

Etapa Técnica: **Vista Software**

Visão geral do problema:

Sistema de Gestão de locação para imobiliárias contendo features de Cadastro, Atualização, Pesquisa (Imóveis, Proprietários, Clientes e Contratos de Locação (além de features que se integram));

- **Back-End:**

PHP (sem frameworks)

- **Front-End:**

HTML, CSS, Javascript e requisições Ajax;

Vale ressaltar que não existe uma separação bem definida entre as camadas do sistema;

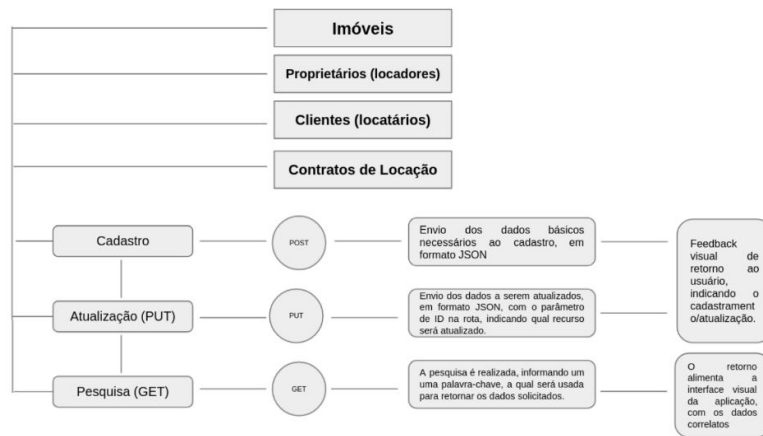
Resolução da Demanda/Problema:

Elaborar um projeto onde pudessem ser resolvidos as demandas de separação das camadas, de forma mais elaborada e, ainda, seguir os requisitos abaixo:

- Facilite as alterações quando é necessário mudar a tela para melhorias de interface;
- Otimize o processo de desenvolvimento do front promovendo a reutilização de componentes em diferentes áreas do sistema;
- Adote padrões e/ou tecnologias atuais que contribuam para qualidade do código;

Separação do Back-End e Front- End

Como estamos falando em refatoração de tal projeto existente, foi pensado a respeito de criar uma API onde pudéssemos prover o serviço das features existentes para os clientes internos e, ainda, para os clientes externos que precisam integrar com nosso sistema;



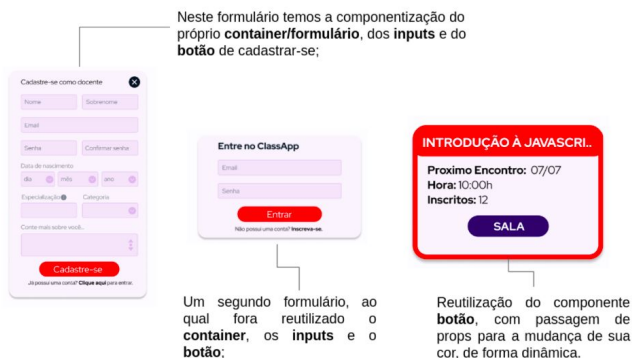
Requisito 1: Facilitar as alterações quando é necessário mudar a tela para melhorias de interface

Visando a manutenção das páginas e seus elementos e, também, visando o seu reuso e, além de otimizar o tempo nas entregas, gerando menor impacto no cliente final, poderíamos trabalhar com **componentização** de elementos. Diante disso, existem algumas bibliotecas que nos ajudam a esse objetivo como o **React**, **Angular** ou **Vue**;

Requisito 2: Otimizar o processo de desenvolvimento do front, promovendo a reutilização de componentes em diferentes áreas do sistema;

Conforme abordado no requisito anterior, o **React** nos traz tal benefício da **componentização de elementos**, diminuindo tempo de desenvolvimento pois, já vão existir vários componentes que podem estar prontos e disponíveis para uso dentro do projeto (visando as boas práticas de que, **todo elemento que tende a se repetir, é um bom candidato a virar um componente**);

Com isso, também ganhamos o benefício da **padronização** e, também, de deixar a **aplicação mais clean**, de forma visual;



Requisito 3: Adote padrões e/ou tecnologias atuais que contribuem para qualidade do código

Aqui, podemos começar a citar as bibliotecas que estariam sendo utilizadas do próprio React e que nos auxiliam nesta demanda como, por exemplo, o **Styled-Components**, o **Material-UI**;

Poderíamos, também, em components que trabalham com lógica, realizá-los utilizando **TDD**, e aplicando os testes durante o seu desenvolvimento;

Por fim, pensando na forma de padronização da equipe interna e visando sua melhor produtividade, poderíamos **padronizar os commits** de cada component ou página dentro do projeto, o que facilitaria a identificar o que cada alteração está fazendo, além de **adotarmos o git flow** que auxiliaria em todas as etapas de desenvolvimento e correções;