Sistema de propostas de aluguel
Architecture Notebook

1. Propósito

Este documento descreve a filosofia, decisões, restrições, justificativas, elementos significativos e quaisquer outros aspectos abrangentes do sistema que moldam o design e a implementação.

2. Objetivos da Arquitetura e filosofias

Com a criação de um sistema que será usado por muitos usuários e deve estar manuseando propriedade econômica dos mesmo, esperasse a criação de uma arquitetura para o software que seja principalmente escalável e confiável. Como a aplicação final terá uma interface web e será hospedada na nuvem, não existem tantas preocupações relacionadas às necessidades de hardware, já que esse pode facilmente ser melhorado.

Dado que o sistema é inicialmente simples, espera-se da arquitetura que possa comportar facilmente novas funcionalidades e também a mudança das já existentes. Dessa forma, ter as regras de negócio separadas das interações com os usuário, tanto lógica quanto fisicamente, é extremamente necessário.

3. Premissas e dependências

Sabe-se que o grupo de desenvolvedores e gerente possuem experiência com desenvolvimento web utilizando arcabouços em Javascript e desenvolvimento usando a tecnologia Ruby-on-rails.

O sistema final será hospedado em plataforma online, portanto não existe preocupação atual com limitações físicas (capacidade da rede, hardware, etc…).

O sistema a ser desenvolvido será dividido em dois componentes principais, o front-end e o back-end. Para a criação do back-end, onde a lógica de negócio será criada e a interação com o banco de dados será feita, a metodologia usada será o Model-View-Controller. Para o desenvolvimento do front-end, onde estarão as lógicas de interação com usuário e a parte visual da aplicação

4. Architecturally significant requirements

- https://pt.wikipedia.org/wiki/MVC
- https://github.com/eduardomoroni/react-clean-architecture

5. Decisions, constraints, and justifications

- O sistema será dividido entre Front-end e back-end que se comunicarão por meio de mensagens http, devido a possibilidade da clara separação da lógica de negócio e interface do usuário. Essa separação também permite o uso de tecnologias distintas para implementação das especificidades do sistema e da parte visual da aplicação.
- O backend usará arquitetura Model-Controller, devido a experiência a priori da equipe e a grande disponibilidade de arcabouços backend que oferecem suporte a esta arquitetura. Esse módulo servirá como uma api usada pelo módulo de front-end.
- O frontend deve ser dividido entre camadas de visualização, apresentação e interação.
- A camada de interação é responsável pela interação com elementos externos ao frontend (backend, api 's externas).
- A camada de apresentação é responsável por receber resultados da camada de interação e criar interfaces específicas. Essa camada intermediária foi criada com o objetivo de isolar a lógica de interação com o

usuário da lógica de comunicação com o backend. Ela é somente responsável por transformar a informação recebida da camada de interação de forma a facilitar o uso pela camada de visualização ou transformar a informação recebida da camada de visualização para ser enviada para a camada de interação.

- A camada de visualização é responsável por mostrar informações para o usuário e se adequar às interações do mesmo.
- Cada camada não deve depender diretamente de outra, todas devem ter interfaces específicas e cabe as implementações de cada camada se adequarem as interfaces. Dessa forma, dada a necessidade de mudança em uma camada, contanto que essa não altere as interfaces especificadas, não necessitará nenhuma mudança em outras camadas.
- Nenhuma comunicação a funcionalidades externas ao módulo de front end deve ser feita fora da camada de interação.
- A camada de visualização não deve necessitar de fazer mudanças nas informações recebidas pela camada superior.
- A camada de apresentação, por ser a camada intermediária, pode estar usando arcabouços para melhorar a performance do sistema (cache das informações, cálculos otimizados).
- A criação de pontos de acesso do backend se dará conforme a necessidade do frontend.

6. Mecanismos arquiteturais

Model-Controller

Arquitetura que será usada pelo módulo de backend da aplicação. Tem o propósito de separar o banco de dados em entidades, cujas regras serão separadas da lógica de comunicação com o exterior, que será implementado pelo controller. Possui dois elementos principais:

- Model: Responsável pela implementação de regras específicas de cada entidade do banco de dados, assim como a comunicação (recebimento e escrita de informação).
- Controller: Responsável por recuperar informações disponibilizadas pelas Models e criar uma interface de comunicação com elementos externos ao módulo.

Camada de Interação

Propósito desse mecanismo é proporcionar a ligação entre os módulos de backend e frontend. Deve ser implementado no frontend. Essa camada se utilizará de requisições http para comunicação com a api externa fornecida pelo backend ou outras exteriores.

Camada de Apresentação

Tem o objetivo de separar a camada de interação com o usuário da lógica de comunicação externa. Serve como camada intermediaria que cria interfaces para ambas camadas. Dessa forma, a mudança em endpoints http, ou formato de resposta de mensagens tem o mínimo de impacto possível na interface da aplicação.

Backend e Frontend

Módulos principais da aplicação. Divisão física dos módulos responsáveis pela implementação de regras de negócio e apresentação de informação para o usuário.