Sumário

[Passo 0 – Verificar programas que serão usados 2](#_Toc103179188)

[Node 2](#_Toc103179189)

[Yarn 2](#_Toc103179190)

[Passo 1 – Criar Projeto 2](#_Toc103179191)

[Criar Projeto ReactJS 2](#_Toc103179192)

[Criar Projeto Spring Boot 3](#_Toc103179193)

[Salvar a Primeira versão do Projeto no Github 4](#_Toc103179194)

[Limpar o Projeto React 5](#_Toc103179195)

[Passo 2 – Adicionar CSS e Bootstrap ao Projeto 5](#_Toc103179196)

[CSS Components 5](#_Toc103179197)

[Index 6](#_Toc103179198)

[Passo 3- Adicionar Navbar 6](#_Toc103179199)

[Tsconfig/json 6](#_Toc103179200)

[src/componentes/Navbar 6](#_Toc103179201)

[App.tsx 7](#_Toc103179202)

[CSS do Navbar 7](#_Toc103179203)

[Passo 4 - Rotas 8](#_Toc103179204)

[Listing 9](#_Toc103179205)

[Form 10](#_Toc103179206)

[Passo 5 - Tela de Formulário 10](#_Toc103179207)

[Index 10](#_Toc103179208)

[CSS 12](#_Toc103179209)

[Passo 6 - Paginação 14](#_Toc103179210)

[Index 14](#_Toc103179211)

[Passo 100 – Verificar programas que serão usados 14](#_Toc103179212)

[Node 14](#_Toc103179213)

# Passo 0 – Verificar programas que serão usados

## Node

Verificar se o node já está instalado com o comando “ *node -v ”* no cmd.

Caso precise instalar, acesse o link <https://nodejs.org/en/download/> e seleciona a versão mais recente LTS.

## Yarn

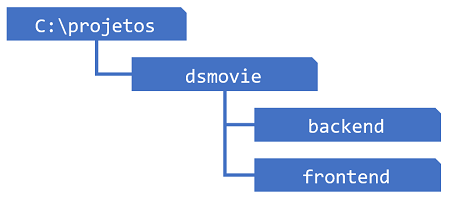
Verificar se o yarn já está instalado com o comando “ *yarn -v ”* no cmd.

Caso precise instalar, cole este comando no cmd “ npm install --global yarn “.

# Passo 1 – Criar Projeto

## Criar Projeto ReactJS

Esta será a estrutura de pastas que vamos criar:



Criar a pasta Projetos (neste estudo foi criado em C\: ). Dentro dessa pasta criar a pasta de nome *DSMOVIE.*

Dentro da pasta, abrir o Git Bash Here ou acessar pelo cmd.

Comando “ yarn create react-app frontend --template typescript “.

Caso não funcione usar ( npx create-react-app frontend --template typescript ).

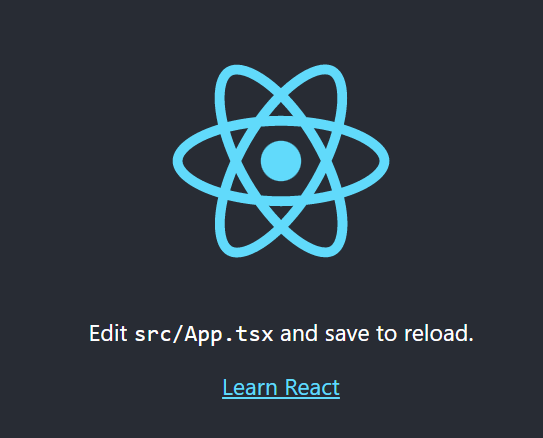
Dentro da pasta Projetos será criado uma pasta de nome *frontend.*

IMPORTANTE: deletar subpasta .git

Caso não apareça, ver se está habilitado “ extensões e arquivos ocultos “.

