Edytor map 3D

Projekt na zajęcia e-systemów Java

Sebastian Wilgosz (195963)
Radosław Skiba()

Termin zajęć: Czwartek, godz. 17:05
Prowadzący: **dr inż. Tomasz Walkowak**

19 kwietnia 2015

Spis treści

Rozdział 1

Cel i zakres projektu

1.1 Zakres projektu

Niniejszy projekt będzie zawierał opis techniczny i specyfikację kroków, wykorzystania narzędzi niezbędnych do stworzenia aplikacji w środowisku **Java IEE** . Projekt będzie uwzględniał następujące etapy :

1. Analiza i Specyfikacja
2. Projektowanie
3. Implementacja
4. Testy
5. Ocena i Optymalizacja

1.2 Cel projektu

Celem projektu jest wykonanie aplikacji serwerowej. Aplikacja będzie budowana w oparciu o technologię **Java IEE**. Sam sposób wdrożenia programu jak i jego zadania wraz z implementacją będą przebiegały wedle przedstawionego pomysłu, który został wstępnie zaakceptowany przez prowadzącego kurs. Projekt ma na celu zapoznać Studentów specjalizacji Inżynieria Internetowa z procesem budowania oprogramowania obecnie stosowanego w świecie aplikacji serwerowych. Projekt dodatkowo dzięki wybranemu tematowi pomoże zgłębić technologie strony klienckiej takie jak **WebGL** i **JavaScript**

Rozdział 2

Analiza i Specyfikacja

2.1 Opis słowny zadania

Celem naszej pracy projektowej jest stworzenie aplikacji zdolnej do tworzenia i zarządzania mapkami 3d terenu. Niniejsza aplikacja serwerowa ma za zadanie wspomagać tworzenia ciekawych wizualizacji terenu dla osób zajmujących się hobbystycznie jak i zawodowo kartografią. Tworzone przez użytkowników prace można będzie łączyć, zlecać wykonanie na nich danych obliczeń w celu wykorzystania ich w danym problemie. Nasz serwis z aplikacją nazwaliśmy
Ąvalanche" *[eng. lawina]*, w dalszej części projektowania i założeń podamy bliższe szczegóły naszych kroków projektowych. Aplikację w przyszłości może uda rozwinąć się do dużo ciekawszych zastosowań.

2.2 Specyfikacja wymagań funkcjonalnych

1. Prezentacja bazy punktów w formie mapy :
– Jest to główna funkcja tej aplikacji, ma ona za zadnie z zadanego zbioru danych generować podgląd danego terenu w formie obrotowej mapki w przestrzeni trójwymiarowej.

2. Oddzielne sesje dla każdego użytkownika :

– Ponieważ aplikacja będzie zawierała w sobie proces tworzenia jakiegoś elementu terenu niezbędne będzie wyodrębnienie pojedynczych działań na aplikacji w formie sesji użytkowników, którzy swoje gotowe mapki będą mogli dzięki powiązaniu z kontem przechowywać w zdalnej przestrzeni dyskowej

3. Wczytywanie mapek z pliku :

– Poza wczytywaniem mapek z bazy serwera możliwe będzie wczytywanie wcześniej zapisanych mapek z pliku. Funkcjonalność taka będzie przydatna gdy użytkownik po wykasowaniu maki bądź usunięciu konta, chciałby odtworzyć swoje prace

4. Łączenie kilku prac w jedną :

– Serwer będzie mógł według kilku definiowanych zasad łączyć kilka zasobów w jedną wspólną mapę, taka funkcjonalność będzie przydatna przy tworzeniu pracy opartej na wkładzie kilku użytkowników

5. Generacja map :

– Generacja mapek na podstawie już istniejącej z podanymi zasadami zmiany oraz tworzenie totalnie losowej mapy

6. Zapisywanie prac na serwerze :

– Każda stworzona mapka będzie dostępna niezależnie od miejsca i sprzętu użytkownika poprzez interfejs webowy dostępny w przeglądarce internetowej

7. Eksport gotowych prac do określonych formatów :

– Każdą z gotowych prac będzie można wyeksportować do formatu możliwego do użytku w celu prezentacji/wizualizacji z założenia są to formaty ***.pdf *.jpeg**.

