# Documentação para Build, Compactação e Deploy do Projeto "Planilha Organizadora"

## 1. Criando o Build do Projeto

Para preparar o projeto para produção, usamos Webpack para gerar um build otimizado.

### Configurando Webpack:

1. Instale o Webpack e Webpack CLI:
2. npm install --save-dev webpack webpack-cli
3. Crie o arquivo webpack.config.js:
4. const path = require('path');
5. module.exports = {
6. entry: './src/index.js',
7. output: {
8. filename: 'bundle.js',
9. path: path.resolve(\_\_dirname, 'dist'),
10. },
11. mode: 'production',
12. };
13. Para gerar o build, execute:
14. npm run build

## 2. Compactação e Minificação do Código

Para reduzir o tamanho do código e melhorar o desempenho, usamos o **Terser**:

npm install --save-dev terser-webpack-plugin

E adicionamos ao webpack.config.js:

const TerserPlugin = require('terser-webpack-plugin');

module.exports = {

mode: 'production',

optimization: {

minimize: true,

minimizer: [new TerserPlugin()],

},

};

Agora, ao rodar npm run build, o código será minificado.

## 3. Adicionando Jest para Testes

Instale o Jest:

npm install --save-dev jest

Crie um arquivo jest.config.js:

module.exports = {

testEnvironment: "node",

verbose: true,

};

Crie um teste em \_\_tests\_\_/utils.test.js:

const soma = require("../src/utils");

test("soma 2 + 3 e retorna 5", () => {

expect(soma(2, 3)).toBe(5);

});

Para rodar os testes:

npx jest

Ou adicione um script ao package.json:

"scripts": {

"test": "jest --watch"

}

E execute:

npm test

## 4. Deploy do Frontend no Vercel

### Passos para Deploy:

1. Instale o CLI do Vercel:
2. npm install -g vercel
3. Faça login:
4. vercel login
5. Inicie o deploy:
6. vercel
7. Para enviar para produção:
8. vercel --prod

## 5. Deploy do Backend no Railway

1. Instale o CLI do Railway:
2. npm install -g railway
3. Faça login:
4. railway login
5. Inicie o projeto Railway:
6. railway init
7. Envie o backend para produção:
8. railway up

## 6. Automação com GitHub Actions (Testes + Deploy Automático)

### Configuração dos Tokens

Para que o deploy funcione automaticamente, configure os **tokens de autenticação** no GitHub:

1. **Obtenha o Token do Vercel:**
   * Acesse [Vercel Tokens](https://vercel.com/account/tokens)
   * Gere um novo token e copie o valor.
2. **Obtenha o Token do Railway:**
   * Acesse [Railway Settings](https://railway.app/)
   * Gere um novo token e copie o valor.
3. **Adicione os Tokens no GitHub:**
   * Vá ao repositório no **GitHub** → **Settings** → **Secrets and variables** → **Actions**.
   * Clique em **"New repository secret"** e adicione:
     + **Nome:** VERCEL\_TOKEN → **Valor:** (cole o token do Vercel)
     + **Nome:** RAILWAY\_TOKEN → **Valor:** (cole o token do Railway)

### Criando o Workflow de Deploy Automático

Crie um arquivo .github/workflows/deploy.yml:

name: Testes e Deploy

on:

push:

branches:

- main

jobs:

test:

runs-on: ubuntu-latest

steps:

- name: Checkout do código

uses: actions/checkout@v3

- name: Instalar dependências

run: npm install

- name: Rodar testes com Jest

run: npm test

frontend:

needs: test

runs-on: ubuntu-latest

steps:

- name: Checkout do código

uses: actions/checkout@v3

- name: Instalar dependências

run: npm install

- name: Criar build do frontend

run: npm run build

- name: Deploy no Vercel

run: vercel --prod --token=${{ secrets.VERCEL\_TOKEN }}

backend:

needs: test

runs-on: ubuntu-latest

steps:

- name: Checkout do código

uses: actions/checkout@v3

- name: Instalar Railway CLI

run: npm install -g railway

- name: Login no Railway

run: railway login --token=${{ secrets.RAILWAY\_TOKEN }}

- name: Deploy no Railway

run: railway up

Agora, toda vez que você fizer git push na branch main, o GitHub Actions: ✅ Baixa o código ✅ Instala as dependências ✅ **Executa os testes com Jest** 🔄 **Se os testes passarem, faz o deploy no Vercel e Railway**

Isso garante que o deploy só aconteça se o código estiver funcionando corretamente! 🚀