Dentro da pasta frontend, abrir o Git Bash Here e executar o comando “ *yarn start “.*

Se estiver tudo certo, será aberta uma página no navegador contendo essa exibição:

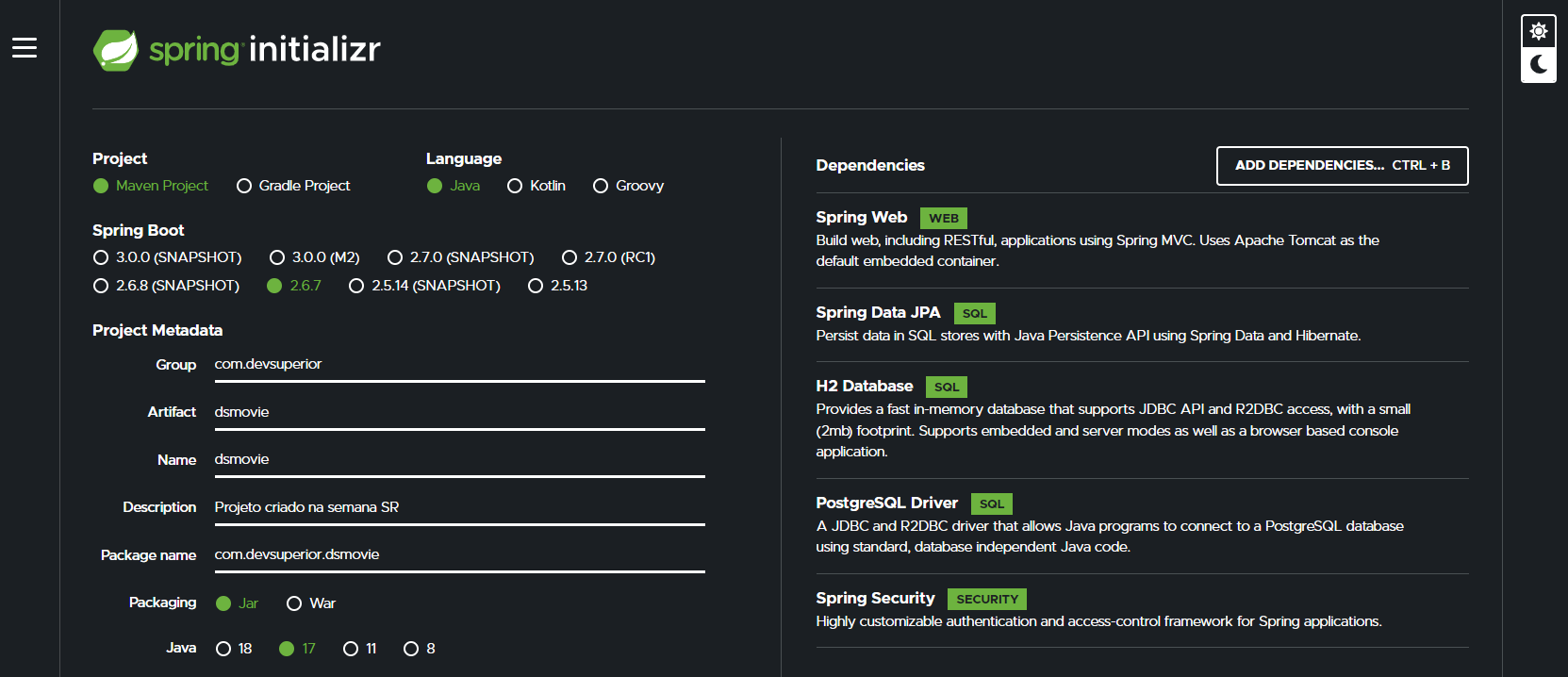


Parar o projeto com “ ctrl + c ” no git.

## Criar Projeto Spring Boot

Criar projeto Spring Boot no Spring Initializr ( <https://start.spring.io/> ) com as seguintes dependências:

Web // JPA // H2 // Postgres // Security



Clicar em “ Generate “ e salvar o arquivo dentro da pasta dsmovie, onde está o frontend.

Descompactar o arquivo baixado e após isso excluir o arquivo zipado. Renomear a pasta dsmovie para *backend.*

Para testar se está tudo certo com o backend, será utilizado o STS.

Baixar em <https://spring.io/tools> selecionando o sistema operacional correto.

