Webpack: Manipulando módulos na sua webapp

- Esse curso capacita a:
  - Aprender o module bundler mais popular do mercado;
  - Diferenciar o build de desenvolvimento do build de produção;
  - Aprender a aplicar técnicas como o lazy loading e o code splitting;
  - Ser mais produtivo com Webpack Dev Server;
  - Separar a aplicação em diferentes bundles;
  - Aplicar na prática as boas práticas seguidas pela comunidade;

Aulas:

1. **Introdução:**
   - Webpack dispensa a utilização de um module loader, justamente por criar em bundles em tempo de desenvolvimento, que nada mais são do que scripts que agregam outros módulos da aplicação;
     - O papel do webpack;
     - Instalação através do npm;
     - Configuração do webpack.config.js;
     - Como executar webpack através de um npm script;
     - O conceito de entry e output;
     - O papel de um loader;
     - Instalação e configuração de um loader;

2. **Preparando o build de produção:**
     - O efeito do parâmetro **–p** para o build de produção;
     - A incompatibilidade do UglifyJS com código que não sejam escritos em ECMASCRIPT 5;
     - babili como plugin que ajuda no processo de minificação;
     - Pegadinhas na atribuição de variáveis de ambiente;
     - O módulo cross-env para garantir compatibilidade do nosso npm script entre diferentes sistemas operacionais;

3. **Webpack Dev Server e configuração:**
     - O papel do Webpack Dev Server e suas vantagens como live reloading;
     - Como instalar o Webpack Dev Server através do npm;
     - A criação de um script para executar o servidor;
     - A importância da propriedade publicPath;

4. **Tratando arquivos CSS como módulos:**
   - Podemos usar o npm da própria plataforma Node.js para gerenciar as dependências de frontend da nossa aplicação. Nesse sentido, Webpack será o responsável pelo carregamento dessas dependências adicionando-as no bundle da aplicação tratando-os como módulos, contanto que haja um loader configurado para lidar com o tipo de arquivo em questão;

- Como utilizar o npm para gerenciar nossas dependências frontend;
- Como o Webpack lida com as dependências em node_modules adicionando-as no bundle da aplicação;
- O papel de loaders;
- Que o padrão é adicionar no bundle scripts e CSS's;
- Que podemos separar CSS's do bundle criado e importa-los através da tag link através do módulo **extract-text-webpack-plugin**;
- A utilizar o plugin **optimize-css-assets-webpack-plugin** para minificar CSS's importados se adicionados no style.css;

5. **Importando scripts:**
   - Que o Webpack importa scripts através da instrução import sem muito mistério;
   - A necessidade de utilizar o webpack.ProvidePlugin;

6. **Boas práticas:**
   - Otimização com scope hoisting;
   - Separação do nosso código das bibliotecas com o CommonsChunkPlugin;
   - Geração de index.html automaticamente com todos os artefatos produzidos pelo Webpack já importados;
   - Code splitting e Lazy loading;
   - Sobre utilizar System.import() ou import no carregamento de módulos;
   - Onde ficam os arquivos para distribuição do projeto;
   - Como alterar o endereço da API no build de produção com o **DefinePlugin**;