

NÍVEL AVANÇADO



PROJETO 20.2

(CONTEÚDO DISPONÍVEL) {
NETFLIX;
CLONE;
(end);
})();

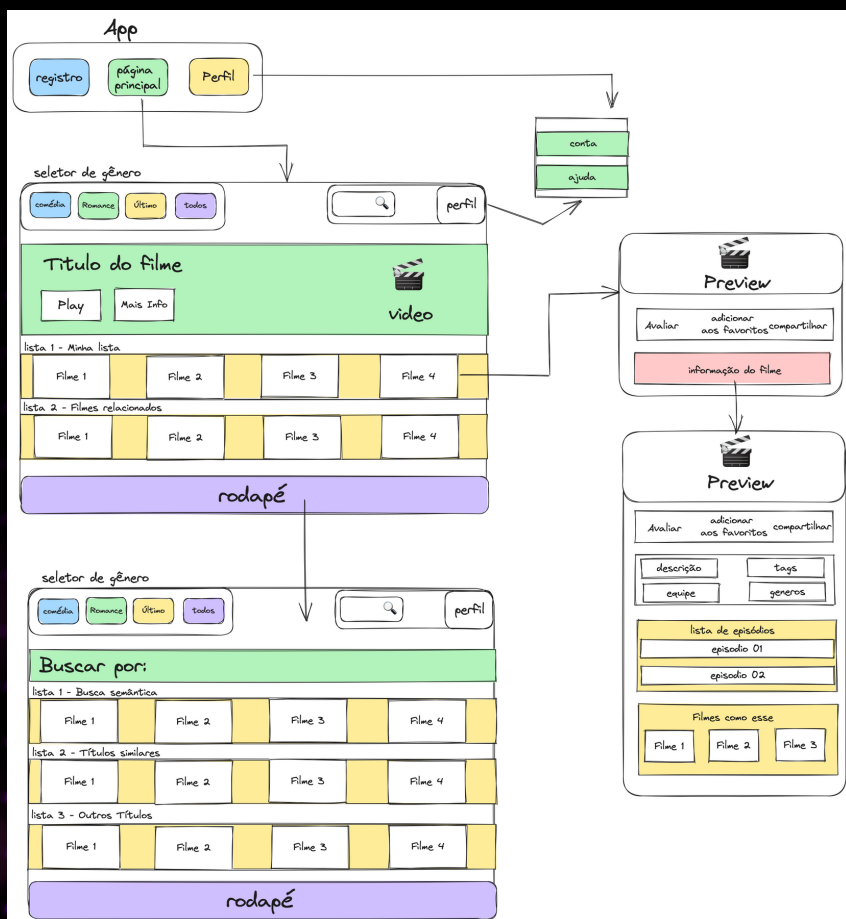
#PORTFÓLIOBOOSTPROGRAM

CONHECIMENTOS REQUIRIDOS:



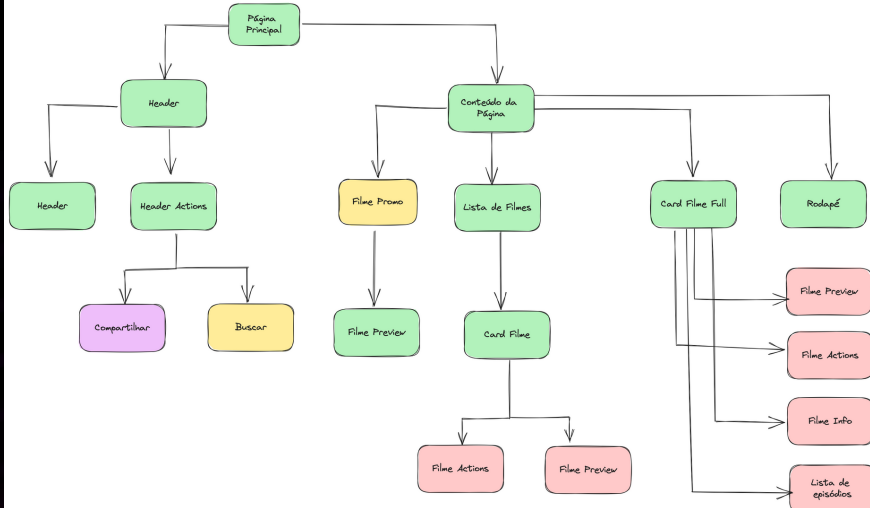
FULL-STACK

WIREFRAME



WIREFRAME

Dependências



Entidades

```

type Movietype Movie =
{
  id: string;
  previewUrl: URL;
  title: string;
  tags: Tag[];
  description?: string;
  episodes?: Episode[];
  rating: number;
  cast: Actor[];
}
  
```

```

type Episode =
{
  id: string;
  movie_id: string;
  previewUrl: URL;
  title: string;
  url: URL[];
  tags: Tag[];
  description?: string;
}
  