Após o arquivo ser baixado, extrair com um descompactador de arquivos, e em seguida extrair a pasta *“ sts “* (neste estudo foi criado em C\: ).

Abrir essa pasta e executar a IDE (SpringToolSuite4). Selecionar File – Import – Maven – Existing Maven… - selecionar a pasta *backend* do vsmovie.

Esperar o Mavem baixar as dependências. Será criado o projeto SpringBoot.

Realizar ajuste no arquivo pom.xml. Na linha de plugins inserir as informações abaixo:

<plugin>

<groupId>org.apache.maven.plugins</groupId>

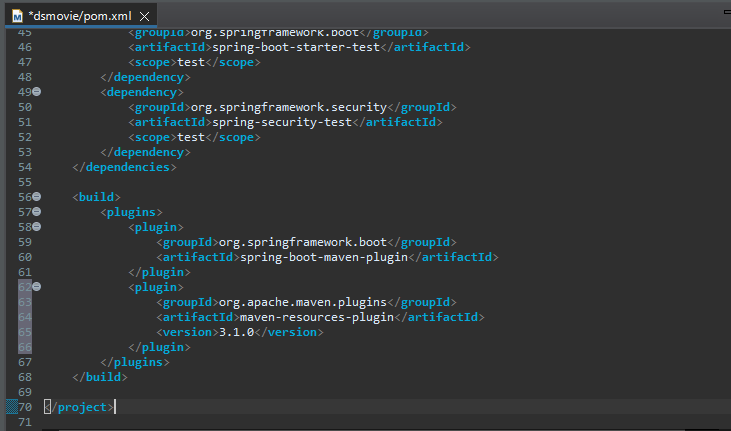
<artifactId>maven-resources-plugin</artifactId>

<version>3.1.0</version>

</plugin>

Salvar o arquivo editado.

Deve ficar assim:



Atualizar o projeto clicando com botão direito sobre o nome do projeto (dsmovie) – Mavem – Update Project – Marcar a caixinha Force Update

## Salvar a Primeira versão do Projeto no Github

No git hub, após criar o repositório, selecionar a chave SSH para subir o projeto.

Abrir o Git Bash Here dentro da pasta *dsmovie* e executar os passos:

* git init // git add . // git commit -m "Projeto criado" // git branch -M main
* git remote add origin git@github.com:seuusuario/seurepositorio.git
* git push -u origin main

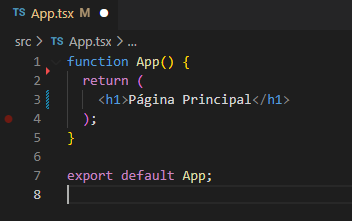
Caso de o erro (fatal: Could not read from remote repositor), executar o passo a passo do vídeo (<https://www.youtube.com/watch?v=_hZf1teRFNg>) abaixo a partir do minuto 12:10.

## Limpar o Projeto React

Dentro da pasta *frontend* abrir o VSCode. Apagar tudo na pasta *public* exceto o *index.html.* Dentro do arquivo html, apagar os comentários e os links para imagens. Ajustar o *title.*

Dentro da pasta src apagar o arquivo app.teste.tsx, app.css, logo.svg, reportwebvitals e setup.tests

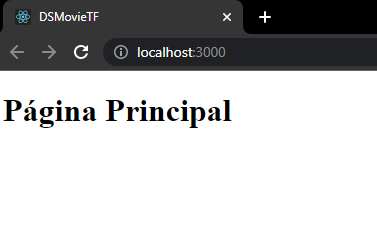
Dentro do App.tsx deixar assim:



Apagar tudo dentro do arquivo index.css.

Dentro do arquivo index.tsx apagar os comentários, reportWebVitals() e import reportWebVitals from './reportWebVitals';

Abrir um terminal do Git Bash Here na pasta frontend e executar o comando *yarn start* para testar. Deverá ser carregado a página principal limpa:



Realizar um novo commit para salvar as alterações no projeto. Exemplo: “Project clean”.

# Passo 2 – Adicionar CSS e Bootstrap ao Projeto

No terminal do *frontend,* executar o comando yarn add bootstrap@5.1.3

Aguardar a instalação ser concluída.

