

Curso Técnico Superior Profissional em Tecnologias e Programação de Sistemas de Informação

Unidade Curricular: Desenvolvimento Web – Front-End

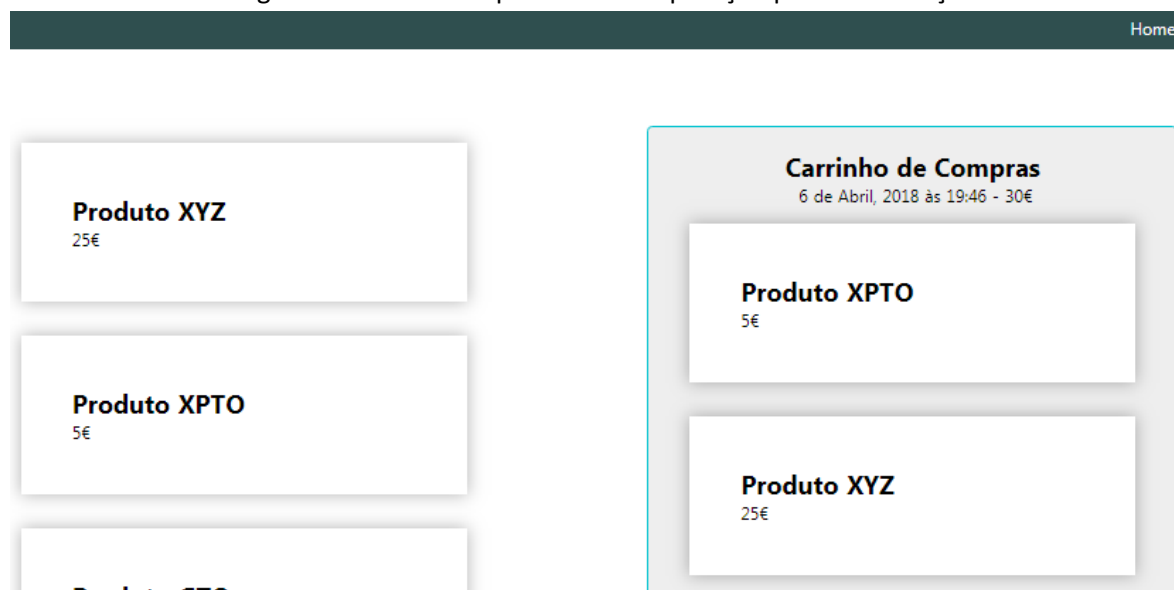
1º Ano/2º Semestre

Docente: Marco Miguel Olival Olim

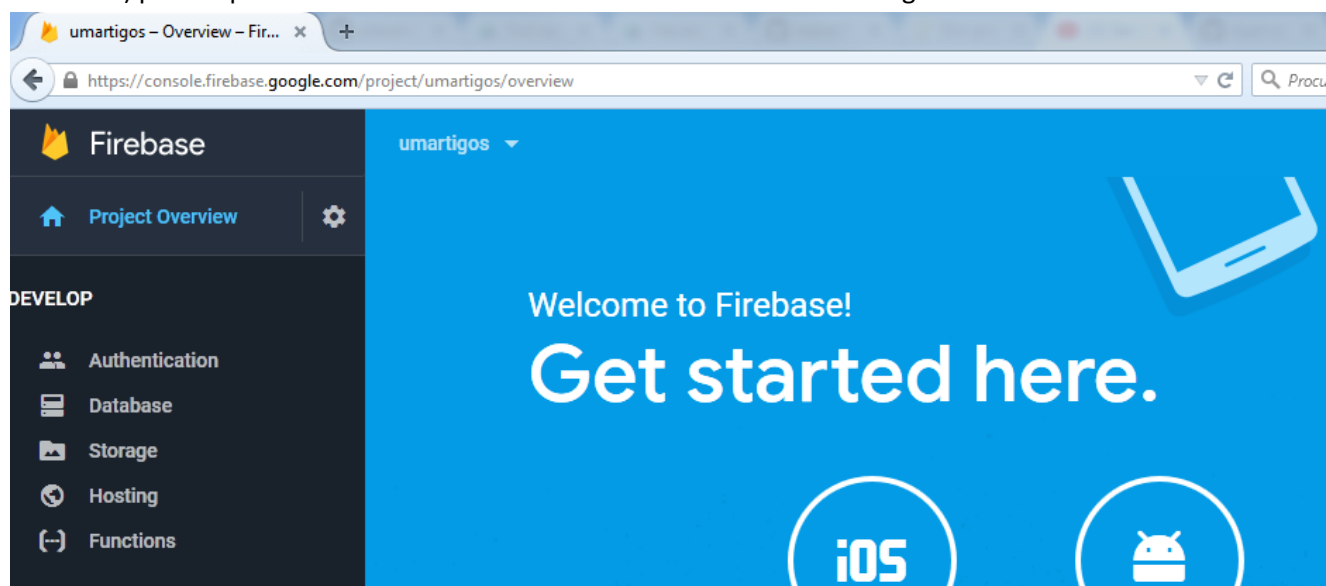
Data 20/04/2018

ESTE EXERCÍCIO ABORDA A PUBLICAÇÃO DE APLICAÇÕES WEB PARA A CLOUD

Após a conclusão do exercício anterior obtivemos uma aplicação web funcional, embora só em ambiente de desenvolvimento. Seguidamente iremos publicar esta aplicação para um serviço de Cloud:






Atendendo a que os dados são fornecidos pelo **Firebase**, iremos também utilizar o serviço de hosting deste, além de que o Firebase também nos disponibiliza um servidor de NodeJS como serviço (designado por *Cloud Functions*) para implementarmos a funcionalidade de Server-Side Rendering do NUXT:

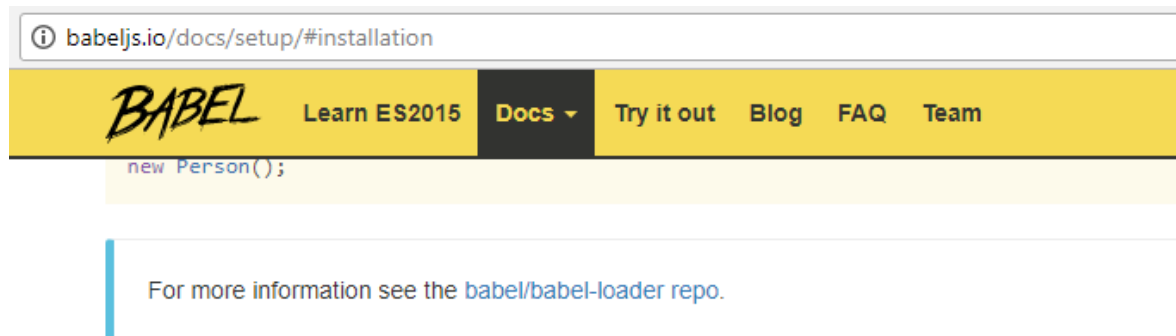


Cofinanciado por:

Um projeto de Cloud no firebase obedece a uma estrutura específica de diretórios para ser publicado. Terá por isso de criar uma nova pasta e copiar o seu projeto para esta, alterando o nome para **src**. Depois terá ainda de criar as pastas **public** e **functions**:

 functions	15-04-2018 08:21	Pasta de ficheiros
 public	15-04-2018 00:41	Pasta de ficheiros
 src	16-04-2018 20:20	Pasta de ficheiros

Infelizmente o NUXT utiliza a versão 8 do nodeJS, enquanto que o Firebase Functions é baseado na versão 6. Além disso, como utilizamos notação de ES2015 no código da nossa aplicação, estamos a limitar o acesso apenas a browsers mais recentes. Para ultrapassarmos estes constrangimentos podemos simplesmente converter (*transpile*) todo o código para uma notação compatível. Usamos o Babel para este efeito:



4 Create **.babelrc** configuration file

Great! You've configured Babel but you haven't made it actually *do* anything. Create a **.babelrc** config
To start, you can use the **env preset**, which enables transforms for ES2015+

Shell

```
npm install babel-preset-env --save-dev
```

In order to enable the preset you have to define it in your **.babelrc** file, like this:

JSON

```
{
  "presets": ["env"]
}
```

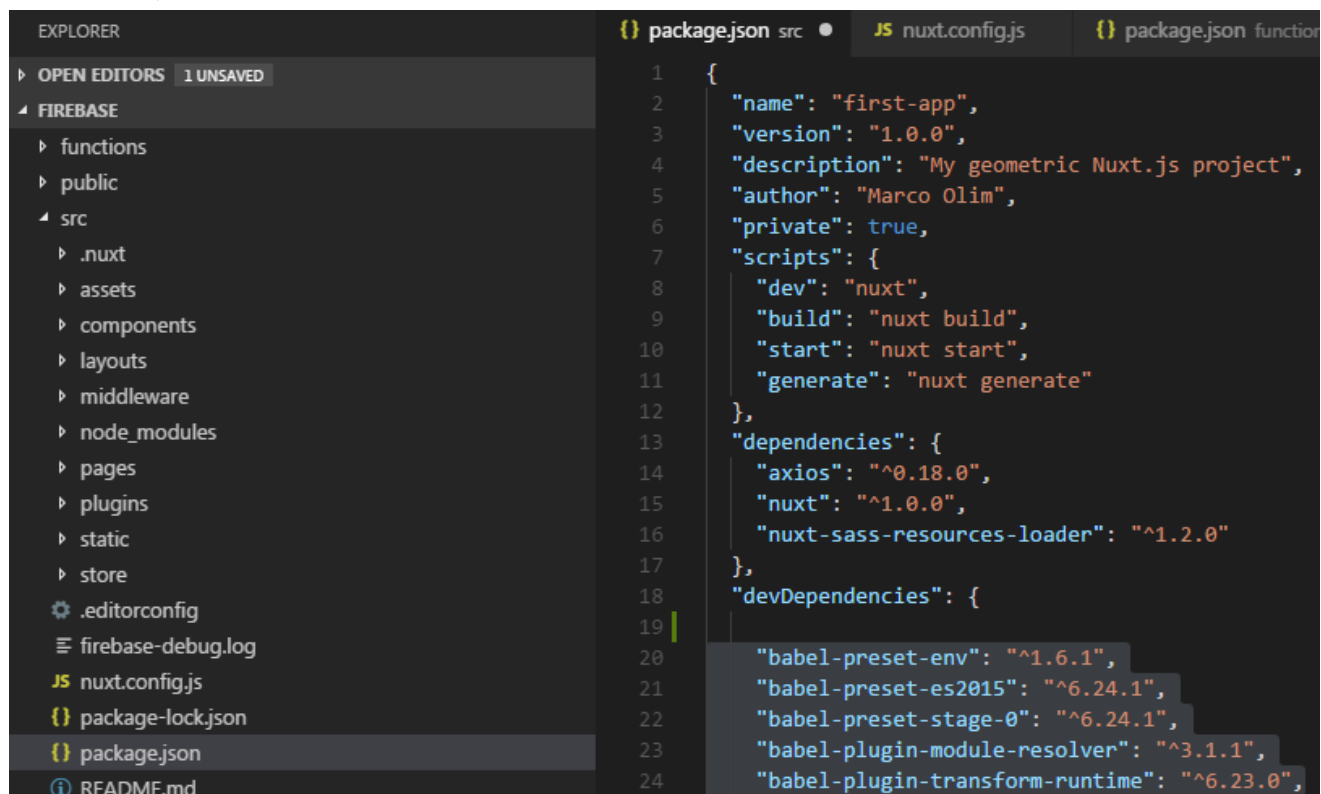
Em vez de instalarmos os ficheiros do Babel necessários à nossa aplicação um de cada vez com o processo acima indicado, podemos simplesmente editar o **package.json**, acrescentar primeiro todos os ficheiros e correr **npm install** apenas uma vez:

```
"babel-preset-env": "^1.6.1",
"babel-preset-es2015": "^6.24.1",
"babel-preset-stage-0": "^6.24.1",
"babel-plugin-module-resolver": "^3.1.1",
"babel-plugin-transform-runtime": "^6.23.0",
```

Cofinanciado por:




Sendo assim, os ficheiros Babel a acrescentar são:



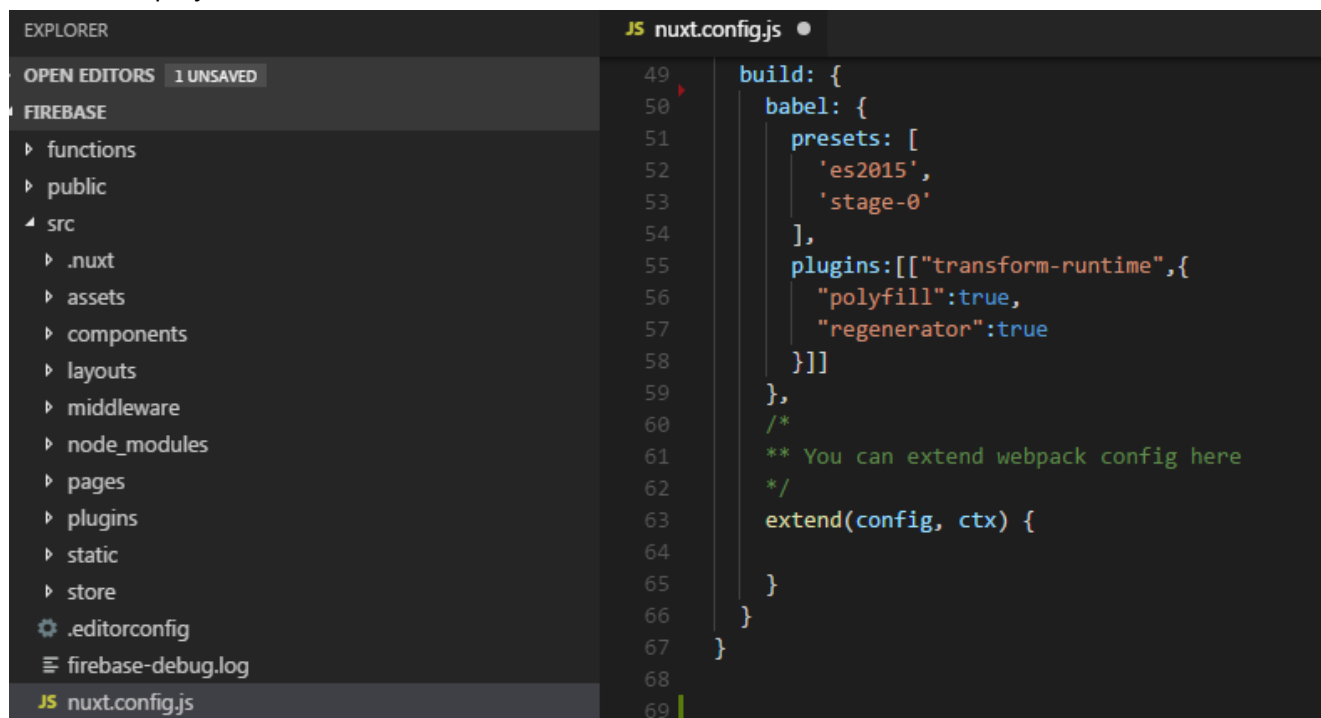
```
{
  "name": "first-app",
  "version": "1.0.0",
  "description": "My geometric Nuxt.js project",
  "author": "Marco Olim",
  "private": true,
  "scripts": {
    "dev": "nuxt",
    "build": "nuxt build",
    "start": "nuxt start",
    "generate": "nuxt generate"
  },
  "dependencies": {
    "axios": "^0.18.0",
    "nuxt": "^1.0.0",
    "nuxt-sass-resources-loader": "^1.2.0"
  },
  "devDependencies": {
    "babel-preset-env": "^1.6.1",
    "babel-preset-es2015": "^6.24.1",
    "babel-preset-stage-0": "^6.24.1",
    "babel-plugin-module-resolver": "^3.1.1",
    "babel-plugin-transform-runtime": "^6.23.0",
```

