**Association App**

**Assignment**

Elaborate a web application that deals with the management of tenant’s associations and their details. The app should supply all the facilities of a physical association and should meet the customer needs, which in this case are clients or residents that pay taxes to an association.

**Introduction**

The main purpose of the application is the digitalization of information, a well-known process these days that helps people to reduce waiting time, costs, etc. and liberate paperwork and walking to the physical place.

The app should consist of a website that manages associations with their representatives, each having clients that have chosen providers for household use, such as water, electricity and gas. Association App also supplies with an archive, where are stored the payments issued, a payment section where the client can see what he has to pay, his backlogs and has an option to pay, which generates a receipt as a proof of payment. All of these facilities are under the supervision of an administrator user that have full control of the application.

Associations collaborate with providers giving the option to clients to choose exactly what provider they wand depending on their needs and offers. For example, a client can choose three different providers for water, electricity, respective gas, depending on the prices at their request.

While a client can register and login in order to see instantly their details and payments, a representative can also create client accounts, can see their details, emit invoices and update penalties of all the clients assigned to them and an admin can manage the users, associations, providers, payments and archives.

**Architecture**

The application is built with the REST architecture (Representational State Transfer), which consists of the separation of the front-end and back-end as two individual components that communicates with each other through a server using requests called by the front-end, that are awaiting a response from back-end.

For the backend is used ASP.NET Core Web Application API with Entity Framework and for the front end is used Angular.Js with Angular Material, Bootstrap and other modules used for aspect (toastr, animate.css).

**User Interface**

The application is user friendly and intuitive for a better experience:

* Forms are clear, with hints and specific warnings, if is the case
* Pages consists mainly of big buttons that open dialogs with the information needed
* Pages are not overwhelmed with information, only the desirable information is displayed at a time
* Application pages and dialogs are dynamic: they display information depending the user logged, his role or his chosen option (for example the register form is the same for every user but its content is dynamic: admin and representative have more options to fill: role, association, etc.)
* Polymorphism for pages, lists and forms, being called same component with different results

When a user enters the app as a guest he is redirected to an authentication page with two buttons for login and register. After a user is registered and logged it is redirected, with route guarding, to his specific page, meaning that a user cannot access functions that are not designated for him.

An admin has in his page more functions, bound to big text buttons and after pressing a button the application shows only the information regarding to his intention. It has buttons for lists, forms and other actions that manage the entire application

The representative page has a similar aspect, only the functions are limited to his permissions and facilities.

The client’s page is the simplest and the cleanest, because it has buttons for his details and payments.

Every page, except the authentication one, has a toolbar with dynamic information about the application, the user and a logout button.

**Application users (roles):**

* Admin
* Representative
* Client

**Product functions:**

* Register new users
* Login (Encryption available via HMAC authentication with SHA algorithm and token generation: “Bearer token”)
* Toolbar
  + Logged user details
  + Application details
  + Logout
* Admin
  + See lists of users
  + Add users
  + Delete users
  + See list of associations
  + Add association
  + Delete association
  + See list of providers
  + Add providers
  + Delete providers
  + See archive
  + See payments
  + Generate payment
  + See receipts
  + Update payments
  + Update backlogs
* Representative
  + See list of his clients
  + See his association details
  + See provider details
  + See his client’s payments
  + Generate payments
  + Update payments
  + Update backlogs
* Client
  + See his details
  + See association details
  + See provider details
  + Pay
  + Generate receipt
  + See receipts

**Database diagram**