2.3 Specyfikacja wymagań niefunkcjonalnych

1. Przejrzysty interfejs webowy :

– Wymaganiem jest by aplikacja była nie przeładowana dodatkami i prosta w obsłudze dla osób nie posiadających zdolności programistycznych

2. Szybkość i oszczędność łącza :

– Aplikacja powinna większość pracy wykonywać bez potrzeby generowania zbędnego ruchu sieciowego co pomoże zaoszczędzić zasoby klienta

3. Multiplatformowość :

– Aplikacja powinna generować taki sam rezultat niezależnie od platformy sprzętowej klienta

4. Baza danych MySQL :

– Zasób danych powinien być przechowywany na łatwej w obsłudze i bezpłatnej dystrybucji bazy danych, takiej jak np. **MySQL**

5. Dane o twórcach jak i projekcie :

– Aplikacja będzie zawierała informacje o okolicznościach w jakich powstała i dla jakich celów

6. Model MVC :

– Aplikacja będzie opierać się na modelu MVC

7. Dokumentacja :

– Przebieg procesu powstawania jak i opis funkcjonalności i sposób wykorzystania poszczególnych funkcji będzie zawarty w dokumentacji projektu

Rozdział 3

Wykorzystywane moduły

3.1 Devise

Devise to łatwy w użyciu i dostosowaniu do aplikacji moduł rozszerzający Ruby on Rails o proste zarządzanie sesjami. Gwarantuje bezpieczne przechowywanie haseł, kontrolę dostępu, kompletne funkcje logowania, rejestracji, powiadomień mailowych.

- Napisany jest w rubym
- Bazuje na silniku RoR, jest całkowicie oparte o strukturę MVC
- Pozwala na zalogowanie wielu modeli jednocześnie
- Bazuje na strukturze modułowej - można dołączać tylko te wtyczki, które rzeczywiście potrzebujemy

Devise składa się z 10 modułów:

- Database Authenticatable: koduje i zapisuje zakodowane hasła w bazie danych w celu zapewnienia autoryzacji podczas logowania.
- Omniauthable: dodaje kompatybilność z mediami społecznościowymi (logowanie przez Facebook, Twitter, Google itp).
- Confirmable: wysyła email z potwierdzeniem rejestracji i weryfikuje czy konto jest potwierdzone czy nie.
- Recoverable: umożliwia resetowanie haseł i wysyła email z instrukcją zresetowania hasła.
- Registerable: zajmuje się tworzeniem kont użytkowników, także edycji ich profilów i usuwaniem swoich kont. Nie pozwala na edycję innych kont ani na usunięcie nie swojego konta, umożliwia automatyczne zalogowanie po rejestracji Rememberable: umożliwia zapamiętanie sesji użytkownika w cookies, dzięki czemu pozostaje on zalogowany nawet po zamknięciu przeglądarki. Trackable: śledzi ilość logowań, adresy IP, rodzaj przeglądarki z której się logowano. Timeoutable: wylogowuje użytkownika po określonym czasie, jeśli nie stwierdzono żadnej aktywności. Validatable: dostarcza walidację adresów email oraz haseł. Sprawdza długość hasła, poprawność formatu wprowadzonego adresu email. Jest to opcjonalne i nie przeszkadza w definiowaniu własnych walidacji. Lockable: blokuje konto po określonej ilości błędnych logowań. Można odblokować konto przez token wysłany na adres email lub po określonym czasie.

Devise gwarantuje przechwytywanie przechwytywanie wyjątków, zapewniając bezpieczeństwo oraz generowanie czytelnych komunikatów we wszystkich możliwych przypadkach użycia.

W naszej aplikacji będziemy wykorzystywać następujące moduły devise:

- database authenticatable
- registerable
- recoverable
- rememberable
- trackable
- :validatable

3.2 ActAsTaggableOn

ActAsTaggable - to moduł wykorzystywany przez nas do grupowania map oraz użytkowników. Wtyczka ta umożliwia szybkie dodanie funkcji otagowania dowolnego modelu, filtrowanie po tagach, dodawanie wielu tagów jednocześnie czy wyświetlania powiązanej chmury tagów.