De seguida, instalar todos de uma vez com:

 Linha de comandos

```
C:\Users\Olim\Desktop\nuxt\firebase\src>npm install
```

Agora temos de editar o ficheiro de configuração do NUXT (**nuxt.config.js**) para definirmos a conversão dos ficheiros do projeto com o Babel:



```
build: {
  babel: {
    presets: [
      'es2015',
      'stage-0'
    ],
    plugins: [
      ["transform-runtime", {
        "polyfill": true,
        "regenerator": true
      }]
    ]
  },
  /*
  ** You can extend webpack config here
  */
  extend(config, ctx) {}
}
```

Cofinanciado por:



Neste mesmo ficheiro definimos também onde será compilado o projeto e qual o ponto de acesso público do nosso site:

```
/*
** Build configuration
*/
buildDir: '../functions/nuxt',
build: {
  publicPath: '/public/',
  vendor: ['axios'],
  extractCSS: true,
  babel: {
    presets: [
      'es2015',
      'stage-0'
    ],
    plugins: ["transform-runtime", {
      "polyfill": true,
      "regenerator": true
    }]
  },
},
```

Se neste momento compila-se-mos o projeto com **npm run build** obteríamos um erro resultante do facto de não existirem ainda os pacotes npm deste projeto (pasta node_modules) no diretório de build (pasta functions), como por exemplo o transform-runtime que aparece neste log de erros:

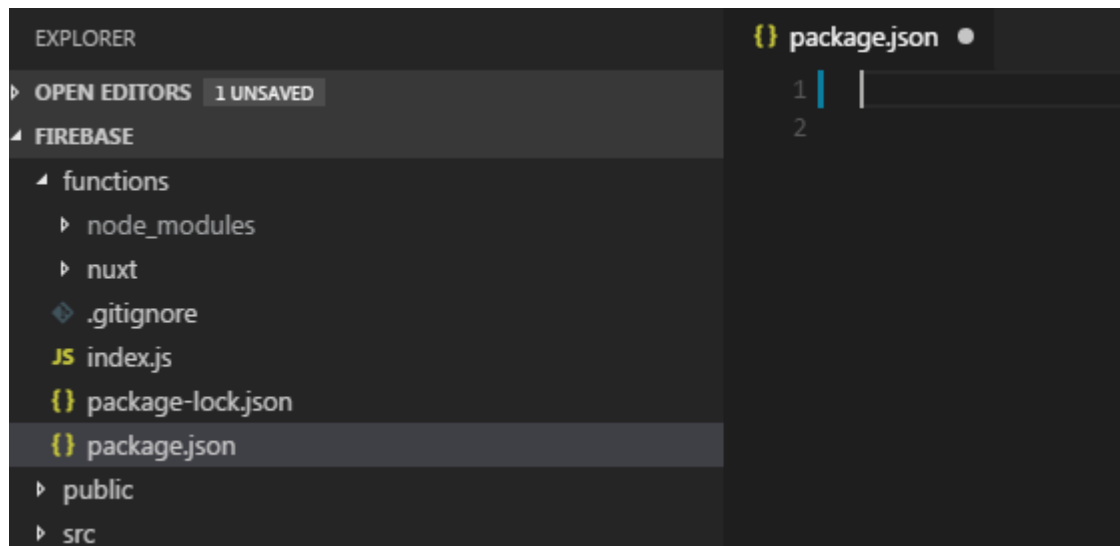
```
32 ],
33 plugins: [
34   ["transform-runtime", {
35     "polyfill": true,
36     "regenerator": true
37   }],
38 ]
39 },
40 /*
41 ** Run ESLint on save
42 */
```

