Quiz App

Quiz App is a web application for geographical quizzes based on openstreetmap data.

Application enables user to pick a quiz about administrative division of a specific area by selecting an area on the map where the quiz will be based and then selecting administrative units to be guessed in the quiz.

In the quiz user is prompted to click areas on the map given their name. User answers by clicking appropriate areas on the map until all of the areas are guessed.

It is a client-server application where the server is responsible for fetching the data via an overpass API which can be then displayed by the client on the map.

Serwer

Strona serwerowa przyjmuje zapytania od klienta i odsyła informacje na temat różnych administracyjnych obiektów geograficznych. Użytkownik może pytać o poziomy administracyjne różnych krajów, pobierać geometrię tych obszarów w formacie geoJson oraz pobierać już wcześniej zapisane quizy. Serwer następnie tworzy zapytanie do serwera overpass wraz z odpowiednimi parametrami i pobiera pożądane informacje ze zdalnej bazy danych. Potem następuje końcowa obróbka danych, które potem są wysyłane do klienta. Niestety, głównie z powodów zamieszania administracyjnego i braku danych, tylko niektóre kraje są obsługiwane. Serwer działa na frameworku Node.js i implementuje endpointy za pomocą biblioteki express. Serwer korzysta z bazy danych mySql (mariaDB) aby przechowywać informacje o obsługiwanych krajach. Do konwersji informacji z osmJson do geoJson, została użyta biblioteka osmtogeojson zawierająca funkcje konwertujące.

Krótki opis poszczególnych komponentów:

endpointy:

- /api/countries - zwraca listę obsługiwanych krajów.
- /api/administratives/:level_number([0-9]+)/:admin_name - zwraca metadane dotyczące obszarów administracyjnych na poziomie administracyjnym równym :level_number w obrębie obszaru zadanego nazwą :admin_name.
- /api/geometry/:level_number([0-9]+)/:factor([0-9]+)/:admin_name - zwraca geometrię wszystkich obszarów na poziomie :level_number w obrębie :admin_name zoptymalizowane o czynnik :factor.
- /api/levels/:countryID([0-9]+) - zwraca informacje o obsługiwanych poziomach administracyjnych w kraju o indentyfikatorze :countryID.
- /api/get_cnt_geom/:factor([0-9]+) - zwraca geometrię wszystkich obsługiwanych krajów przeskalowaną o czynnik :factor.
- /api/quizzes/:factor([0-9]+)/:id([0-9]+) - zwraca zapisany quiz o identyfikatorze :id przeskalowany o czynnik :factor.
- / - zwraca aplikację kliencką.

schemat bazy danych:

- tabela countries - przechowuje informacje o obslugiwanych krajach.
- tabela administrative_levels - przechowuje informacje o obslugiwanych poziomach administracyjnych. Złączona z tabelą countries relacją "kraj ma poziomy".

klasy:

- overpassAPI - obsługuje zapytania oraz wysokopoziomowe działania. Jest fasadą dla całego systemu zapytań do overpass oraz obróbki danych.
- geoJsonParser - zajmuje się obróbką danych już w formacie geoJson. Jest obiektem.

pliki:

- create_database_schema.sql - tworzy schemat bazy danych.
- populate_database.sql - wstawia do bazy danych informacje o krajach, które są obsługiwane.
- geoJsonModule.js - zawiera klasę geoJsonParser.
- index.js - server Node.js.
- overpassModule - zawiera klasę overpassAPI.
- queries.js - zbiór zapytań w postaci tablicy.
- quizzes_meta.js - metadane dotyczące kwerend w queries.js.

Client side

Main class of the app is **MapLoader** class responsible for loading the map with all the controls. It posseses a reference to an **EventHandler** object which is either of type **QuizChooser** or **QuizHandler**. It determines in which mode the map is at the moment and provides event handlers for click events for areas on the map.