Dodatkowe funkcje zapeniane przez ten moduł, to znajdowanie najczęściej i narzadziej używanych tagów, powiązanie wielu rodzajów tagów z jednym modelem (np: tagi jako kategorie, grupy, szkoły - wszystko może być powiązane z modelem użytkownika)

3.3 CanCanCan

CanCanCan to biblioteka do autoryzacji dla Ruby on Rails. Pozwala określić, do jakich zasobów aktualnie zalogowany użytkownik ma dostęp. Wszystkie prawa dostępu są zdefiniowane w pojedynczej lokalizacji (klasie Ability) i nie są powtarzane w kontrolerach, widokach czy zapytaniach do bazy danych.

3.4 PG

Wykorzystywana przez nas baza danych to postgreSQL - wybraliśmy ją, ponieważ serwer Heroku, na którym hostujemy naszą aplikację wykorzystuje tę bazę jako domyślną, co uprościło konfigurację.

Ruby on Rails to framework zaprojektowany do współpracy z bazą danych. Klasa ActiveRecord zawiera zdefiniowany komplet funkcji komunikującej się z bazą, tworzeniem, edycją, czytaniem i usuwaniem rekordów. Jedyne, czego potrzebujemy, by sprawić, by komunikacja z bazą zaczęła działać, to dodanie modułu precyzującego, jaką bazę nasza aplikacja będzie wykorzystywać.

PG - to interfejs dla Ruby umożliwiający współpracę z bazą danych PostgreSql.

3.5 Uproszczona składnia

Aby przyśpieszyć pracę nad aplikacją i ułatwić zarządzanie stylami/skryptami, dzieląc je na wiele plików postanowiliśmy nie wykorzystywać bogatych w niepotrzebne znaki standardowych składni aplikacji webowych. Zainteresowaliśmy się nakładkami na wykorzystywane przez nas języki (HTML, CSS, JavaScript), które po wrzuceniu na serwer produkcyjny są komplikowane i łączone w pojedyncze pliki, by nie generować dziesiątek requestów do serwera.

- Slim-Rails - jest to interpreter języka Slim, nakładki na HTML pozwalającej na pominięcie wszystkich tagów zamykających, bazujący na wcięciach (podobnie do składni Python'a).
- Coffe-Rails - interpreter języka CoffeeScript będącego udoskonaleniem składni JavaScript. Opiera się na tych samych założeniach co Slim czy Sass.
- Sass-Rails - interpreter języka Sass, będącego nakładką na standardową składnię CSS, usuwający wszelkie nawiasy klamrowe, średniki, umożliwiający definiowanie zagnieżdżonych styli, wprowadzenia zmiennych, czy działań matematycznych.
- Uglifier - kompressor do assets. Umożliwia minimalizację kompresję plików *.coffee oraz *.sass do zminimalizowanych odpowiedników *.js oraz *.css. Dzięki temu usunięte sa wszystkie białe znaki, a nazwy funkcji zamieniane są na jednoliterowe.
- Less-Rails - łączy wszystkie pliki assets w jeden, dzieki czemu do serwera idzie jeden request zamiast osobnego requesta na każdy plik ze stylami czy skryptem.

3.6 TwitterBootstrapRails

Ta wtyczka umożliwia integrację naszej aplikacji z jednym z najbardziej znanych frameworków frontendowym wyekstraktowanym z Twittera - Bootstrapem.

3.7 SimpleForm

Dodatek umożliwjający łatwe generowanie skomplikowanych formularzy HTML dla dowolnego obiektu.

3.8 TurboLinks

By uniknąć pobierania z serwera skryptób javaScript oraz styli CSS za każdym razem, gdy użytkownik kliknie w link na stronie, używamy turbo linków. Pozwalają one na odświeżenie wyłącznie kodu zawartego między tagami <body></body>, zaś cała sekcja <head></head> jest ładowana tylko raz.

Dzięki temu zabiegowi nasza aplikacja ładuje się wielokrotnie szybciej.

