



# Introdução ao Design System, Storybook e criação de uma biblioteca de componentes

🕒 Created	@October 18, 2022 10:07 PM
🏷️ Tags	Workshop

▼ **Data**

Porta-voz: Alexandre Gonçalves & Humberto Vieira

Duração: 30 minutos

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4587ec79-ad9f-4105-8cb2-7f0ce0826228/Workshop\\_DesignSystem\\_Storybook\\_BibliotecaComponentes.pptx](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4587ec79-ad9f-4105-8cb2-7f0ce0826228/Workshop_DesignSystem_Storybook_BibliotecaComponentes.pptx)

▼ **Design system:**

Este é um sistema unico de todos os conteúdos relacionados com design de elementos, componentes. Contém as regras e princípios para o desenvolvimento dos mesmos.

A ideia por detrás da criação deste sistema é facilitar o trabalho tanto para os desenvolvedores tanto para os designers para além de criar uma identidade unificada para aplicação.

Os benefícios do uso de um design system são bastante visíveis, um deles sendo a consistência na criação de interfaces visto que os elementos de UI estão padronizados, com isto, a aplicação tem uma maior qualidade aprimorando a user experience e diminuindo em erros e bugs.

Os Design Systems funcionam também como fonte da verdade, isto é, discussões entre desenvolvedores e designers tendem a diminuir uma vez que todas as regras estão escritas e oficializadas.

E claro, visto que todos os componentes já se encontram criados, o desenvolvimento de interfaces acaba por se tornar bem mais ágil, movendo o foco do desenvolvedor maioritariamente para o UX da aplicação.

▼ **Storybook**

É uma ferramenta para o desenvolvimento de interfaces, assim como referido anteriormente torna o desenvolvimento mais rápido e fácil isolando os componentes.  
Permite o desenvolvimento e manutenção de cada componente sem necessidade de iniciar uma aplicação complexa ou ler dados de algum lado.

Ajuda também na documentação dos vários componentes e visualização de testes para prevenir bugs. O Storybook suporta um ecossistema gigante de addons que ajudam por exemplo com a criação de layouts responsivos e verificação de acessibilidade.

▼ **Setup - Create a React Component Library**

Começar um projeto vazio e correr o comando `npm init`

Criar uma pasta `/src` na **root do projeto**.

Na pasta `/src` vamos criar o ficheiro `index.js` e a pasta `/src/components`

Instalar o **react e o react-dom como devDependencies**. `npm i --save-dev react react-dom`

Colocar o **react e o react-dom como peerDependencies**

```
"peerDependencies": {
  "react": "^18.2.0",
  "react-dom": "^18.2.0"
}
```

Para instalar o storybook basta correr o comando `npx sb init` , visto que temos o react instalado como dependencia o Storybook vai detetar que estamos dentro de uma aplicação de React.

Após isto, temos que **criar dentro da `/src/components` os componentes que queremos exportar.**

Vamos no ficheiro `index.js` dentro da pasta `/src` exportar todos os nossos componentes.

```
export * from './components';
```

Assim que tiveres a base do projeto montada com uma lista de componentes vamos utilizar o rollup.js para dar bundle ao nosso código.

**Corremos o seguinte comando na consola:** `npm i rollup rollup-plugin-babel @rollup/plugin-node-resolve rollup-plugin-peer-deps-external rollup-plugin-postcss @babel/preset-react @babel/preset-typescript @rollup/plugin-typescript @rollup/plugin-commonjs rollup-plugin-dts rollup-plugin-terser --save-dev`