## CSS Components

No arquivo index.css do components, colar o código padrão abaixo:

@import url('https://fonts.googleapis.com/css2?family=Open+Sans:wght@400;700&display=swap');

:root { --color-primary: #4D41C0;}

\* { box-sizing: border-box;

font-family: 'Open Sans', sans-serif;}

html, body { background-color: #E5E5E5;}

a, a:hover { text-decoration: none;

color: unset;}

## Index

No arquivo index.tsx, colar o código padrão abaixo:

import 'bootstrap/dist/css/bootstrap.css';

Realizar um novo commit para salvar as alterações no projeto. Exemplo: “Bootstrap”.

# Passo 3- Adicionar Navbar

## Tsconfig/json

No arquivo tsconfig.json colar o macete para encurtar pastas ao selecionar arquivos:

"baseUrl": "./src", (reinicie o yarn (ctrl+c para parar)

## src/componentes/Navbar

Na pasta src, criar uma pasta *componentes* e dentro dessa criar outra pasta de nome *Navbar*. Criar um arquivo index.tsx dentro dessa pasta.

import {ReactComponent as GithubIcon} from 'assets/img/github.svg';

import './styles.css';

function Navbar() {

return (

<header>

<nav className="container">

<div className="dsmovie-nav-content">

<h1>DSMovie</h1>

<a

href="https://github.com/devsuperior"

target="\_blank"

rel="noreferrer"

>

<div className="dsmovie-contact-container">

<GithubIcon />

<p className="dsmovie-contact-link">/devsuperior</p>

</div>

</a>

</div>

</nav>

</header>

);

}

## App.tsx

import Navbar from "./components/Navbar";

function App() {

return(

<Navbar />

)

}

export default App;

## CSS do Navbar

Dentro da pasta Navbar criar o arquivo *styles.css*, no caso deste projeto foi estilizado assim:

header {

height: 60px;

background-color: var(--color-primary);

display: flex;

align-items: center;

}

.dsmovie-nav-content {

display: flex;

align-items: center;

justify-content: space-between;

color: #fff;

}

.dsmovie-nav-content h1 {

font-size: 24px;

margin: 0;

font-weight: 700;

}

.dsmovie-contact-container {

display: flex;

align-items: center;

}

.dsmovie-contact-link {

margin: 0 0 0 10px;

font-size: 12px;

}

# Passo 4 - Rotas

Instalar React Router DOM no terminal do frontend com o comando

yarn add react-router-dom@6.2.1 @types/react-router-dom@5.3.2

Após a instalação colar o código abaixo no app.tsx (substituir o código que está lá):

import {

BrowserRouter,

Routes,

Route

} from "react-router-dom";

import Listing from 'pages/Listing';

import Form from 'pages/Form';

import Navbar from "components/Navbar";

function App() {

return (

<BrowserRouter>

<Navbar />

<Routes>

<Route path="/" element={<Listing />} />

<Route path="/form">

<Route path=":movieId" element={<Form />} />

</Route>

</Routes>

</BrowserRouter>

);

}

export default App;

## Listing

Dentro da pasta src criar a pasta *pages* e dentro desta a pasta *Listing*. Dentro desta criar o arquivo index.tsx e colocar o código abaixo:

function Listing(){

return(

<h1>Página de Listagem</h1>

);

}

export default Listing;

## Form

Dentro da pasta src criar a pasta *form*. Dentro desta criar o arquivo index.tsx e colocar o código abaixo:

Import './styles.css';

function Listing(){

return(

<h1>Página de Formulário</h1>

);

}

export default Listing;

Realizar um novo commit para salvar as alterações no projeto. Exemplo: “Routes”.