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL
app.401fd82e668fa25d53f0.js	1.54 kB	1 [emitted]	app
manifest.ba0573df2b3dcdf0bbb4.js	1.51 kB	2 [emitted]	manifest
LICENSES	269 bytes	[emitted]	
+ 6 hidden assets			
ERROR in ../functions/nuxt/client.js			
Module build failed: ReferenceError: Unknown plugin "transform-runtime" specified in "base" at 0, attempted to load.			
at /Users/deast/Documents/nuxt/nuxt-ssr/src/node_modules/babel-core/lib/transformation/file/options/normalizePluginList.js:105:13			
at Array.map (native)			
at Function.normalizePlugins (/Users/deast/Documents/nuxt/nuxt-ssr/src/node_modules/babel-core/lib/transformation/file/options/normalizePluginList.js:105:13)			

Cofinanciado por:



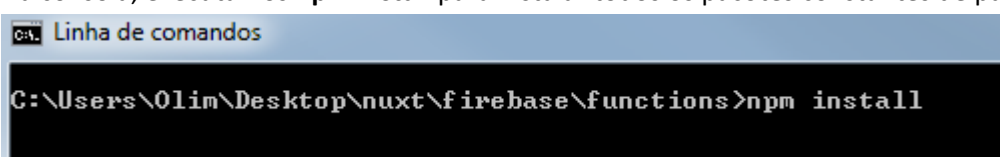
Sendo assim, na pasta **functions** criamos um novo ficheiro **package.json** de modo a usarmos a mesma estratégia que para a instalação do Babel:



Neste ficheiro enumeramos os ficheiros a instalar:

```
{
  "name": "functions",
  "description": "Cloud Functions for Firebase",
  "dependencies": {
    "babel-preset-env": "^1.6.1",
    "babel-preset-es2015": "^6.24.1",
    "babel-preset-stage-0": "^6.24.1",
    "babel-plugin-module-resolver": "^3.1.1",
    "babel-plugin-transform-runtime": "^6.23.0",
    "babel-runtime": "^6.23.0",
    "clone": "^2.1.1",
    "debug": "^3.1.0",
    "es6-promise": "^4.1.1",
    "express": "^4.15.4",
    "firebase-admin": "^5.0.1",
    "firebase-functions": "^0.6.1",
    "axios": "^0.18.0",
    "lodash": "^4.17.4",
    "nuxt": "1.0.0-rc11",
    "vue": "~2.4.2",
    "vue-meta": "^1.2.0",
    "vue-router": "^3.0.1",
    "vuex": "^3.0.0"
  },
  "private": true
}
```

Na consola, executamos **npm install** para instalar todos os pacotes constantes de package.json



Cofinanciado por:

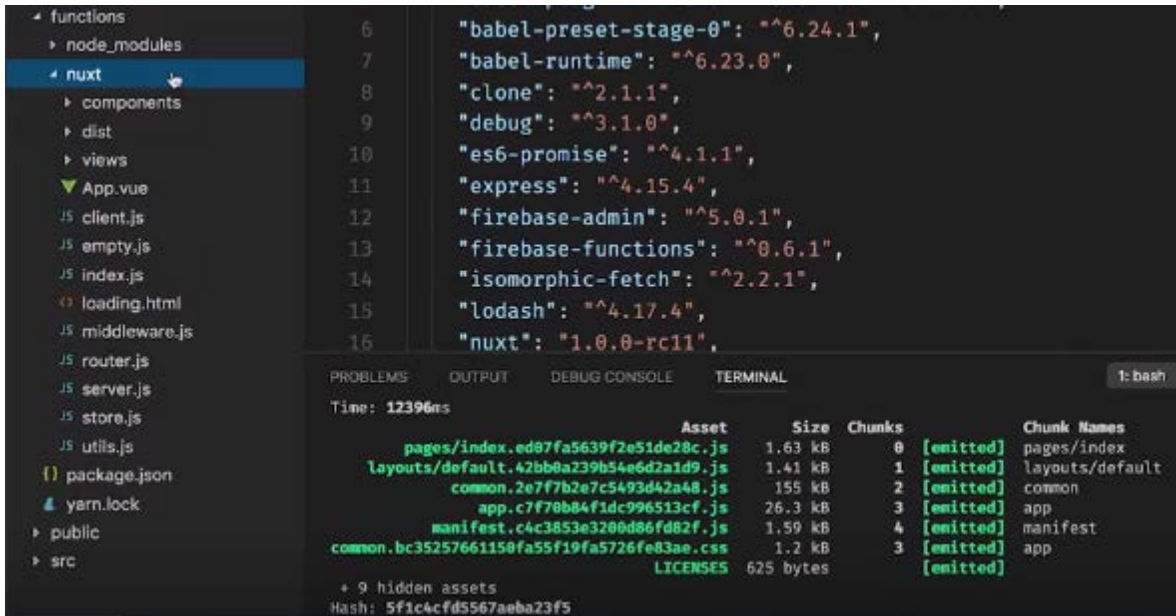


Após a instalação, voltamos a src para compilar o projeto

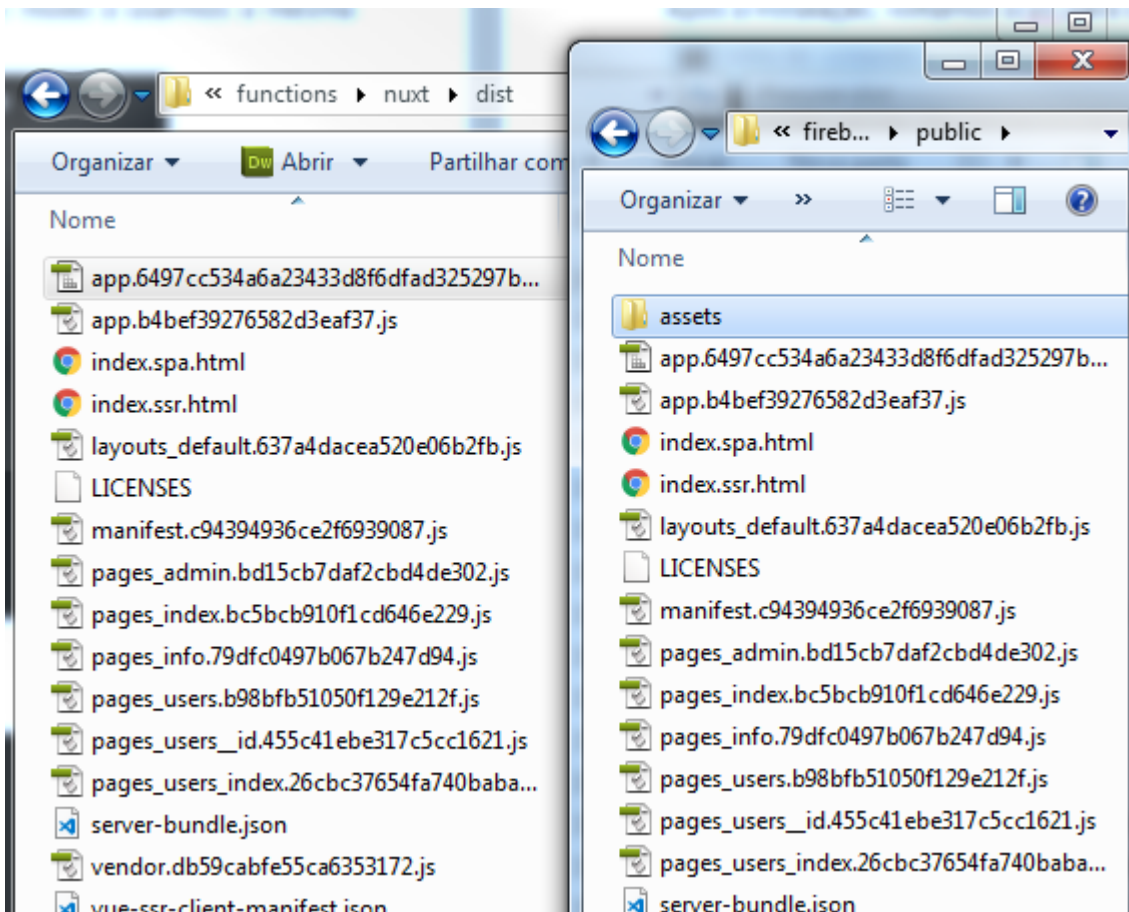
```
Ca. Linha de comandos

C:\Users\Olim\Desktop\nuxt\firebase\functions>cd..
C:\Users\Olim\Desktop\nuxt\firebase>cd src
C:\Users\Olim\Desktop\nuxt\firebase\src>npm run build
```

São então gerados os ficheiros finais da Aplicação Web, surgindo agora na pasta functions/nuxt os ficheiros para o servidor web, e na pasta functions/nuxt/dist os ficheiros públicos do site