**rollup/plugin-node-resolve:** Encontra e da bundle a dependencias de terceiros em node\_modules

**rollup/plugin-babel:** Integração com a existente configuração do `Babel`

**rollup/plugin-peer-deps-external:** Retira as `peerDependencies` do bundle

**rollup-plugin-postcss:** Integração com o `postCSS`

**@babel/preset-react:** `Babel` Preset para todos os plugins do `React`

**@babel/preset-typescript:** Preset para todos os plugins do `Typescript`

**@rollup/plugin-typescript:** Integração do `rollup` com o `typescript`

**@rollup/plugin-commonjs:** Converte `commonJS` modules para `ES6`

**rollup-plugin-dts:** Plugin para dar bundle aos ficheiros `.d.ts`

**rollup-plugin-terser:** Plugin para cumpririr bundles es gerados.

**Nota: Lembrando que alguns destes plugins e presets são opcionais.**

Na root do projeto criamos o ficheiro `rollup.config.js` e fazemos as seguintes alterações:

```
import babel from 'rollup-plugin-babel';
import resolve from '@rollup/plugin-node-resolve';
import external from 'rollup-plugin-peer-deps-external';
import postcss from 'rollup-plugin-postcss';
import typescript from '@rollup/plugin-typescript';
import commonjs from '@rollup/plugin-commonjs';
import dts from 'rollup-plugin-dts';
import { terser } from 'rollup-plugin-terser';

const packageJson = require('./package.json');

export default [
  {
    input: './src/index.js',
    //exports as commonJS and ESMODULES
    //Each item of the following array will be a different configuration for each type of output
    output: [
      //commonJS Configuration
      {
        file: packageJson.main,
        format: 'cjs',
        sourcemap: true,
      },
      //ES6 Configuration
      {
        file: packageJson.module,
        format: 'esm',
        sourcemap: true,
      },
    ],
    plugins: [
      postcss({
        plugins: [],
```

```
    minimize: true,
  }),
  typescript({ tsconfig: './tsconfig.json' }),
  babel({
    exclude: 'node_modules/**',
    presets: ['@babel/preset-react', '@babel/preset-typescript'],
  }),
  external(),
  resolve({
    extensions: ['.js', '.jsx', '.json', '.ts', '.tsx'],
  }),
  commonjs(),
  terser(),
],
},
{
  input: 'dist/esm/index.d.ts',
  output: [{ file: 'dist/index.d.ts', format: 'esm' }],
  plugins: [dts()],
},
];
```

Na **root do projeto** vamos criar um ficheiro `tsconfig.json` e configura-lo da seguinte forma:

```
{
  "compilerOptions": {
    "skipLibCheck": true,
    "jsx": "react",
    "module": "ESNext",
    "declaration": true,
    "declarationDir": "type",
    "sourceMap": true,
    "outDir": "dist",
    "moduleResolution": "node",
    "allowSyntheticDefaultImports": true,
    "emitDeclarationOnly": true
  }
}
```

Criar um script no ficheiro `package.json` para correr o rollup.

```
"scripts": {
  "rollup": "rollup -c"
},
```

O próximo step é organizar o `package.json` na **root do projeto**.

```
{
  "name": "PACKAGE_NAME",
  "version": "0.0.1",
  "description": "A simple react component library built for a workshop presented at @ITSector",
  "scripts": {
    "storybook": "start-storybook -p 6006",
    "build-storybook": "build-storybook",
    //script para rodar o rollup
    "rollup": "rollup -c"
  },
  "author": "Alexandre Gonçalves",
  "license": "MIT",
  "devDependencies": {
    "@babel/core": "^7.20.2",
    "@babel/preset-react": "^7.18.6",
    "@babel/preset-typescript": "^7.18.6",
    "@radix-ui/react-checkbox": "^1.0.1",
    "@radix-ui/react-slot": "^1.0.1",
    "@rollup/plugin-commonjs": "^23.0.2",
    "@rollup/plugin-node-resolve": "^15.0.1",
    "@rollup/plugin-typescript": "^9.0.2",
    "@storybook/addon-actions": "^6.5.13",
    "@storybook/addon-essentials": "^6.4.22",
    "@storybook/addon-interactions": "^6.5.13",
    "@storybook/addon-links": "^6.5.13",
    "@storybook/builder-webpack4": "^6.5.13",
    "@storybook/manager-webpack4": "^6.5.13",
    "@storybook/react": "^6.1.21",
    "@storybook/testing-library": "^0.0.13",
    "autoprefixer": "^10.4.13",
    "babel-loader": "^8.3.0",
    "clsx": "^1.2.1",
    "phosphor-react": "^1.4.1",
```

```
    "postcss": "^8.4.18",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "rollup": "^2.79.1",
    "rollup-plugin-babel": "^4.4.0",
    "rollup-plugin-dts": "^4.0.1",
    "rollup-plugin-peer-deps-external": "^2.2.4",
    "rollup-plugin-postcss": "^4.0.2",
    "rollup-plugin-tailwindcss": "^1.0.0",
    "rollup-plugin-terser": "^7.0.2",
    "tailwindcss": "^3.2.2",
    "typescript": "^4.8.4"
  },
  "peerDependencies": {
    "@radix-ui/react-checkbox": "^1.0.1",
    "@radix-ui/react-slot": "^1.0.1",
    "autoprefixer": "^10.4.13",
    "clsx": "^1.2.1",
    "phosphor-react": "^1.4.1",
    "postcss": "^8.4.18",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "tailwindcss": "^3.2.2",
    "typescript": "^4.8.4"
  },
  "main": "dist/cjs/index.js",
  "module": "dist/esm/index.js",
  "files": [
    "dist"
  ],
  "types": "dist/index.d.ts",
}
```

