****

Goldfinger

Team Project Assignment

# **Java Team Project**

This document describes the **project assignment** from **Haemimont AD** for the **Java cohort** at Telerik Academy.

## **Project Description**

Your task is to develop a web application that enables users to inspect hazards and risks at any point of the world. Data is provided in the form of geospatial files (ESRI shapefiles) and your app should be able to visualize this data on a map and display the metadata associated with each shape. An example hazard data that can be analyzed this way is the soil types for different regions of the world (in North Bulgaria one of the dominant soil types for example is *Calcic Chernozems*).

In addition to that, an auditability framework should be created for logging different types of events from your application which can be later exported upon admin request. These events represent all sorts of different activities that users can perform such as login, logout, upload a file, request soil type info for a given location, etc. It’s important to track what your users are doing within the system.

# **Project Requirements**

## **Web application**

### **Public Part**

The public part of your projects should be **visible without authentication**. This includes the application start page, the user login and user registration forms. People that are not authenticated cannot interact with the rest of the website. Admin users are seeded upon deployment, only regular users can sign-up through the web form. Every admin user has also the permissions of regular users.

Audit events for login and logout should be recorded.

### **Private Part (Users only)**

The web application provides users with UI to inspect different types of hazards and risks available on the platform. Users should be presented with an option to select the type of data to inspect (“Soil Type”, “Earthquake Frequency”, etc.) and a map on which they can see information for a region of their choice.

The actions available to users, related to hazard/risk retrieval are:

* Double click on the map

Retrieves a polygon area of the hazard/risk category to which the clicked location belongs and visualizes it on the map. E.g. if a user double-clicks on Sofia, when the selected hazard/risk type is “Soil type”, then your app should retrieve the soil region to which Sofia belongs and visualize it on the map.

Only support for polygon shape files is required.

As part of the mandatory project requirements only two data types should be supported – “Soil Type” and “Earthquake Frequency”. Make sure that shapes for different types of soil (DOMSOIL field) in case of “Soil Type” and different frequencies (DN field) in case of “Earthquake Frequency” are filled with a different color (same values are the same color of course).

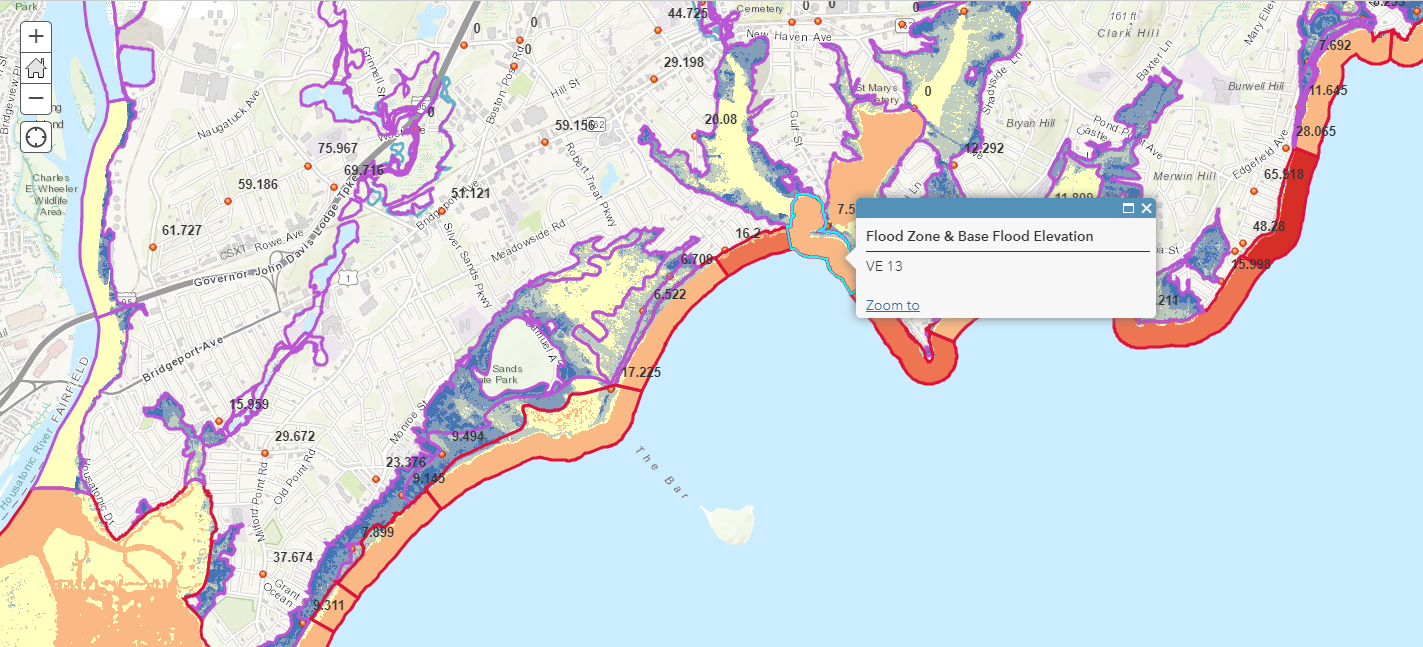
* Click or hover over an area on the map for which data is retrieved with a double click

Once we have one or more areas visualized on the map, users would like to see more details related to this area. E.g. “size in square km” or “soil type”, etc. That may happen either by clicking on the area with the mouse or if you just hover over this area.

* Clear the map with a button click or context menu click

Removes anything visualized on the map with the actions above.

* Clear the map if a different data type is selected



Audit events for each retrieval of a shape with a double click on the map should be recorded. The latitude and longitude of the clicked location and the requested data type should be written as part of the auditability record.

### **Administration Part**

**System administrators** should have administrative access to the system and permissions to administer and review information which is not accessible to regular users.

