Students: Constantinova Carina and Postovan Teodora

Topic: An alternative for Yandex Taxi

Service
Service nodes are the ones that do the work a client is interested in. They receive tasks, process them and send back responses. Processing requests is usually costly so we imply the help of gateways and caches. Each service has a database.
- User Service;
- Trip Service;
- Payment service;

Features to be implemented:
- Concurent task limit;
- Status Endpoint;
- Service Discovery;

Gateway
The gateway is the node that receives and forwards user tasks to the service nodes. Based on some logic, the gateway caches responses and balances the load of service nodes. Finally, it has a service registry and chooses from registered services when load balancing.
Features to be implemented:
- Load Balancing: Round Robin;
- Service Discovery;
- Circuit Breaker: Trip if a call to a service fails;

Cache
A cache allows your system to temporary store responses given by your services and serve them without bothering the service nodes. Using caches makes your system more responsive. Usually caches use in-memory storage.
Features to be implemented:
- Keep-Alive Connection;
- Multiple Simultaneous Connections;

Languages to be used: Typescript, Javascript;
Tools to be used:
- Node.js using tool Nx that generates the application;
- MongoDB easy to use, no constraints, no tables (for services);
- Docker;

As stipulated above, there will be 3 main services:
User service where users will be created (name and phone number) with the following API endpoints called by clients in order to register themselves to have the possibility to order a car:
GET /api/user – get a user;
POST /api/user – create a new user;

Trip Service used for driver side of the application:
- GET /trip/location/get – get the location of the driver;
- POST /trip/location/push – set the location of the drive;

Payment service will be used to perform payments for the trip:

- POST /api/payment/register-intent API that registers user intent to pay for the trip, the client will sent the payment intent and the amount;
- POST /payment/confirm update the payment status, status of payment intent is "pending" by default;
- GET /payment/intent simple get intent by ID;

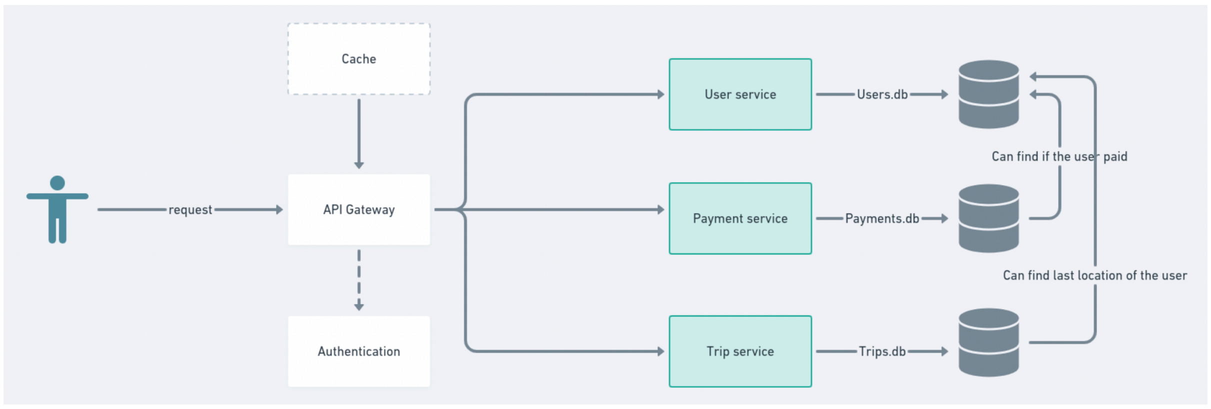
To handle the requests and store some responses will be responsible:
Gateway (handle the requests) through load balancer/Round Robin**:**

- GET /api/status to check if service is alive;
- POST /api/register-service it will call an internal API from gateway service;
  *** To test some requests for the services (POSTMAN) ***
- GET /user/api/user;
- POST /api/user;
- GET trip/location/get;
- POST /trip/location/push;
- POST /payment/api/payment/register-intent;
- POST /payment/api/payment/confirm;
- GET /payment/api/payment/intent;

Cache (store responses)**:**

- GET /cache/status add a route to check cache current values;
- GET /api/status register a health check endpoint;
- GET /get add a route to get a value from cache;
- POST /get add a route to set a value in cache;

Diagram



request
Cache
API Gateway
Authentication
User service
Payment service
Trip service
Users.db
Payments.db
Trips.db
Can find if the user paid
Can find last location of the user