```

```

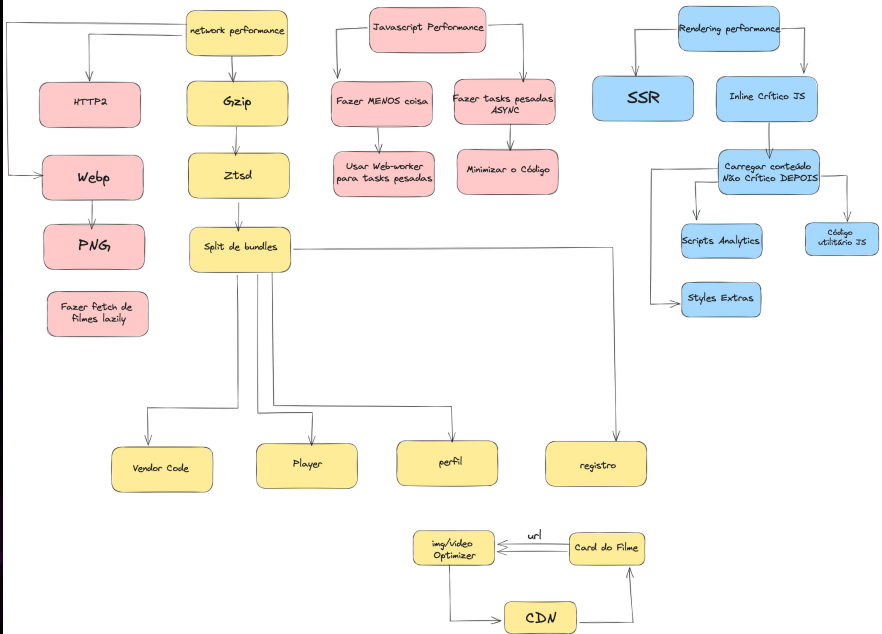
type Actor = {
  id: string;
  fullName: string;
  url: string;
}
  
```

```

type DashboardMap = Map<Tag, Movie[]>
  
```

WIREFRAME

Otimização (Bônus)



NETFLIX CLONE

Clone do app de streaming de vídeo Netflix.

TECH STACK

- React
- TailwindCSS
- nextJS
- Vercel (conhecimento básico de serverless)
- Prisma
- TypeScript



LIBRARIES

- clerkJS
- Jotai Atomic State Management



BRIEFING

Esse projeto será um dos mais **completos** de nosso guia, poucas pessoas tem em seu portfólio um app que mimetiza tão bem a versão do Netflix. Esse App é um **game changer** em seu portfolio.



NÍVEL 1

No primeiro nível iremos fazer uma estrutura de **autenticação** básica, que dessa vez iremos utilizar novamente o **clerkJS** para facilitar nosso desenvolvimento e após o usuário estar logado exibir a tela principal.

NÍVEL 2

Nesse nível iremos inserir a **busca semântica**, todos os atributos e funcionalidades relacionadas a isso. Também será adicionado as informações do filme

NÍVEL 3

Quer correr a milha extra? Crie as funcionalidades: **avaliar**, **adicionar aos favoritos e compartilhar**.

Siga algum dos nossos **schemas** que serão disponibilizados para otimizar o seu código.



REQUISITOS DETALHADOS

Idéia geral de como você irá desenvolver o clone do Netflix:

- ➔ Crie um novo **repositório**.
- ➔ Instale **Next.js v13** e **Tailwind CSS** em seu repositório.
- ➔ Crie a tela de Login/Registro com **ClerkJS**, e habilite também **OAuth** como **login do Google e GitHub**.
- ➔ Crie seu **schema** no **Prisma** com os modelos **User**, **Account**, **Session** e **VerificationToken** que serão disponibilizados.
- ➔ Conecte-se a um **BaaS** como **Railway** ou **PlanetScale**.
- ➔ Antes de iniciar de fato o projeto **teste** a aplicação em produção usando **Vercel**.
- ➔ Crie um arquivo chamado **middleware.ts** dentro da pasta **/src**.
- ➔ Este arquivo será nosso **middleware** para todos os **requests**.
- ➔ Não se esqueça de adicionar a variável de ambiente **CLERK_SECRET_KEY**.
- ➔ Não se esqueça de proteger todas as rotas que não poderemos acessar sem estar autenticados. Inicialmente, somente as rotas

```
publicPaths = ["/", "/sign-in*", "/sign-up*"]
```

serão públicas.

- ➡ Crie a função de **registro** dentro do seu índice não autenticado.
- ➡ Crie a função de **login** no **front-end**, utilizando os componentes do **Clerk**. Veja a documentação do Clerk para obter mais informações:

CLERK



- ➡ O objetivo do login é levar você à rota principal, onde todos os filmes estão listados.
- ➡ Podemos listar todos os filmes disponíveis dentro da pasta **/api/movies**. Você pode listá-los como um **JSON** comum (**NÃO RECOMENDADO**), ou inserir **manualmente** cada filme em seu banco de dados.
- ➡ Se você nunca usou o **Prisma Studio**, esta é uma excelente oportunidade. O Prisma Studio é um editor visual nativo do Prisma que permite inserir dados em seu banco de dados a partir de um endereço **localhost**. Veja mais em:

PRISMA



- ➡ Vou disponibilizar nos recursos um **mock** de alguns filmes para consulta. Siga nossas dependências e entidades para se guiar na criação dos tipos. Lembre-se de que os tipos do **schema.prisma** devem ser idênticos aos das entidades no front.
- ➡ Isso irá assegurar que você tem uma aplicação **end-to-end** **Type Safety**. Para gerenciador de estados, recomendo usar o **JOTAI**:

JOTAI



Um state management atômico, simples e menos verboso que o **Redux**.