific needs for the client such as authentication and authorization as well as the responsibility of removing the need for the client to know each microservice.
- Microservices
    - A collection of services that is responsible for communicating with the client via the gateway and the database via the persistence layer. These services will be responsible for fulfilling all of the clients requests.
- Persistence layer
    - Persistence layer will be responsible for executing CRUD requests to the database(s). Microservices should go through the persistence layer in order to communicate with the database.
- Database

- Relational or non-relational databases for services will be used to store data for different services. Rather than using 1 database for all services, there are multiple databases to store data for appropriate services.


Front-end
Presentation Layer
Desktop Client
Views
Mobile Client
Views
User Interface
Data binding
UI Configurations
Identity provider
User controls
Requests a service
async/ajax
Back-end
Microservices
API Gateway
Authentication
Authorization
Security
Notifies
Manages
Error Handling service
Route management service
Safety service
Account service
Privacy service
Communication service
...Service n
3rd Party API
Map service
Persistence layer
Persistence API
Persists on
Retrieves from
Database Layer
Saferide database 1
Saferide database N
Third-party database
Traffic database