13.06.2020

**Politechnika Śląska**

**Wydział Automatyki, Elektroniki i Informatyki**



**Computer Programming 4 – Project report**

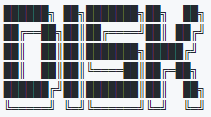
**Project title: Disk**

Authors: Bartłomiej Piura & Paweł Szafraniec

Instructor: dr. inż. Paweł Foszner

Academic year: 2019/2020

Lab group: Monday 12:00 – 13:30



**Contents:**

1. **Introduction**
2. **Project analysis**
   1. Designing
   2. Data structures
   3. Class diagram
3. **External specification**
4. **Internal specification**

4.1 Program overview

4.2 Description of types and functions

4.3 Requirements realization

4.3.1 Reflection API

4.3.2 Exception mechanisms

4.3.3 Generic types

4.3.4 Collections

4.3.5 LINQ

4.3.6 Delegates

4.3.7 Threads basics (System.Threading, Task)

4.3.8 Regular expressions

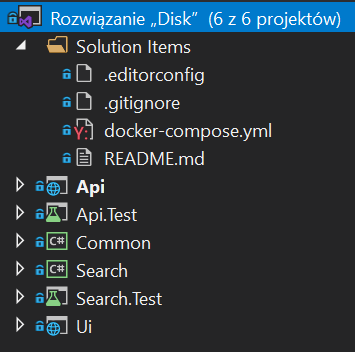
1. **Testing**
2. **Conclusions**
3. **Appendix**
4. **Introduction**

Main purpose of the project was to create application which works in similar way   
to Google Drive but in limited environment. It was supposed to store files in specified order which could there be uploaded and simply downloaded. We found the idea of such project handy and useful for us. Moreover, we came to the conclusion that the best ideas of creating subject projects are those which are useable in the future.

1. **Project analysis**
   1. **Designing**

Project is written in C# programming language using ASP .NET CORE 3.1 framework. Whole solution is divided into 6 projects:

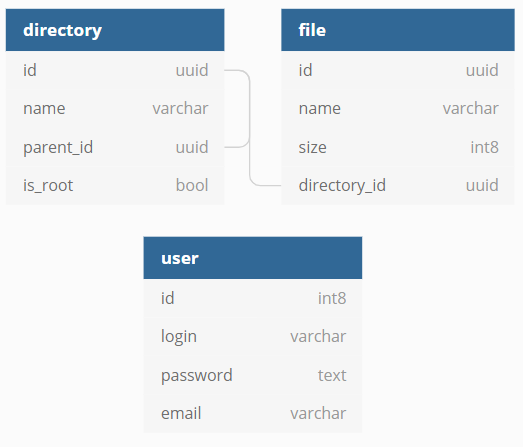
* Api – defines logic for managing directories, files etc. and communicate with database.
* Api.Test – contains basic test for Api project
* Common - contains classes shared between Api and Ui
* Search - contains classes responsible for translating string queries into LINQ expressions.
* Search.Test - contains basic tests for Search project
* Ui - is responsible for all aspects considered with user interaction as well as for graphical interface of the program.



* 1. **Data structures**

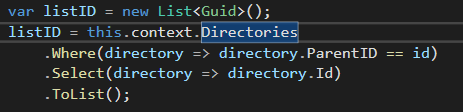
Main data storage item is database managed by PostgreSQL system. It consist of 3 tables (directory, file, user) which store values of attributes used for proper of our system.

Database structure:



Api project handles connection with database as well as querying methods extracting important values from it.

Beside the database, there are other data structures used in the program – mainly lists used e.g. for mentioned above querying methods. Example is shown below.