# Passo 5 - Tela de Formulário

## Index

No arquivo index.tsx da pasta Form, inserir o objeto abaixo como exemplo em caráter de teste.

function Form() {

const movie = {

id: 1,

image:

"https://www.themoviedb.org/t/p/w533\_and\_h300\_bestv2/jBJWaqoSCiARWtfV0GlqHrcdidd.jpg",

title: "The Witcher",

count: 2,

score: 4.5,

};

return (

<div className="dsmovie-form-container">

<img className="dsmovie-movie-card-image" src={movie.image} alt={movie.title} />

<div className="dsmovie-card-bottom-container">

<h3>{movie.title}</h3>

<form className="dsmovie-form">

<div className="form-group dsmovie-form-group">

<label htmlFor="email">Informe seu email</label>

<input type="email" className="form-control" id="email" />

</div>

<div className="form-group dsmovie-form-group">

<label htmlFor="score">Informe sua avaliação</label>

<select className="form-control" id="score">

<option>1</option>

<option>2</option>

<option>3</option>

<option>4</option>

<option>5</option>

</select>

</div>

<div className="dsmovie-form-btn-container">

<button type="submit" className="btn btn-primary dsmovie-btn">

Salvar

</button>

</div>

</form>

<button className="btn btn-primary dsmovie-btn mt-3">Cancelar</button>

</div>

</div>

);

}

export default Form;

## CSS

Dentro da pasta src criar o arquivo styles.css e colocar o código abaixo:

.dsmovie-form-container {

max-width: 480px;

margin: 40px auto;

padding: 20px;

}

.dsmovie-form {

width: calc(100% - 20px);

}

.dsmovie-form-group {

margin-bottom: 20px;

}

.dsmovie-form-group label {

font-size: 12px;

color: #aaa;

}

.dsmovie-form-btn-container {

display: flex;

justify-content: center;

}

.dsmovie-movie-card-image {

width: 100%;

border-radius: 8px 8px 0 0;

}

.dsmovie-card-bottom-container {

background-color: #fff;

display: flex;

flex-direction: column;

align-items: center;

padding: 10px 10px 20px 10px;

border-radius: 0 0 8px 8px;

}

.dsmovie-card-bottom-container h3 {

color: #4A4A4A;

text-align: center;

font-size: 14px;

font-weight: 700;

margin-bottom: 10px;

min-height: 40px;

}

.dsmovie-btn {

background-color: var(--color-primary);

font-size: 14px;

font-weight: 700;

height: 40px;

width: 180px;

display: flex;

align-items: center;

justify-content: center;

}

Realizar um novo commit para salvar as alterações no projeto. Exemplo: “Form layout”.

# Passo 6 - Paginação

## Index

No arquivo index.tsx da pasta Pagination, inserir código abaixo:

import { ReactComponent as Arrow} from 'assets/img/arrow.svg';

import './styles.css';

function Pagination() {

return (

<div className="dsmovie-pagination-container">

<div className="dsmovie-pagination-box">

<button className="dsmovie-pagination-button" disabled={true}>

<Arrow />

</button>

<p>{`${1} de ${3}`}</p>

<button className="dsmovie-pagination-button" disabled={false}>

<Arrow className="dsmovie-flip-horizontal" />

</button>

</div>

</div>

);

}

export default Pagination;

## Listing

No arquivo index.tsx na pasta Listing, inserir código abaixo:

import Pagination from "components/Pagination";

function Listing() {

return (

<Pagination />

);

}

export default Listing;

## CSS Pagination

Criar arquivo styles.css na pasta Pagination e colar o código:

.dsmovie-pagination-container {

padding: 15px 0;

display: flex;

justify-content: center;

align-items: center;

}

.dsmovie-pagination-box {

width: 180px;

display: flex;

justify-content: space-between;

align-items: center;

}

.dsmovie-pagination-box form {

width: 100%;

display: flex;

}

.dsmovie-pagination-button {

width: 40px;

height: 40px;

border-radius: 4px;

background-color: #fff;

display: flex;

justify-content: center;

align-items: center;

border: 1px solid var(--color-primary);

cursor: pointer;

}

.dsmovie-pagination-button svg {

filter: brightness(0) saturate(100%) invert(26%) sepia(19%) saturate(7395%) hue-rotate(234deg) brightness(89%) contrast(92%);

}

.dsmovie-pagination-button:disabled {

border: 1px solid #c2c2c2;

cursor: unset;

}

.dsmovie-pagination-button:disabled svg {

filter: none;

}

.dsmovie-pagination-container p {

margin: 0;

font-size: 12px;

color: var(--color-primary);

}

.dsmovie-flip-horizontal {

transform: rotate(180deg);

}

Realizar um novo commit para salvar as alterações no projeto. Exemplo: “Pagination layout”.