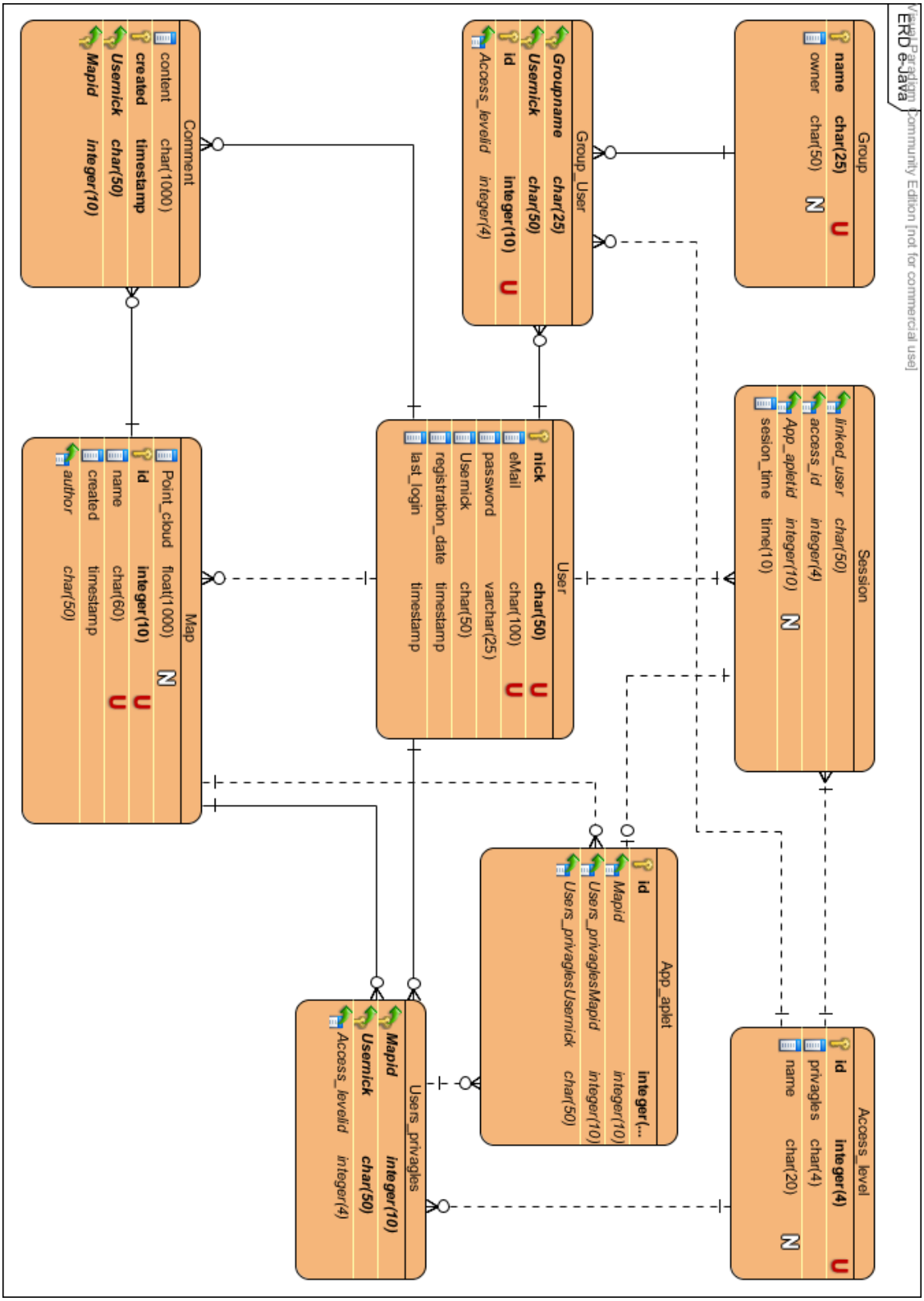
3.9 gravastic

Moduł pozwalający na wyświetlania Gravatarów użytkowników oraz komentujących mapy, bazujac na emailu podanym w formularzu.

Rozdział 4

Modele systemów

4.1 Diagram ERD



Rysunek 4.1: Diagram ERD

[b]