As part of the mandatory features, administrators will be only capable of viewing an auditability report for what actions have been performed by users (see above for details about what needs to be tracked).

To display the report, administrators need to provide a time range for which a report is requested and optionally a query string for filtering the records. Administrators should not be allowed to export data for a period longer than 3 months.

The query string supports the following format:

* *<word1> <word2> <word3>*

Searches for any of the words *<word1>, <word2>* or *<word3>* in username, IP and the custom data associated with the records (full-text search, similar to what Google does).

* “*<word1> <word2> <word3>*”

Searches for the whole phrase “*<word1> <word2> <word3>”* in username, IP and the custom data associated with the records (full-text search).

* *<key>: <value>* or *<key>: “<value>”*

Searches for *<value>* using full-text search (see the rules above) but only for the field *<key>*. E.g. “*username: krasi”* will search for krasi in username using full-text search

* *<key> = <value>*

Searches for an exact match of *<value>* for the field *<key>*.

E.g. “*dataType = soil*” will search for dataType equal to “*soil*” (this will match any records with dataType “*soil*” but will not match records with dataType “*soil type*”, which “*dataType: soil*” will match)

The report should be in the form of a table which includes the following information:

* Username
* IP
* Time of the event
* Additional details part of the message (depends on the action being tracked)

The table should support paging and sorting by username, IP and time of the event (default sorting is by relevance with the query). And administrators should also be presented with the option to download the report in the form of a CSV file.

## **REST API**

The core logic of the application (auditability framework + map server (shape data retrieval)) should be implemented as REST micro-services using Spring Boot or a similar framework (Dropwizard, Vert.x, etc.)). Modern UI frameworks such as React can call REST APIs directly, but you can use Spring MVC for serving the web content and from your controllers, you can then make calls to the two web services – auditability and map-server.

## **Database**

There is no restriction of what database you are going to use, but requirements can be achieved with **MySQL database** even if this is not always the best choice**.**

# **Technical Requirements**

Your project should meet the following requirements (these requirements will be used by TA trainers during project defense):

* Use **IntelliJ IDEA**
* For UI, you can use **Bootstrap**, **Materialize** or anything else that can satisfy the needs of the app (feel free to change the standard theme and modify it to apply own web design and visual styles, if you like)
* You can use a mapping service of your choice, but Bing Maps should be well suited for your needs.
* Apply **error handling** and **data validation** to avoid crashes when invalid data is entered (both client-side and server-side)
* Application should be seeded with a few admin accounts upon deployment. Regular users will sign-up by themselves.
* Integrate only the provided shape files for “Soil Type” and “Earthquake Frequency” (Haemimont will provide them to you). For “Soil Type” visualize only the fields *FAOSOIL, DOMSOI, COUNTRY, SQKM* from the metadata.
* Not required to secure the access to the auditability and map micro-services but make sure they are not public available if the app is deployed on a hosting provider.
* Create **unit tests** for your "business" functionality following the best practices for writing unit tests (**at least 80% code coverage**)
* Use **Inversion of Control** principle and **Dependency Injection** technique.
* Follow **OOP** best practices and **SOLID** principles.
* Use **Git** for source control management
* Use **Trello** for project management
* Create **user documentation / manual** of the application

# **Optional Requirements (bonus points)**

* Add a functionality for the administrators to upload new shape files, providing the name for the data type they represent and which columns from their metadata to be displayed on the map.
* Upon adding a new shape file, add an option to define how to color the different shapes based on metadata (E.g. area < 1000 sq. km is light green, area between 1000 and 10000 sq. km is darker green, etc.). Support only field exact matches and ranges. Also add a default color. If nothing is provided, randomize the colors.
* Add support in the web app to show all the shapes for the data type. This means that double clicking is redundant in that mode and only click/hover functionality is required to display the information for the shape. If this action is requested, an audit record should be created specifying the action invoked and the data type for which it applies.
* Host your application’s backend in a public hosting provider of your choice (e.g. AWS, Azure)
* Integrate your app with a **Continuous Integration server**  (Jenkins (<https://travis-ci.org/>), Circle CI (<https://circleci.com/>) or any other). Configure your unit tests to run on each commit to each branch and deploy your app (if hosted online) on push to **master**.
* Add an option for admins to create new admins or regular users as part of the user management (lower priority)
* Add options for users to view and edit their profile (lower priority)

**IMPORTANT: Optional requirements are more than what can be likely accomplished as part of this project. If you have time for any of them you can pick a subset of these requirements of your choice and implement them in the project.**

# **Deliverables**

Provide link to a **Git** repository with the following information:

* Link to the Trello board
* Project documentation
* Screenshots of your application
* URL of the application (if hosted online)

# **Public Project Defense**

Each student must make a **public defense** of its work to the trainers, Partner and students (~30-40 minutes). It includes:

* Live **demonstration** of the developed web application (please prepare sample data).
* Explain application structure and its **source code**

# **References**

This a list of useful links that you might find useful during the development of your project.

* [Top 10 Mapping APIs](https://www.programmableweb.com/news/top-10-mapping-apis-google-maps-microsoft-bing-maps-and-mapquest/analysis/2015/02/23)
* [Bing Examples](https://docs.microsoft.com/en-us/bingmaps/v8-web-control/map-control-concepts/map-shapes-polylines-and-polygons/polygons-with-holes-example)
* [GeoTools](http://docs.geotools.org/)
* [Hibernate Spatial](http://www.hibernatespatial.org/documentation/02-Tutorial/01-tutorial4/)
* [Free GIS Data (Shape Files or other format)](https://freegisdata.rtwilson.com/)
* [Quantum GIS (Open Source GIS Software)](https://qgis.org/en/site/)