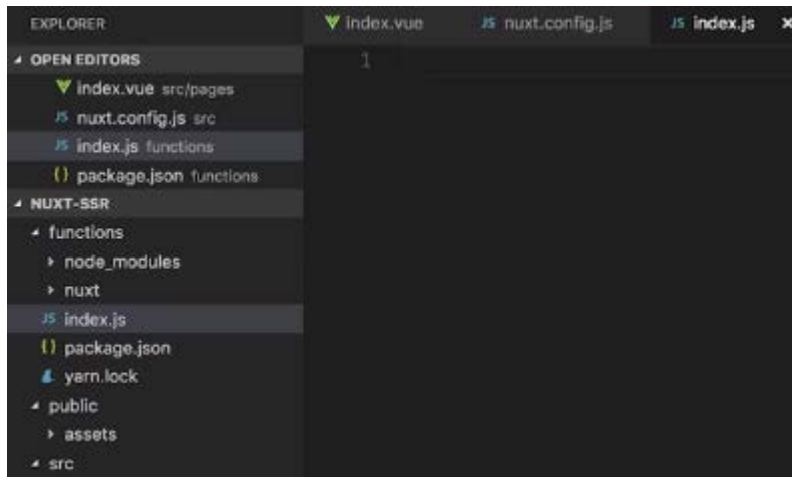


Temos, por isso, de copiar manualmente estes ficheiro de dist para a pasta public



Cofinanciado por:

De volta a functions vamos criar o ficheiro index.js que definirá o servidor de NodeJS



Basicamente este servidor, definido como ssrapp, indica ao Firebase como renderizar o html com o NUXT

```
const functions = require('firebase-functions');
const { Nuxt } = require('nuxt');
const express = require('express');

const app = express();

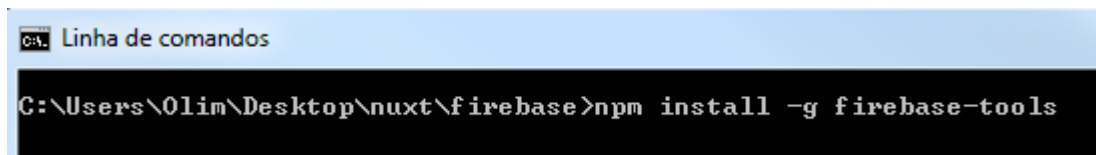
const config = {
  dev: false,
  buildDir: 'nuxt',
  build: {
    publicPath: '/public/'
  }
};

const nuxt = new Nuxt(config);

function handleRequest(req, res) {
  res.set('Cache-Control', 'public, max-age=150, s-maxage=150');
  return new Promise((resolve, reject) => {
    nuxt.render(req, res, promise => {
      promise.then(resolve).catch(reject)
    })
  });
}

app.use(handleRequest);
exports.ssrapp = functions.https.onRequest(app);
```

Instalamos agora uma aplicação do firebase para publicar o projeto



Cofinanciado por:




```
C:\Users\Olim\Desktop\nuxt\firebase>firebase init hosting
```



You're about to initialize a Firebase project in this directory:

```
/Users/deast/Documents/nuxt/nuxt-ssr
```

=== Project Setup

First, let's associate this project directory with a Firebase project. You can create multiple project aliases by running **firebase use --add**, but for now we'll just set up a default project.

```

Hosting Setup

Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? public
? Configure as a single-page app (rewrite all urls to /index.html)? No
✓ Wrote public/404.html
✓ Wrote public/index.html

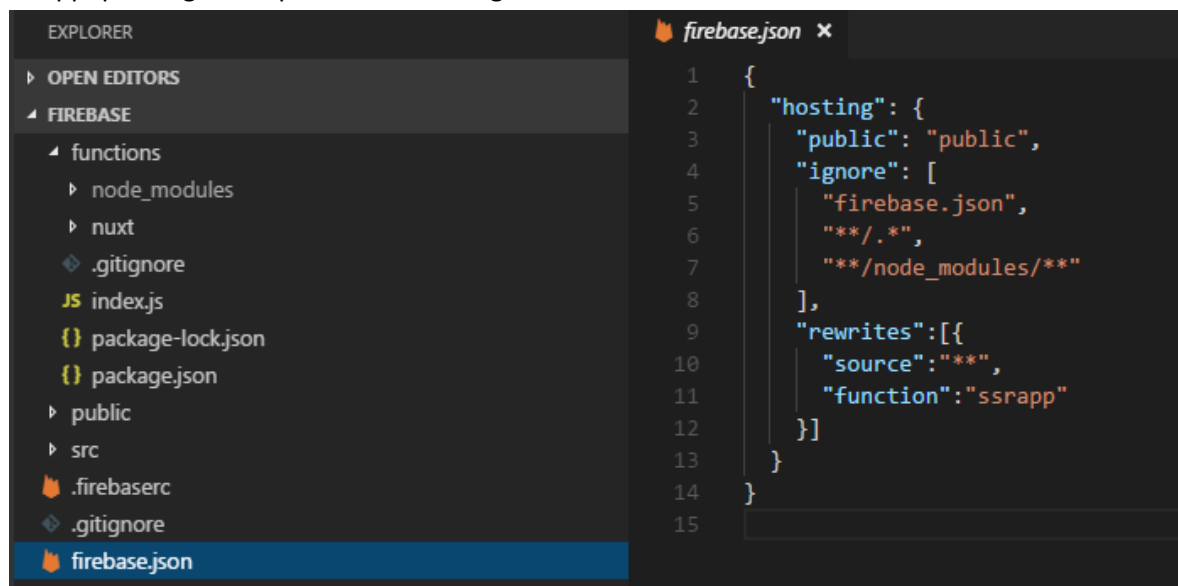
i Writing configuration info to firebase.json...
i Writing project information to .firebaserc...

✓ Firebase initialization complete!

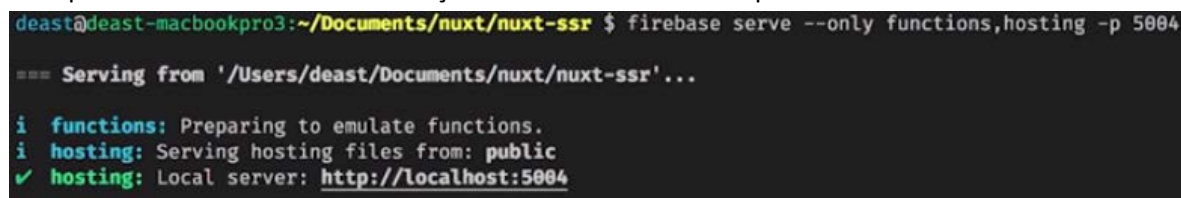
```