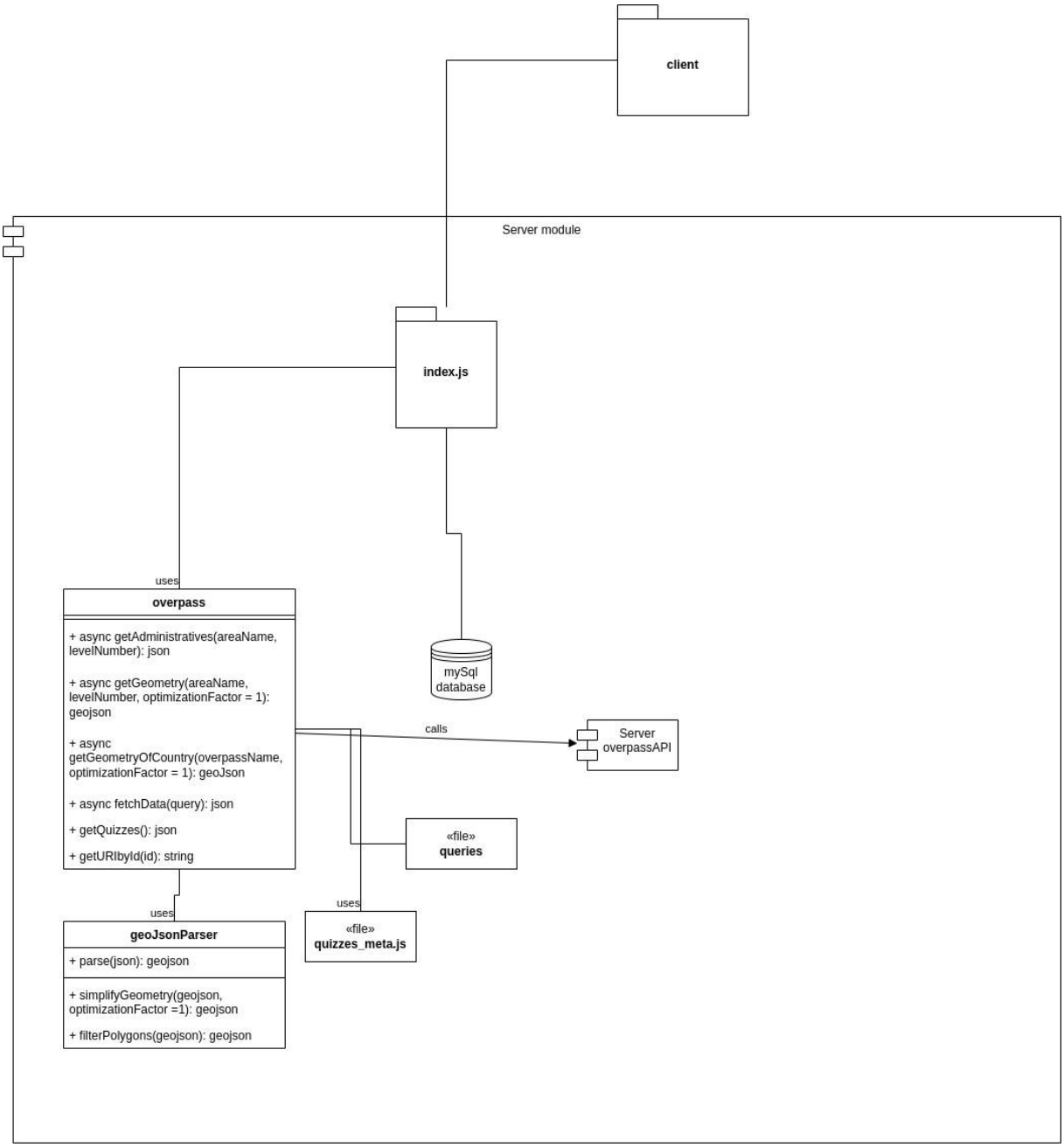
In quiz chooser mode user chooses the quiz by clicking the areas on the map. Upon clicking an area its administrative division data is fetched. Currently loaded area is stored in an object of type **MapArea** which is of type **World** or **AdminUnit**. These classes form a *state* design pattern. They provide methods to get the data essential to display the map and create control in the top right corner.

