ngx-admin Angular Frontend instruction

Intro

This is readme and instructions how start using Angular Frontend part of backend bundle from Akveo.
Backend bundle is integrated solution of Backend Code and Frontend code.
Backend code plays mostly API role, giving data to the client side as REST API.

Running Instruction

- run back-end part on your localhost and save the URL to access it's api. it could be, for example, http://localhost:3001/api (http://localhost:3001/api) or similar.
- make sure you have node v10 installed in your machine
- open `frontend/src/environments/environment.ts` and put correct API url for .NET. Please check the port number in preferences of backend project

```
  apiUrl: 'http://localhost:3001/api',
```

- open terminal and navigate to `frontend` directory
- run `npm install` command to install dependencies
- run `npm start` command to compile TypeScript code and start angular frontend
- once compiled, it will start own web server
- open web browser and navigate to url: `localhost:4200` - this is your frontend angular application running
- you can do angular code changes without closing this terminal session, Angular will check for changes and run re-compile only for changed modules, browser will update opened app automatically

Tech Stack

Frontend Part is an Angular project with following stack:

- Angular 8.0.0
- RxJs 6.5.2
- Nebular 4.1.2
- Eva-icons 1.1.0
- Typescript 3.4.5
- ...

Frontend part is based on the latest ngx-admin dashboard template, but with edited UI components and service layer for data getting. Bundle UI of full bundle (E-Commerce or IoT) supports both data from API and mock data, you can switch it inside file `core.module.ts` by editing `NB_CORE_PROVIDERS` collection.

Basic Code Structure

Code is organized in following structure

- frontend

- package.json
- angular.json
- src – *angular code here*
    - environments – *app settings are here*
    - app – *components and modules*
        - @auth – *authentication module, handling of security, tokens, roles*
        - @components – *reusable components*
        - @core – *backend access, wrappers over httpClient service calls*
        - @theme – *module* to setup nebular themes and styles, main visual components like header
        - pages – *module with all functional components*
        - app.component.ts – start component
        - app.module.ts – root angular module of the app
        - app-routing.module.ts – root routing module of the app

Demo Authentication

For demo purposes we added the code to put special JWT token into you local storage on application start. So you will be logged in without necessity to put email and password into login form. But you still can do it by logging out and registering new user if needed.

```
export class AppComponent implements OnInit {

  constructor(...) {
    // for demo only: init localstorage with token for demo user when login for
the first time
    this.initTestUserToken();
  }

  initTestUserToken() {
    const demoTokenInitKey = 'demo_token_initialized';
    const demoTokenWasInitialized = localStorage.getItem(demoTokenInitKey);
    const currentToken = this.tokenStorage.get();
    if (!demoTokenWasInitialized && !currentToken.isValid()) {
      // local storage is clear, let's setup demo user token for better demo
experience
      this.tokenStorage.set(this.authStrategy.createToken<NbAuthToken>
(environment.testUser.token));
      localStorage.setItem(demoTokenInitKey, 'true');
    }
  }
}
```

For production purposes, please remove function `initTestUserToken` from `AppComponent.ts`

Test User / Password

You can use these test users for application testing:

1. user@user.user / 12345
2. admin@admin.admin / !2e4S

Frontend Deployment

- Angular part can be deployed to any server or cloud platform as separate web application or just set of static JS files.
    - correct environment.prod.ts to setup actual api url (to deployed web api)
    - run `npm install` and `npm run build:prod` commands to get compiled angular javascript files
    - in case of deploying to azure, generate standard web.config file. See the sample in the end of this instruction
    - put angular output files (from folder 'dist') to the server

web.config sample

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>

<system.webServer>
  <rewrite>
    <rules>
      <rule name="Angular Routes" stopProcessing="true">
        <match url=".*" />
        <conditions logicalGrouping="MatchAll">
          <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />
          <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true"
/>
        </conditions>
        <action type="Rewrite" url="./index.html" />
      </rule>
    </rules>
  </rewrite>
  <staticContent>
        <mimeMap fileExtension=".json" mimeType="application/json" />
        <remove fileExtension=".woff"/>
        <mimeMap fileExtension=".woff" mimeType="application/font-woff" />
        <mimeMap fileExtension=".woff2" mimeType="application/font-woff2" />
      </staticContent>
</system.webServer>
</configuration>
```

Support

Please post issues in [Bundle Support Issue Tracker (https://github.com/akveo/ngx-admin-bundle-support/issues)](https://github.com/akveo/ngx-admin-bundle-support/issues)