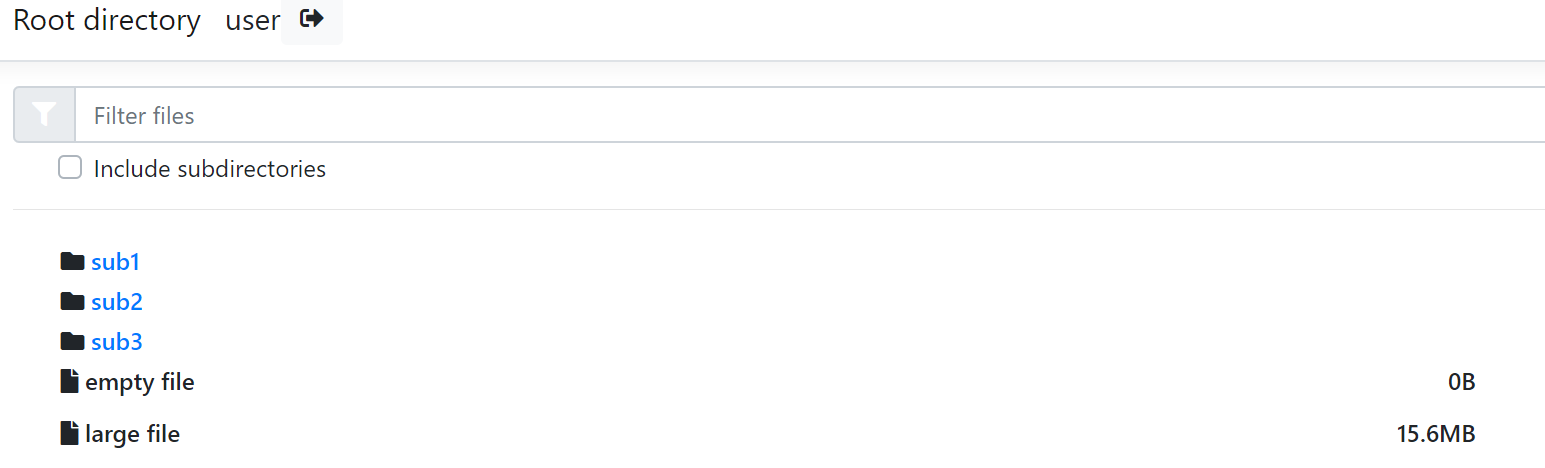
**3. External Specification**

Program is available on the url: s2.piura.pl

After entering the website, the user is asked to log in. Since the assumption of the application was exclusive availability for its authors, there is no registration option. The available logging options are controlled by the database.

However, after logging into the system, user can see list of files in root directory or in one of its sub directories. Files contain information about its size and name and are ready to be pulled.

Example view after logging:



Program can also be executed from the console level. To do that Ui and Api projects need to be run simultaneously. It can be done by running two consoles in Api and Ui project directories and typing following command:

**dotnet run**

For purpose of testing there are some test accounts:

* login: test password: test
* login: user password: user

**4. Internal Specification**

**4.1 Program overview**

In general, Api provides REST API which can be used by Ui or any external application. Ui, based on the provided REST API generates views for the user. Additionally, from the level of Api itself we can access Swagger documentation. It presents all available actions which return data, add files to the storage etc. Program is with constant communication with database, which contains data representing users and files structure.