Corremos o comando `npm run rollup` na consola, o objetivo é ver a pasta dist ser criada.

Para publicar o nosso package temos duas opções:

▼ **Opção 1.**

- 1. Precisamos de alterar o ficheiro `.npmrc` na root da nossa machine (cd ~) para o seguinte formato:

```
registry=https://registry.npmjs.org/
@YOUR_GITHUB_USERNAME:registry=https://npm.pkg.github.com/
//npm.pkg.github.com/:_authToken=YOUR_AUTH_TOKEN
```

- 2. Dentro do `package.json` na root o name do projeto deve ser `@GITHUB_NAME/REPO_NAME`
- 3. Adicionar a seguinte informação ao `package.json`

```
{
  "publishConfig": {
    "access": "public",
    "registry": "https://npm.pkg.github.com/github_name"
  }
}
```

- 4. Correr o comando `npm publish` na consola.

▼ **Opção 2.**

- 1. Correr o comando `npm login` na consola e fazer o login com a conta no website [npmjs.org](https://npmjs.org)
- 4. Correr o comando `npm publish` na consola.

Ou npm login


▼ **Sources/Credits:**

Design System: Como Funciona e Por Que Usá-lo em UX Design?

Um dos grandes desafios do passado era conseguir produzir designs com velocidade e com capacidade escalável. Para tanto, uma ferramenta chamada Design System foi criada para atender essa necessidade. Mas será que esse conceito é tão simples assim? Confira este artigo e entenda <https://aelaschool.com/designdeinteracao/design-system-como-funciona-e-por-que-usa-lo/>




Storybook in 100 Seconds

 [https://www.youtube.com/watch?v=gdITFPebzAU&ab\\_channel=Fireship](https://www.youtube.com/watch?v=gdITFPebzAU&ab_channel=Fireship)



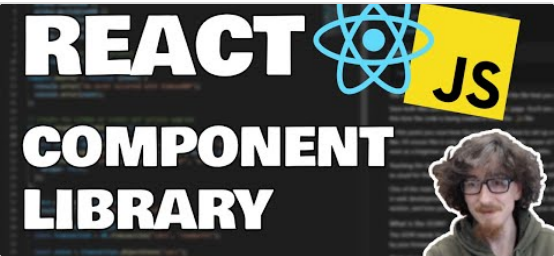
Build And Publish A React Component Library

 [https://www.youtube.com/watch?v=hf6Z8OZanec&t=819s&ab\\_channel=PortEXE](https://www.youtube.com/watch?v=hf6Z8OZanec&t=819s&ab_channel=PortEXE)



How to Create and Publish a React Component Library

 [https://www.youtube.com/watch?v=XHQi5a0TmMc&ab\\_channel=AlexEagleson](https://www.youtube.com/watch?v=XHQi5a0TmMc&ab_channel=AlexEagleson)



What are peer dependencies in a Node module?


In some package.json files, you might see a few lines like this: You might have already seen dependencies and devDependencies, but not peerDependencies. dependencies are the packages your project depends on. devDependencies are the packages that are needed during the

 <https://flaviocopes.com/npm-peer-dependencies/>



Technology Consulting Google Slides Theme and PPT Template

If you want to attract new clients to your technology company and to keep them satisfied, design your own consulting sales pitch with these minimalistic slides. This Consulting Sales Pitch presentation is pretty cool. The slides are white, and we have included several illustrations of people

 <https://slidesgo.com/theme/technology-consulting>

