Lista de ferramentas, bibliotecas e frameworks usado:

Ferramentas:

ESLint

Prettier

Nodemon

Frameworks:

Express

Sequelize

Swagger

Bibliotecas:

bcrypt

body-parser

cookie-parser

cors

debug

dotenv

ejs

express-rate-limit

express-validator

helmet

http-errors

ip-address

jsonwebtoken

morgan

mysql2

Dependências de Desenvolvimento:

@eslint/js

eslint-config-prettier

eslint-plugin-node

eslint-plugin-prettier

globals

Comandos de instalação e configuração:

- **Tutorial de Inicialização e Configuração do Projeto -**

1. Pré-requisitos

- Node.js versão 20.17.0 (use o NVM para gerenciar versões)

- PowerShell 7 recomendado para usuários Windows

2. Inicialização do Projeto Express

1. Criar o projeto Express:

express --view=ejs --css=css --git libraryapi

cd libraryapi

2. Instalar dependências principais:

npm install express@latest mysql2 sequelize cors helmet body-parser express-validator dotenv jsonwebtoken bcrypt ip-address

3. Instalar dependências de desenvolvimento:

npm install -D nodemon eslint prettier eslint-plugin-prettier eslint-config-prettier eslint-plugin-node @eslint/js

4. Instalar dependências e atualizar pacotes:

npm install

npm update

npm audit fix --force

5. Configurar o ESLint:

Inicialize o ESLint com o seguinte comando:

npx eslint --init

3. Configurações Recomendadas

- **Configuração do ESLint**

Crie o arquivo eslint.config.mjs na pasta raiz do projeto libraryapi com o seguinte conteúdo:

```
import globals from "globals";
import pluginJs from "@eslint/js";
import eslintPluginNode from "eslint-plugin-node";
import eslintPluginPrettier from "eslint-plugin-prettier";

/** @type {import('eslint').Linter.Config[]} */
export default [
 {
   files: ["**/*.js"],
   languageOptions: { sourceType: "commonjs" },
   env: {
```

```
    node: true,
    es2021: true
  },
  parserOptions: {
    ecmaVersion: 2021,
    sourceType: "commonjs"
  },
  plugins: {
    node: eslintPluginNode,
    prettier: eslintPluginPrettier
  },
  extends: [
    "eslint:recommended",
    "plugin:node/recommended",
    "plugin:prettier/recommended"
  ],
  rules: {
    "no-console": "off",
    "no-unused-vars": "warn",
    "prettier/prettier": "error"
  }
 },
 {
  languageOptions: { globals: globals.node }
 },
 pluginJs.configs.recommended
];
```

- **Configuração do Prettier**

Crie o arquivo .prettierrc com o seguinte conteúdo:

```
{
  "singleQuote": true,
  "trailingComma": "es5",
  "tabWidth": 2,
  "semi": true
}
```

Crie o arquivo .prettierignore com o seguinte conteúdo:

```
node_modules/
public/
views/
```

- **Configuração do ESLint Ignore**

Crie o arquivo .eslintignore com o seguinte conteúdo:

```
node_modules/
public/
views/
```

4. Scripts no package.json

Adicione os seguintes scripts no package.json:

```
"scripts": {
  "start": "node ./bin/www",
  "dev": "nodemon ./bin/www",
  "lint": "eslint . --ext .js",
  "format": "prettier --write ."
}
```

5. Criação de Pastas Dentro do Projeto

Crie as pastas dentro de libraryapi:

mkdir "config", "controllers", "middlewares", "models"

6. Criar a Estrutura de Pastas Fora de libraryapi

1. Crie a pasta librarydocs:

cd ..

mkdir "librarydocs"

2. Dentro de librarydocs, crie as pastas Models, Sql e UseCase:

cd librarydocs

mkdir "Models", "Sql", "UseCase"

3. Dentro da pasta Models, crie as subpastas classes, database e mindmap:

cd Models

mkdir "classes", "database", "mindmap"

7. Criar .gitkeep para Manter as Pastas Vazias no Push

1. Crie o script PowerShell para gerar os arquivos .gitkeep:

cd ../../

```
$diretorios = @(
   "config",
   "models",
   "controllers",
   "middlewares",
   "librarydocs",
```

```
    "librarydocs/Sql",
    "librarydocs/UseCase",
    "librarydocs/Models/classes",
    "librarydocs/Models/database",
    "librarydocs/Models/mindmap"
)

foreach ($diretorio in $diretorios) {
    $gitkeepPath = Join-Path -Path (Get-Location) -ChildPath "$diretorio/.gitkeep"

    if (-not (Test-Path -Path $gitkeepPath)) {
        New-Item -Path $gitkeepPath -ItemType File -Force
    }
}
```