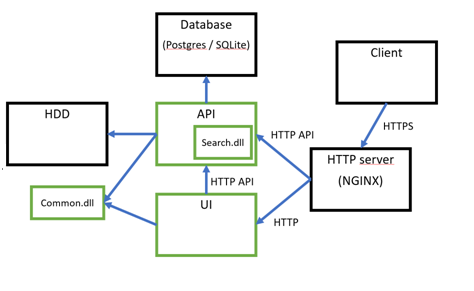


Fig. Diagram representing program communication

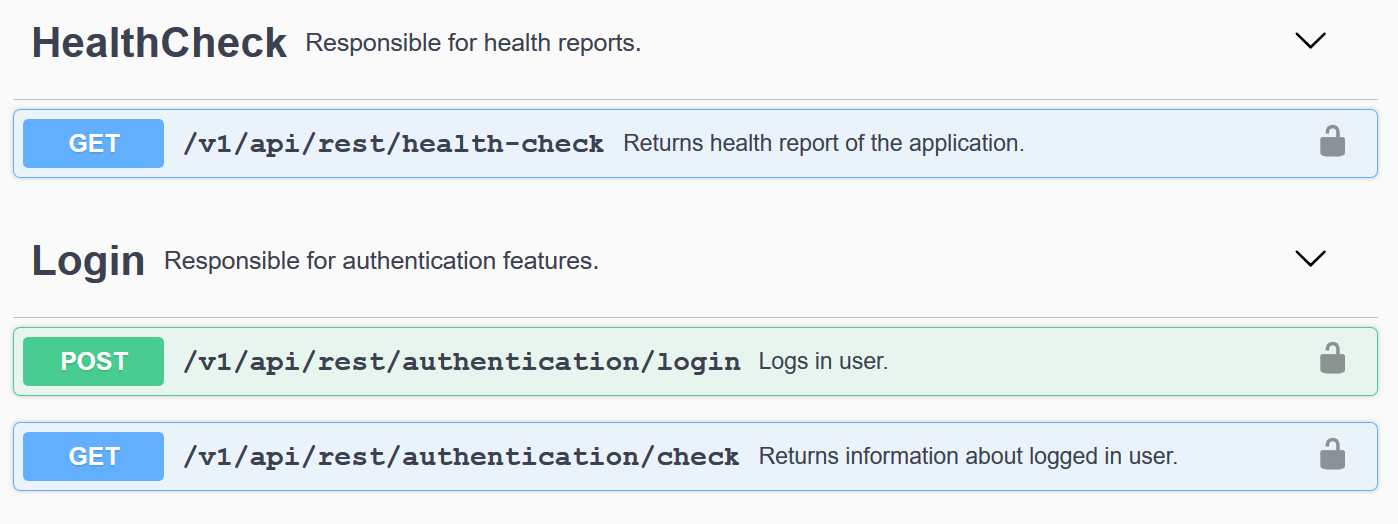
****

Fig. Swagger view

**4.2 Description of types and functions**

Description of files and types is moved to the appendix.

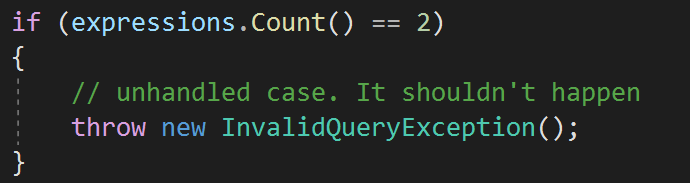
**4.3 Requirements realization**

**4.3.1 Reflection API**

Reflection is used when building expression and properties of different objects have to be used in it. By using reflection, program gets available properties on entity classes. It also detects entities by scanning given DbContext class.

**4.3.2 Exception mechanisms**

When building expressions exceptions are thrown when query is invalid. Also try block is used in general to catch different exceptions, for example exception thrown when trying to read file which doesn’t exists.



**4.3.3 Generic types**

Class responsible for mapping property names to expressions is generic class. It takes as parameter the type of the DbContext. It simplifies working with multiple DbContext classes when using DI container (not in this project).

**4.3.4 Collections**

Collections as a data structure providing flexible way to work with objects and allowing dynamic expansion was the best way to deal with needs to store data where exact number of elements is not known. It was especially useful in process of gathering data from database. Also collections allowed easy access to specific attributes of the objects and work on it. Example of collection used in the project in shown below.

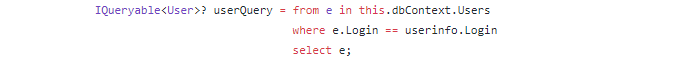
xda.png

List declared above is supposed to gather Guids from database objects satisfying given condition. As number of objects satisfying the condition may not be known e.g. because of large amount of resources in the database, usage of collection is the best way to deal with that problem.

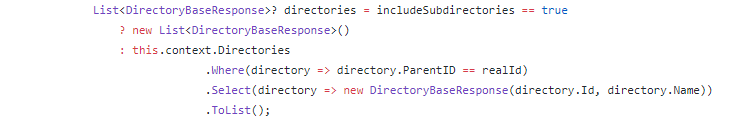
**4.3.5 LINQ**

Language Integrated Query allows querying on the objects. That technology found great application in the program. It made the communication with connected database very convenient as de facto program specification required it. Three different applications of LINQ technology are shown below.

1.



2.