# Passo 7 – Movie Star

## Index

Criar na pasta componentes uma pasta de nome *MovieStar* e dentro dessa um arquivo index.tsx com o código:

import{ ReactComponent as StarFull} from 'assets/img/star-full.svg';

import{ ReactComponent as StarHalf} from 'assets/img/star-half.svg';

import{ ReactComponent as StarEmpty} from 'assets/img/star-empt.svg';

import './styles.css';

function MovieStar() {

return (

<div className="dsmovie-stars-container">

<StarFull />

<StarFull />

<StarFull />

<StarHalf />

<StarEmpty />

</div>

);

}

export default MovieStar;

## Listing -> Index

Inserir as MovieStar na Index:

import MovieStar from "components/MovieStar";

import Pagination from "components/Pagination";

function Listing() {

return (

<>

<Pagination />

<MovieStar />

</>

);

}

export default Listing;

## MovieStar -> CSS

Criar o arquivo styles.css na pasta MovieStar e colar o código:

.dsmovie-stars-container {

width: 130px;

display: flex;

justify-content: space-between;

}

.dsmovie-stars-container svg {

width: 22px;

height: auto;

}

# Passo 8 – Movie Score

## Index

Criar na pasta componentes uma pasta de nome *MovieScore* e dentro dessa um arquivo index.tsx com o código:

import MovieStar from "components/MovieStar";

import './styles.css';

function MovieScore() {

const score = 3.5;

const count= 13;

return (

<div className="dsmovie-score-container">

<p className="dsmovie-score-value">{score > 0 ? score.toFixed(1) : "-"}</p>

<MovieScore />

<p className="dsmovie-score-count">{count} avaliações</p>

</div>

);

}

export default MovieScore;

## MovieScore -> CSS

Criar o arquivo styles.css na pasta MovieStar e colar o código:

.dsmovie-score-container {

display: flex;

flex-direction: column;

align-items: center;

}

.dsmovie-score-value {

margin: 0;

color: #FFBB3A;

font-size: 16px;

font-weight: 700;

}

.dsmovie-score-count {

font-size: 12px;

color: #989898;

margin: 4px 0 10px 0;

}

## Listing -> Index

Inserir MovieScore na Index:

import MovieScore from "components/MovieScore";

import Pagination from "components/Pagination";

function Listing() {

return (

<>

<Pagination />

<MovieScore />

</>

);

}

export default Listing;

# Passo 9 – Movie Card

## Index

Criar na pasta componentes uma pasta de nome *MovieCard* e dentro dessa um arquivo index.tsx com o código:

import MovieScore from "components/MovieScore";

function MovieCard() {

const movie = {

id: 1,

image:

"https://www.themoviedb.org/t/p/w533\_and\_h300\_bestv2/jBJWaqoSCiARWtfV0GlqHrcdidd.jpg",

title: "The Witcher",

count: 2,

score: 4.5,

};

return (

<div>

<img

className="dsmovie-movie-card-image"

src={movie.image}

alt={movie.title}

/>

<div className="dsmovie-card-bottom-container">

<h3>{movie.title}</h3>

<MovieScore />

<div className="btn btn-primary dsmovie-btn">Avaliar</div>

</div>

</div>

);

}

export default MovieCard;

## Listing -> Index

Inserir MovieCard na Index:

import MovieCard from "components/MovieCard";

import Pagination from "components/Pagination";

function Listing() {

return (

<>

<Pagination />

<MovieCard />

</>

);

}

export default Listing;

# Passo 100 – Verificar programas que serão usados

## Node

Verificar se o node já está instalado com o comando “ *node -v ”* no cmd.

Caso precise instalar, acesse o link <https://nodejs.org/en/download/> e seleciona a versão mais recente LTS.