Online shop app

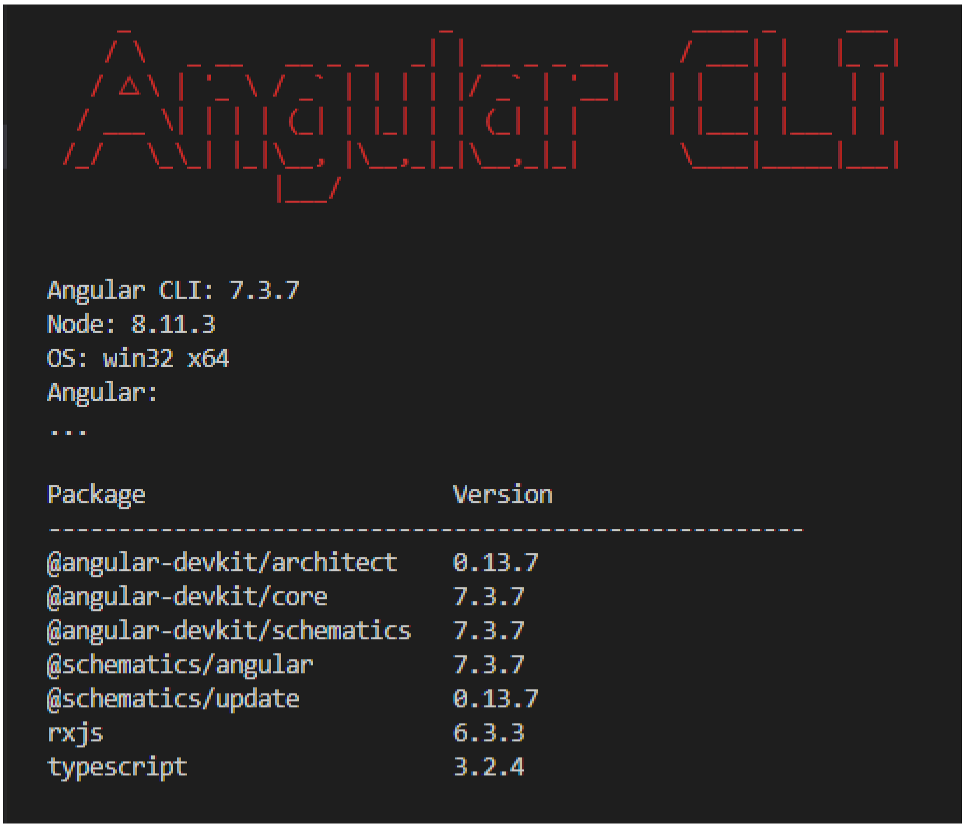
Loading on local machine

Back-end:

Project was created using Visual Studio Community 2017 15.9.9 with installed ASP.NET and web development elements. To host the project set as startup project in solution WebApi project(right click on project > Set as StartUp Project) and launch app(press Ctrl + F5).

Front-end:

Project is created using Angular with next specification



```
     _                      _                 ____ _     ___
    / \   _ __   __ _ _   _| | __ _ _ __     / ___| |   |_ _|
   / △ \ | '_ \ / _` | | | | |/ _` | '__|   | |   | |    | |
  / ___ \| | | | (_| | |_| | | (_| | |      | |___| |___ | |
 /_/   \_\_| |_|\__, |\__,_|_|\__,_|_|       \____|_____|___|
                |___/


Angular CLI: 7.3.7
Node: 8.11.3
OS: win32 x64
Angular:
...

Package                           Version
------------------------------------------------------
@angular-devkit/architect         0.13.7
@angular-devkit/core              7.3.7
@angular-devkit/schematics        7.3.7
@schematics/angular               7.3.7
@schematics/update                0.13.7
rxjs                              6.3.3
typescript                        3.2.4
```

To load project you need to:

1. Install Angular CLI ;
2. In project folder OnlineShop\OnlineShopAngular\OnlineShop call command **npm install**;
3. In same project folder (see previous step) call command **ng serve.**
4. Open project in browser(project is hosted on localhost:4200).

Description

Solution Hierarchy: This is a C# .Net solution-project, and it contains three sub-projects:

1. DAL – A class library project. Contains realization of mechanisms of accessing database using Entity Framework.

2. BLL – A class library project. Contains realization of provided services in the current app.

3. WebApi – Web API project. Contains realization and configuration of web API controllers which provide main functions of the app.

Web API:

Web API layer contains three API controllers:

1. OrdersController: Controller dedicated to work with orders and order stages.
   Routes description:
   - Get all orders: “api/orders”;
   - Get all order stages: “api/stages”;
   - Get stage by id: “api/orders/stages/{id}”.
2. ProductController: Controller dedicated to work with products.
   Routes description:
   - Get all products: “api/products”;
   - Get product by id: “api/product/{id}”.
3. ProductCategoriesController: Controler dedicated to work with product categories and their descriptions.
   Routes description:
   - Get all categories: “api/categories”;
   - Get category description: “api/categories/{id}/description”

Exception handling

Web API exception handling in the current project is managed using Exception Filter Attribute class.

Exception filters:

ValidateModelAttribute: Filter which handles errors related to model state validation.

FailedToFindExceptionAttribute: Filter which handles errors related to FailedToFindException which occurs on BLL layer.

Exception filters are set globally in WebApiConfig.cs, as a result each web API controller related to the current project can handle those exceptions.

Dependency injection

In the current project dependency injection is managed using Ninject. There are two ninject modules in the app:

- DalModule: manages all dependencies related to work with data base(IUnitOfWork, IOnlineDbContext and object repositories);
- BllModule: manages all dependencies related to work with business logic of the app (IOrdersService, IProductCategoriesService, IProductsService).