Caderno de Arquitetura

1. Propósito

Este documento descreve a filosofia, decisões, restrições, elementos importantes e quaisquer outros aspectos que envolvem o sistema que dá forma ao design e implementação.

2. Objetivos e Filosofia Arquitetural

A arquitetura escolhida será MVC pois é simples de se compreender para um primeiro desenvolvimento, O sistema terá problemas com manutenção pois é comum boa parte do código se acumular nas controladoras, o time terá de lidar com a aprendizagem da linguagem Swift e como o sistema contará com transações financeiras é necessário manutenção para garantir integridade da segurança com certa periodicidade.

Os objetivos críticos incluem: Cadastro, Login, Recuperação de senha, Cadastro de Viagens, Aceitar Viagens, Integração com serviço de cartão de créditos (pagamentos). Respeitando esses objetivos o aplicativo deve funcionar para a maioria dos casos (assumindo que cancelamentos de viagens ocorreriam com menor frequência).

3. Assunções e Dependências

Uma aplicação que armazena dados cadastrais deve zelar pela segurança dos dados de seus clientes, CPF, nome, endereço, telefone, cartão de crédito, conta do Paypal... Para isso os dados sensíveis dos clientes serão armazenados na forma de um Hash com criptografia SHA-256.

Outra preocupação é com a segurança das transações, para isso usaremos API's de entidades financeiras, Visa, Mastercard, Paypal. Elas tem suas regras de uso e documentação se adequar as exigências de cada uma delas nos dá acesso as medidas de segurança já implementadas pelas companhias e a comodidade de não ter de implementar nada quanto a essas transações e convencer a qualquer uma das instituições que suas transações.

A última preocupação que temos é de exibir os trajetos de maneira correta, isso demanda a exibição de um mapa, para evitar o retrabalho de implementar algo como o google maps ou apple maps, usaremos a API do google maps para exibir e marcar rotas.

Definidas as necessidades de Criptografia e de escolher uma linguagem suportada pelas API's/Frameworks selecionadas, escolhemos a linguagem Swift, por ser a linguagem adotada pela Apple no desenvolvimento de aplicativos para iOS, suportarem as API's mencionadas, como também conta com bibliotecas de criptografia e contar com diversos tutoriais gratuitos na internet.

4. Requerimentos Arquiteturalmente necessários.

https://developer.visa.com/ (API de pagamento para cartões visa)
https://developer.mastercard.com/apis (API mastercard)
https://developer.paypal.com/ (API Paypal)
https://developer.apple.com/security/ (campos: "Protecting User Data" e "Cryptographic Interfaces").

5. Decisões, Restrições e Justificativas

A escolha da arquitetura se deu pela simplicidade, existem modelos mais maduros que trazem benefícios diversos com uma maior complexidade para se aprender durante 6 meses e com responsabilidades diversas além do projeto.

A linguagem de programação, Swift, pelo mesmo motivo, Open source e criada para substituir o objC da Apple, é voltado para desenvolvimento mobile e conta com vários frameworks de testes, pagamentos e comunicação com bancos de dados. Como linguagem conta com menos liberdade que C, mas restrições vem com comodidades pois acesso à variáveis é implicitamente feito por getters e setters.

A restrição de lidar com pagamentos vai será feita através de API's de instituições financeiras, o motivo é simples, essas instituições fornecem um framework ao qual nós apenas precisamos nos adequar e gozamos também da segurança fornecida pela empresa que fornece a API, nos poupando também as dificuldades de aprender como implementar tais transições.

Para poder mostrar as rotas e tracking do percurso, cada celular conta com localização GPS que serve para monitorar viagens e para exibir percursos e mapas precisamos de integração da API do Google maps por sua popularidade e robustez.

Com a escolha da linguagem os membros que não contam com uma maquina que contenha o macOS terão de contornar este impedimento através de algum serviço de acesso remoto para alugar uma maquina com o OS e poder programar em Swift.

6. Mecanismos Arquiteturais

- **Modulo de Login (Login/Cadastro Usuário).**

Este modulo engloba a implementação das classes dos usuários (motorista/ carona), validação de e-mail e CPF, bem como a criptografia dos dados como um todo. - Implementado Parcialmente.

- **Modulo de Viagens (Cadastro Carona/Cancelar Carona/Pedir Carona/ Recusar Carona).**

Esse modulo engloba a implementação das classes de carona (horário de saída, custo, duração, vagas), validação de cancelamentos, validação de vagas disponíveis ao solicitar vaga. - Não Implementado.

- **Modulo de Pagamento (Integração com API's).**

Esse modulo consiste de se adequar as diferentes API's de pagamento listadas, cada API tem suas regras de negócio e segurança, ao respeitar as respectivas regras contamos com a estrutura de segurança já estabelecida de cada API com os dados dos clientes. - Não Implementado.

7. Abstrações Chave

- **Cadastro e Login: Formulário de cadastro com validação e autenticação.**
- **Pagamento: Integração com API's de pagamento.**
- **Sistema de Caronas: Oferecer carona, Aceitar Carona e respectivos cancelamentos exibindo rota no mapa (API Maps) e validações para as ações.**