PUC Minas

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS - PUC

Especialização em Arquitetura de Sistemas Distribuídos

AFEOL - Arquitetura de Front End

Professor Samuel Martins

Trabalho Final

Arquitetura de um ambiente de desenvolvimento front-end

Equipe

Guilherme Pavarina de Sá - 1333514

Luan Neves da Silva - 1333386

São Paulo
2021

PUC Minas

Resumo

Este trabalho tem como objetivo propor uma solução relacionada com um simples contexto de mercado. A solução aqui apresentada, tem a intenção de abordar diversos pontos relacionados com a criação de um sistema voltado inteiramente para o que se tem disponível no mercado relacionado ao front end. Após a abordagem do projeto fictício, algumas considerações e justificativas a respeito das tecnologias escolhidas para o desenvolvimento deste projeto são realizadas.

São Paulo
2021

PUC Minas

Unifying Stream

Existem algumas dores conhecidas relacionadas com o uso de aplicações de serviço de streaming em geral. A maioria existe logo na página inicial da plataforma, como por exemplo a grande variedade de conteúdo para se escolher. Já outros são possíveis de se identificar antes mesmo de se logar na plataforma. O problema que o projeto proposto visa atacar se encontra nesta última classe de problemas citada. A plataforma *Unifying Stream* tem o objetivo de acabar com este problema de escolha entre plataformas.

O projeto consiste em um sistema capaz de unificar todos os serviços de streaming disponíveis, em um único só local. A plataforma permite buscas entre múltiplos serviços de uma forma transparente ao usuário final. A intenção é que o usuário não tenha que decidir entre uma ou duas plataformas para consumir o conteúdo.

É importante lembrar que existem serviços disponíveis que já realizam este trabalho, entretanto, o Unifying Stream inova no modelo de negócios. A ideia é que o usuário realize o pagamento de forma centralizada. Em outras palavras, o usuário final deve pagar apenas um pacote para a plataforma e, com isso, consumir conteúdo de todas as plataformas disponíveis. Este modelo é possível graças às parcerias existentes entre os serviços disponíveis e o Unifying Stream.

Para realizar o desenvolvimento deste marketplace de serviços de stream, o seguinte ambiente é necessário:

○ Time de desenvolvimento

1. A equipe de desenvolvimento inserida neste contexto possui um vasto conhecimento e grande destreza com ambiente de back-end;
2. As linguagens mais utilizadas pela equipe são fortemente tipadas (Tais quais Java e C/C++);
3. A equipe aceitou o desafio de se especializar em uma ferramenta front-end até que novos colaboradores especializados em front-end fossem contratados.

○ Linguagens de desenvolvimento

Typescript

1. O Background do time é totalmente back-end, com linguagens fortemente tipadas (Java e C). Angular utiliza typescript.
2. Permite fácil manutenção ao longo do tempo, utilizando padrões de projetos orientados a objetos de forma mais simples e fácil de entender.
3. Apesar da curva de aprendizado para novos colaboradores ser maior, dentro do contexto proposto a escolha parece interessante.

São Paulo
2021

PUC Minas

○ IDE ou Softwares de edição/compilação de código

VSCode

1. Simples (e às vezes até nativa) integração com plugins voltados para o framework escolhido (Angular);
2. Snippets de código de forma simples (Por exemplo emmet):
   ```
   div.row
   <div class="row"> </div>
   div.row>div.col-md-4
   <div class="row"> <div class="col-md-4"> </div> </div>
   ```
3. Configurações de ESlint automática;
4. Compartilhamento de regras de código entre a equipe de maneira simples.

○ Definições de padrão de código CSS

OOCSS

1. Facilitar a manutenção do código para o time;
2. Compatível com a escolha do pré-processador SCSS;
3. Nomes de classes mais enxutos.

○ Pré-processador de CSS

SCSS

1. Tem uma boa compatibilidade com o uso de OOCSS;
2. O framework JavaScript escolhido utiliza este pré-processador por padrão;
3. A criação de mixins acaba facilitando bastante o reuso de código entre a equipe;
4. O entendimento sobre mixins parece simples para uma equipe com o background proposto.

○ Framework Visual de CSS

Material Design

1. O framework Material Design foi escolhido devido à sua fácil integração com o framework Angular. Apesar de existir o ng-bootstrap, a integração com angular Material parece mais simples de manter;
2. Outro motivo para escolha do Material, foi o seu design minimalista. Como o time não tem muita especialidade em front-end, um design minimalista não facilita a poluição da interface interferindo diretamente na capacidade cognitiva do usuário;
3. A aplicação sugerida não necessita de muitas features na interface.

PUC Minas

○ Framework JavaScript

Angular

1. Por padrão ele é mais completo do que React. Não é necessário instalação de bibliotecas para cada funcionalidade;
2. Similaridades com linguagens já utilizadas pelo time (Com possibilidade de injeção de dependências, por exemplo);
3. Como o Angular utiliza o typescript por padrão, fica mais fácil manter o sistema.

○ Gerenciador de pacotes e de dependência

npm

A utilização do npm parece mais adequada por conta do suporte e popularidade. Apesar da maior performance do yarn (Comparação), para um time que está iniciando com a tecnologia, utilizar o gerenciador padrão parece a escolha mais correta a se fazer;

○ Estilo Arquitetural

Monorepos

1. A utilização de micro front-ends pode acarretar em um aumento de complexidade desnecessário (Ao menos para início do projeto);
2. A escolha aqui foi utilizar monorepos pois a interface não possui grande complexidade.

○ Padrão Arquitetural

Server-side Rendering

1. Não foi escolhido o estilo de um simples SPA pois é necessário pensar no funcionamento da aplicação em diversos tipos de dispositivos. Não é possível garantir a performance em todos os tipos de dispositivos que vão utilizar a plataforma. Neste caso, um SSR é uma escolha interessante;
2. Favorece o sistema de SEO para a aplicação;
3. Serverless-side rendering não foi escolhido para o projeto inicial por conta de segurança com possíveis crawlers que podem acabar onerando muito o sistema.

○ Ferramentas de testes

Cypress

1. Não existe a necessidade de instalação de drivers para cada tipo de navegador;
2. Assim como os outros frameworks de testes, possui uma fácil integração com jenkins, circleCI, etc;
3. Esta ferramenta de teste possui uma melhor performance do que o Selenium em relação à execução dos testes (Comparação).