- public
 - assets
 - 404.html
 - index.html
 - src
 - .firebasesrc
 - firebase.json

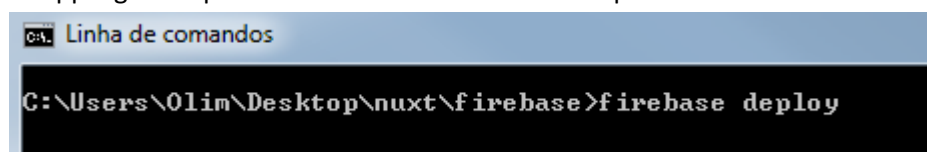
Foi também gerado um ficheiro `firebase.json` que temos de editar para indicar que será o servidor de nodeJS `ssrapp` que irá gerir os pedidos ao hosting do firebase



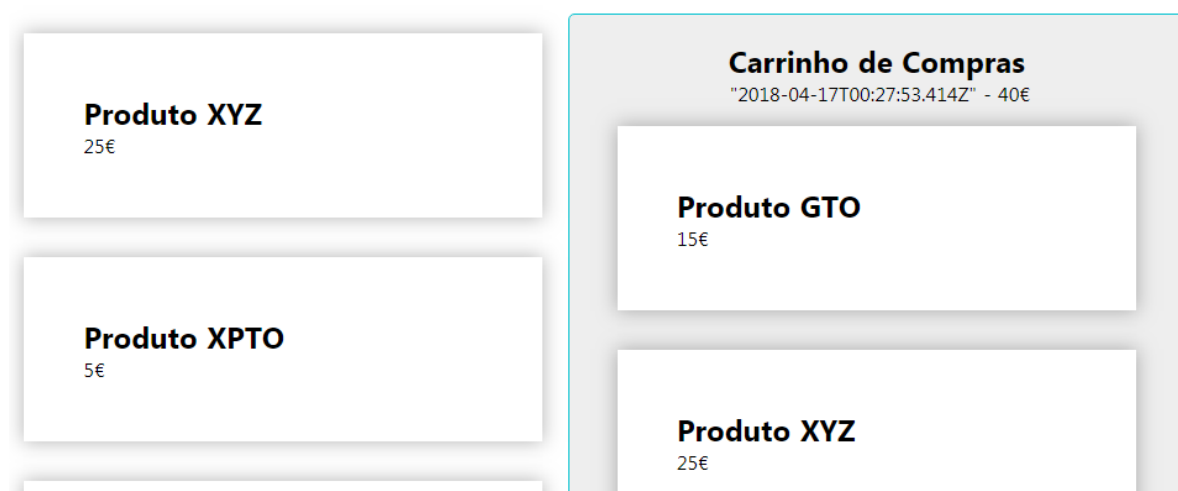
Há a possibilidade de emular o serviço de Firebase localmente para verificar a existência de erros:



Finalmente podemos publicar a nossa app com **firebase deploy**. Nesta fase é inicializada a app no hosting e a `ssrapp` é gerada para o functions. Na conclusão do processo é indicado o link para o site da app no firebase



Refira-se que como este hosting é baseado em CDN, pode demorar alguns segundos a propagar.



Cofinanciado por:

