Database aplications (APBD)

Piotr Gago (pgago@pja.edu.pl)

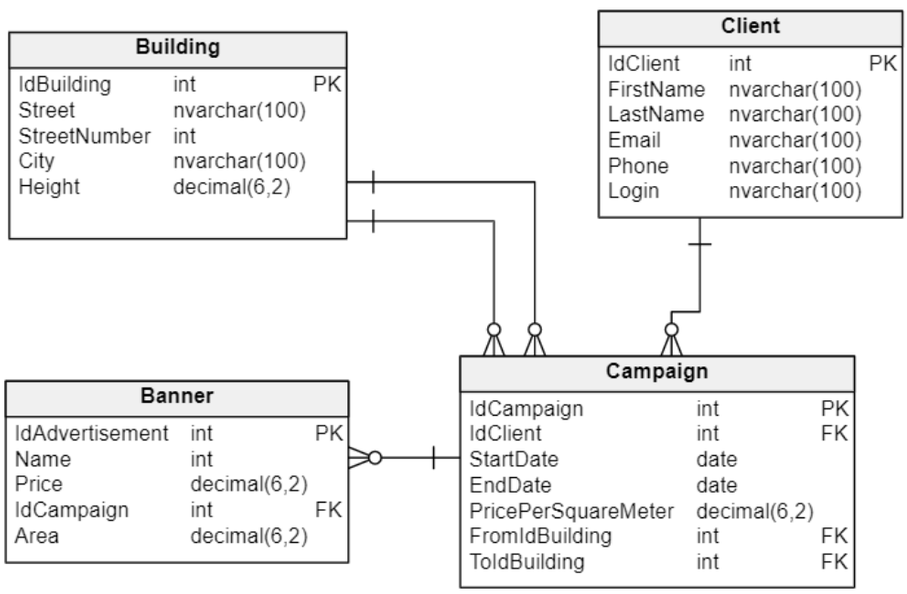
June 6, 2020

1 Project

This project is optional. The student who wants to complete the project should have at least grade 4.5. It is only required for people who would like to get a 5.

1.1 Description

We are preparing an application that is used for booking, buying and managing advertising space. The application stores information about buildings, advertising campaigns and customers. The following diagram represents the database.



1.2 Functional requirements

- Create a REST API application with the name AdvertAPI
- Create a database using EF and either CodeFirst or DB first.
- Remember about the correct naming of all variables, migrations, methods etc.
- Remember to separate a layer of communication with the database (separate DAL layer or service injected into the controller)

- Remember about correct model validation and error handling
- Properly use HTTP methods
- Prepare the appropriate DTO models (request, response).
- Prepare the endpoints described in the following requirements.
- Add automatically generated API documentation with the help of Swagger
- Prepare a separate project with unit and integration tests for prepared endpoints.

1.3 Endpoint for registering new users

Endpoint for registering new users. Remember to implement the password protection mechanism (hash and salting) correctly. After successful registration, return the appropriate access token and refresh token. If necessary, add the necessary columns to the User table.

Example of the request:

```
POST /api/clients HTTP/1.1
Host: localhost:54965
Content-Type: application/json

{
"FirstName": "John",
"LastName": "Kowalski",
"Email": "kowalski@wp.pl",
"Phone": "454-232-222"
"Login": "Jan125",
"Password": "asd124"
}
```

1.4 Refresh token endpoint

Prepare the endpoint that will accept the client's refresh token and on this basis will issue a new access token and refresh token.

1.5 Login endpoint

Prepare an endpoint that will allow the user to log in and get access token and refresh token.

```
POST /api/clients/login HTTP/1.1
Host: localhost:54965
Content-Type: application/json

{
"Login": "asd1245",
"Password": "AlaMaKota"
}
```

1.6 List of campaign

Prepare the endpoint available only for logged in users, which will allow to return a list of campaigns along with customer data and ads related to the campaign. Campaigns should be sorted in descending order after the campaign start date.

1.7 Create new campaign

The endpoint for creating new campaigns contains additional business logic. Below is an example request containing all necessary information. To start with, we should make sure that the client has provided all necessary data. Note the information about two buildings. These are two buildings between which we want to hang an advertisement.

1.8 Use case scenario

- Actors: API client
- Initial conditions: In the database, we store information about buildings in a given city. The customer making the request has previously been registered. His data is in the database.
- Final conditions: Information about the new campaign is saved in the database.

Main scenario:

1. The customer sends all required data. Example below.
2. API checks if the customer has chosen two buildings on which the banner will be displayed. Buildings must be on the same street. Otherwise, we return the code 400.
3. Next, we calculate the cost of the ad. Advertising banners are to be hung between two buildings.
4. Each customer advertising campaign includes two banners. The banners have a rectangular shape. As a company, we want to minimize the cost of banner printing, so we try to calculate a set of two banners that cover the space between two buildings and cover the smallest possible space (we want to minimize material consumption). Our endpoint should try to find a layout of banners so that their surface is as small as possible and at the same time that both banners cover the required space.
5. Then we add the data to the database.
6. We return the newly created object representing the campaign together with data about two ads. The whole should be wrapped in return code 201.

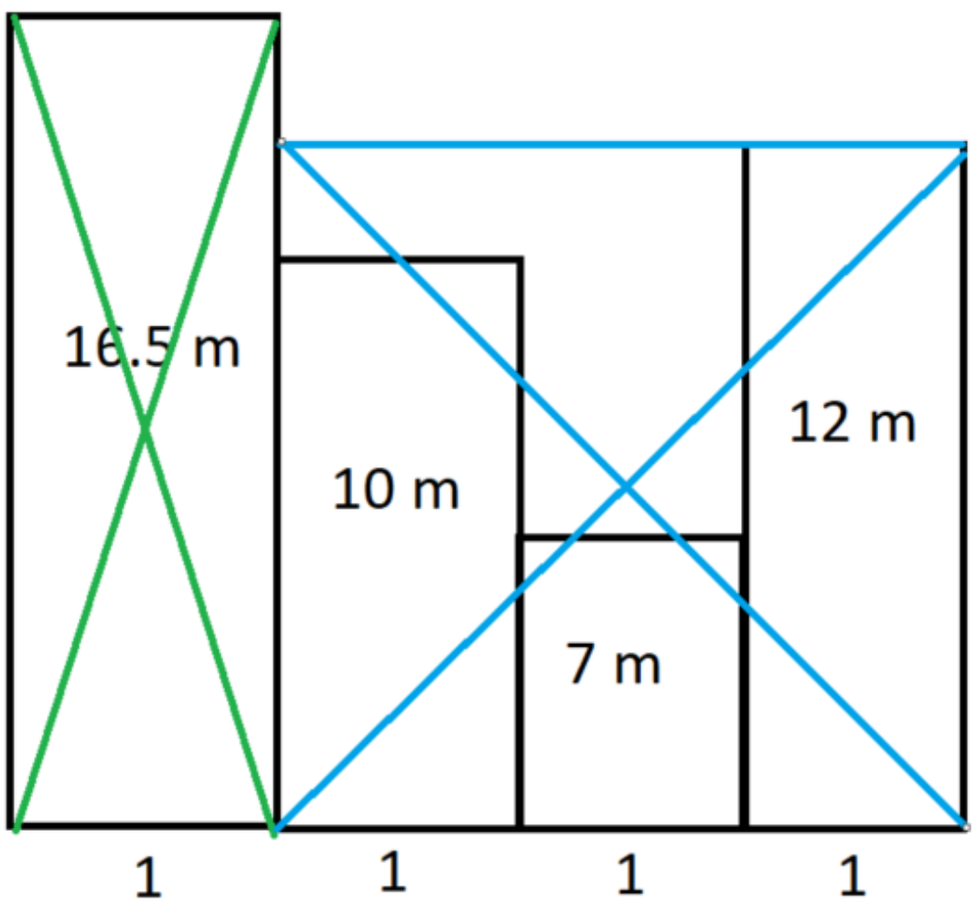
1.9 An example

1. The client sends a request in the form

```
{
"IdClient": 1,
"StartDate": "2020-1-1",
"EndDate": "2020-3-1",
```

```
"PricePerSquareMeter": 35,
"FromIdBuilding": 1,
"ToIdBuilding": 4
}
```

2. The server application then validates the request. Then the appropriate service calculates the area of two ads. Let's assume that buildings with ids 1 and 4 are on the same street. Let's assume that their height is similar to the picture below. For simplicity, we assume that the width of all buildings is the same - 1.



In the picture above, we can see 4 buildings next to each other. Each of them has a certain height. The green and blue lines define the two banners we have calculated. Remember that we always want to prepare 2 banners and they must cover entire buildings.They "stick out" outside the building. We want their area to be as small as possible. In this case:

- Baner 1 = 16.5 * 1 = 16.5 m$\hat{2}$
- Baner 2 = 12 * 3 = 36 m$\hat{2}$

We calculate the final price as follows:

- TotalPrice = (Baner1 + Baner2) * PricePerSquareMeter= (16.5 + 36) * 35 = 1837.5

2 Non-functional requirements:

- Make sure that the service responsible for the calculation of the most optimal area of the advertising space (minimized area) has the lowest possible computational and memory complexity.

3 Passing conditions

Try to implement as many requirements as possible. Even if the functionality could not be fully realized - it is worth to send such code.