Course Project

Обектно Ориентирано Прогремиране

(OOP with C#.NET)

Ранно предизвестяване при възникнали пожари

Acceptable Programming Languages:	**C#.NET**
Deadline:	**Февруари, 2019**
Instructor:	**Dr. Evgeny Krustev**

Problem Statement:

С оглед опазване на горите от природни бедствия, а именно пожари, да се реализира система, предоставяща възможност за ранно известяване на потребителя при възникнал пожар и наблюдение. Съществуват сензори, разположени на територията на европейския съюз, с цел рапортуване на данни за техните околности. Сензорите са от различни типове – температура, влажност, прахови частици, въглероден оксид, въглероден диоксид, налягане и п.р. Всеки сензор може да притежава набор от измервателни уреди, т.е комбинация от посочените по-горе. Системата трябва да може да показва сензорите и информация за тях на карта на света, да обработва правила, представляващи логика за детекция на пожари, и да известява потребителя при възникнал пожар.

Системата трябва да се реализира като WPF приложение с табове, с приложени следните изисквания:

1. Първи таб – визуализира сензорите на карта на света, чрез open-source картите на **OpenStreetMap (**http://openstreetmap.com/). За технология да се използва **MapsUI** (https://github.com/Mapsui/Mapsui). Таб 1 трябва да има следните функционалности:
    - Визуализиране на карта
    - Визуализиране на сензори върху карта на съответните им местоположения. Информацията за сензорите ще бъде предоставена чрез ICB GraphQL API (https://graphql.org/) . Студентът трябва да напише отделна библиотека (**class library**), където да бъде имплементирана връзката с API-то. Очакван отговор за един сензор е n-орка от **SensorID, Latitude, Longitude и Type,** съответно представляващи уникален идентификатор на сензора, координати в света и тип на сензора. Сензорите могат да бъдат визуализирани чрез кръгчета или друг формат.
    - Визуализиране на последните стойности за сензор при посочване (**hover**) с мишка върху него. Да бъде създадено ново прозорче, върху картата, което да се показва, при задържане на мишката върху сензора и скрива при отдръпване. Последните стойности за сензори могат да бъдат извличани чрез ICB GraphQL API. Очаква се имплементацията да бъде в същата библиотека, като тази по-горе.
    - Съществуват сензори от тип камера. При наличие на такъв сензор да се показва последната картинка, която камерата е направила.

2. Втори таб – настройки на картата
    - Настройка за центриране на картата – текстово поле, представляващо координати, което да бъде използване от картата в таб 1, за да може да бъде центрирана
    - Настройка за приближаване на картата – текстово после, представляващо цифра от 0 до 19, което да бъде използвана от картата в таб 1, за да може да се определя нивото на приближаване (**zoom**)
    - Бутон за запазване на настройките. Настройките да бъдат сериализирани в JSON файл (**settings.json**), и презареждани при стартиране на приложението. (https://github.com/JamesNK/Newtonsoft.Json)
3. Трети таб – дефиниране на аларми
    - За всеки сензор трябва да може да бъде дефинирано **правило** на база на негов показател и критична стойност. Примерен потребителски интерфейс:
        i. Dropdown с ID-та на всички сензори
        ii. Dropdown с параметри на избрания сензор
        iii. Текстово поле за математически знак за сравнение (>, >=, <, <=, =, !=)
        iv. Поле за критична стойност

        Едно финализирано правило би изглеждало по следния начин:

        TEST_SENSOR, **TEMPERATURE > 50**

    - Обработване на правила – правилата трябва да бъдат обработвани постоянно в отделна нишка. Обработването представлява циклене по списък от правила върху последна стойност, извлечена за всяко измерване на всеки един сензор.
    - Аларми – при резултат от правило = **true**, трябва да възникне аларма. Алармата представлява известяване на потребителя, което може да бъде постигнато чрез:
        i. Windows popup
        ii. Звук
        iii. Промяна на цвят на иконките
        iv. Вдигате на приложението на преден фон
        v. Премигване на прозореца
    - Умно известяване – при извършено известяване за едно измерване на един сензор, в случай на възникване отново на първата аларма, преди тя да е приключила, потребителят да не бъде известяван отново.
    - Външно известяване – приложението трябва, също така, да известява при предизвикани аларми от ICB GraphQL API (**optional**)
4. Четвърти таб – история на известията. Всяка възникнала аларма да бъде записвана в списък и визуализирана в този прозорец. При добавяне на нова възникнала аларма, всички аларми да бъдат сериализирани в **JSON** файл (**alarms.json**). При стартиране на приложението да бъдат зареждани и визуализирани последните 50.

Бонус точки при:

- Реализиране на известяване на външните аларми
- Използване на **async/await** механизма
- Използване на **MVVM** моделът

На студентите ще бъде предоставена документация за ICB GraphQL API и ще бъде проведено съответното обучение за използването му.

Evaluation:

Your project will be evaluated on the following **general points**:

- **Sophistication/complexity/originality** of the problem being solved/investigated and of the **solution(s)/approaches** considered.
- **Demonstrated ability to extract/analyze** concurrency-related problems/issues from a general problem/area of interest.
- **Clarity of explanations, and for implementations programming skill/quality**. Your report **(in Bulgarian!)** should be well written and free of grammatical and spelling errors.
  Programs must be **well-commented and in a professional style**.
- **Awareness of related work.** Others have considered the same or similar problems before you. Your work does not have to be novel, but you should be able to contextualize your approach. Be sure to explain how each referenced work is *related* to your work. Note that a 5-minute *Google* search will not be adequate; if you are unfamiliar with the required textbooks for the course:
- **Completeness** of the project.

Deliverables:

1. the source code
2. the executable code
3. the instructions for compiling your source code
4. the report explaining the data structures and the algorithm implementation, describe things such as how your code has been tested, limitations of your code, problems encountered, and problems remaining
5. any files used to test the implementation of the program with an explanation about it included in the report.