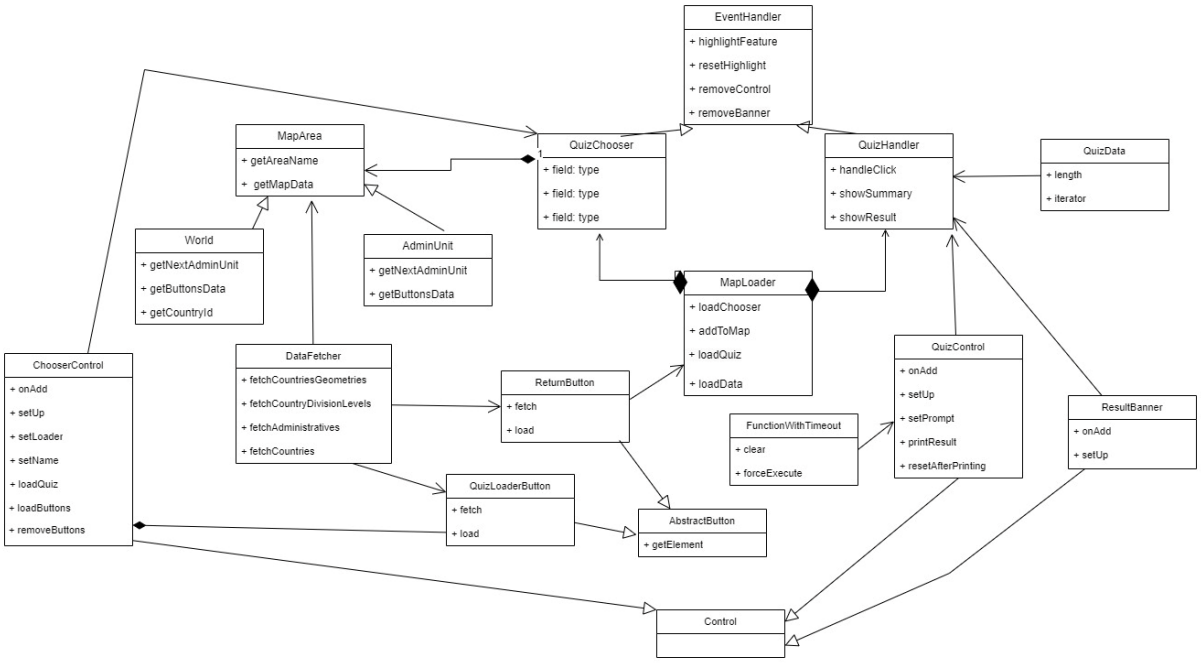
The aforementioned control of type **ChooserControl** contains buttons of type **QuizLoaderButton** used to load the quiz.

In quiz handler mode user is prompted to click areas on the map. All the areas names are stored in **QuizData** data structure and **QuizHandler** uses an *iterator* to get them. Upon clicking an area result is shown in the control **QuizControl** and on the map then **FunctionWithTimeout** object is created containing a function that resets the state of the map. This forms a *command* design pattern and is executed when the timeout of two seconds elapses or another area is clicked.

When the **QuizData** data structure empties the quiz ends and **ResultsBanner** is displayed. During the entire time **ReturnButton** occupies the bottom right corner enabling users to go back to the starting view any moment. Both buttons inherit from **AbstractButton** which constructor is a *template method*.

Styles for areas on the map are provided by **PolygonStyles** object and all the data are fetched by **DataFetcher** object which is a *singleton* and provides a *facade* to the server side.


client
Server module
index.js
uses
overpass
+ async getAdministratives(areaName, levelNumber): json
+ async getGeometry(areaName, levelNumber, optimizationFactor = 1): geojson
+ async getGeometryOfCountry(overpassName, optimizationFactor = 1): geoJson
+ async fetchData(query): json
+ getQuizzes(): json
+ getURIbyId(id): string
mySql database
calls
Server overpassAPI
«file» queries
uses
uses
geoJsonParser
+ parse(json): geojson
+ simplifyGeometry(geojson, optimizationFactor =1): geojson
+ filterPolygons(geojson): geojson
«file» quizzes_meta.js


EventHandler
+ highlightFeature
+ resetHighlight
+ removeControl
+ removeBanner
MapArea
+ getAreaName
+ getMapData
QuizChooser
+ field: type
+ field: type
+ field: type
QuizHandler
+ handleClick
+ showSummary
+ showResult
QuizData
+ length
+ iterator
World
+ getNextAdminUnit
+ getButtonsData
+ getCountryId
AdminUnit
+ getNextAdminUnit
+ getButtonsData
MapLoader
+ loadChooser
+ addToMap
+ loadQuiz
+ loadData
ChooserControl
+ onAdd
+ setUp
+ setLoader
+ setName
+ loadQuiz
+ loadButtons
+ removeButtons
DataFetcher
+ fetchCountriesGeometries
+ fetchCountryDivisionLevels
+ fetchAdministratives
+ fetchCountries
ReturnButton
+ fetch
+ load
QuizControl
+ onAdd
+ setUp
+ setPrompt
+ printResult
+ resetAfterPrinting
ResultBanner
+ onAdd
+ setUp
FunctionWithTimeout
+ clear
+ forceExecute
QuizLoaderButton
+ fetch
+ load
AbstractButton
+ getElement
Control