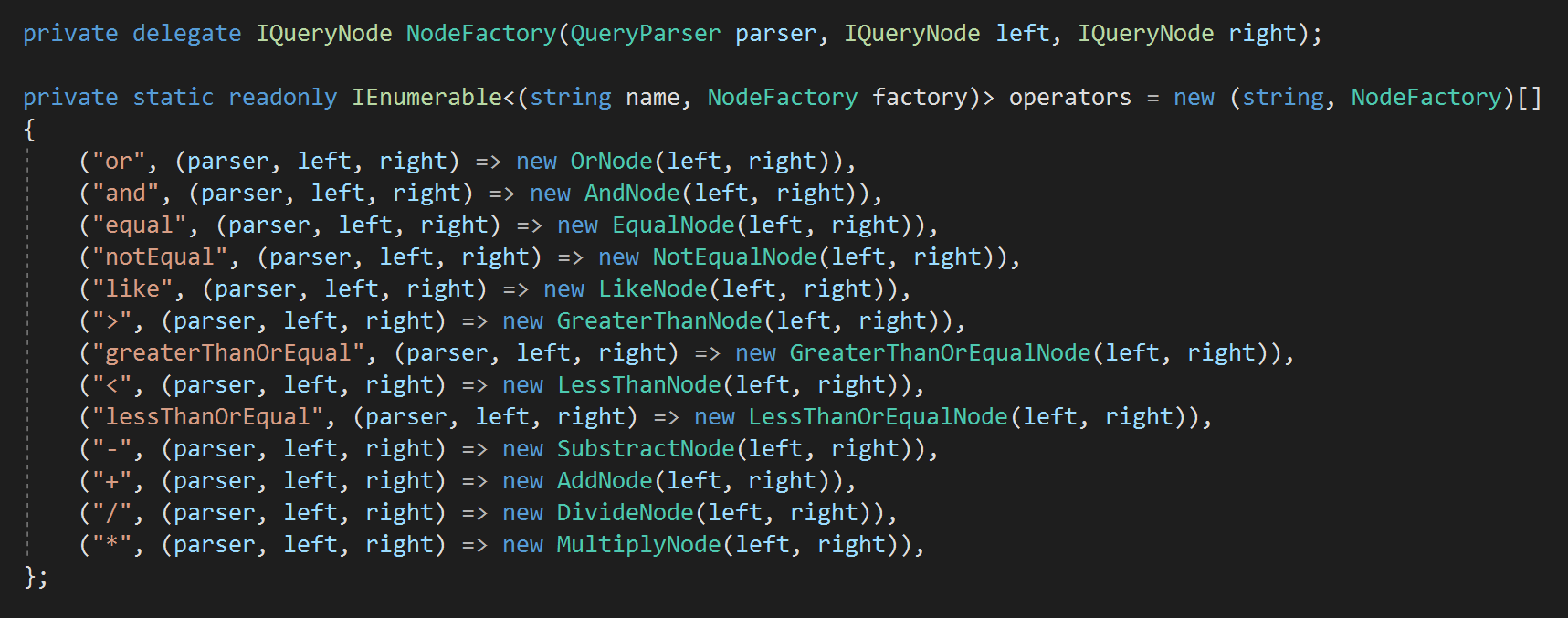


3.

Building dynamic queries using LINQ expressions in Search project.

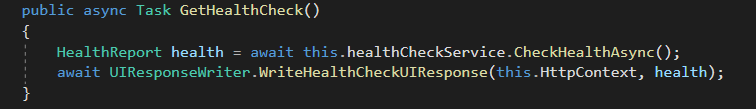
4.3.6 Delegates

Delegates are used to map operator stored in string to some action which produces query node and takes parameters.



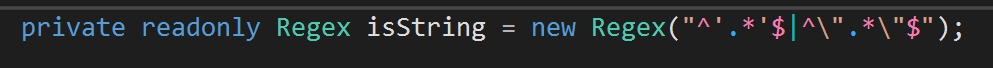
* + 1. Threads basics (System.Threading, Task)

Tasks are used with *await* and *async* keywords. It makes the application work asynchronously. Example of Task application in the program is shown below.



* + 1. Regular expressions

Regular expressions are used when parsing query, for example to replace some characters or detect type of given value.



**5. Testing**

Program contains basic tests for Api and Ui projects and it was as well tested manually. For each given input, output was exactly as expected. There are no bugs found in the usage of the program.

1. **Conclusions**

By creating this project we improved our skills in developing web applications. We gained knowledge of specific C# and .NET CORE mechanisms. Knowledge of other languages mechanisms i.e. HTML and SQL found practical application in this program as well. We found many useful resources supporting process of development, e.g. Swagger software. Finally, we came to the conclusion that the final result was worth the time devoted and the project is going to be used in the future, maybe even expanded over bigger community.