AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA
W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI
I TELEINFORMATYKI

KATEDRA INFORMATYKI

Dokumentacja Specyfikacyjna
Kryptonim: **OtoKonfa**

Aplikacja umożliwiająca wyszukiwanie i porównywanie ośrodków wynajmujących sale na spotkania o charakterze konferencyjnym

Autor: *S. Chmiela, M. Górka, M. Imiełowski, P. Stawicki, P. Wolak*
Kierunek studiów: *Informatyka*
Opiekun pracy: *mgr inż. Witold Rakoczy*

Kraków, 2016

Spis treści

1. Stos technologiczny

1.1. Wybrana technologia

Cały projekt został wykonany przy użyciu Ruby on Rails. Zdecydowaliśmy się na tę technologię, gdyż pozwala ona na szybki rozwój aplikacji, a co za tym idzie łatwe pisanie prototypów. Dodatkowo sama technologia oferuje dużą liczbę gemów (zależności Railsów), dzięki którym można znacznie uprościć pisanie kodu, gdyż znaczna część funkcjonalności jest już zaimplementowana w którymś z nich.

1.2. Zależności

W projekcie wykorzystujemy następujące zależności zewnętrzne:

1. **jQuery**
   Mała i bogata w funkcje biblioteka JavaScript sprawiająca, że rzeczy takie jak np. obsługa zdarzeń, animacje czy Ajax stają się znacznie prostsze.

2. **Materialize**
   Front-endowy framework znacznie upraszczający i przyspieszający pisanie widoków aplikacji webowych.

3. **Google Maps**
   Skrypt umożliwiający wyświetlanie na stronie mapy (lokalizacji).

4. **Boostrap Tags Input**
   Dodatek do jQuery umożliwiający zarządzanie i wyświetlanie tagów w stylu Twittera.

5. **NoUISlider**
   Skrypt napisany w języku JavaScript, umożliwiający wyświetlanie i obsługę zaawansowanych suwaków na stronie.

6. **Material Icons**
   Czcionka od Google

7. **Typeahead**
   Skrypt umożliwiający wyświetlanie podpowiedzi

8. **Devise**
   Biblioteka znacznie znacznie ułatwiająca zaimplementowanie systemu użytkowników oraz ich autentykacje

9. **MySQL2**
   Biblioteka umożliwiająca użycie MySQL jako bazy danych Active Record

10. **Turbolinks**
    Przyspiesza przechodzenie pomiędzy linkami w aplikacji

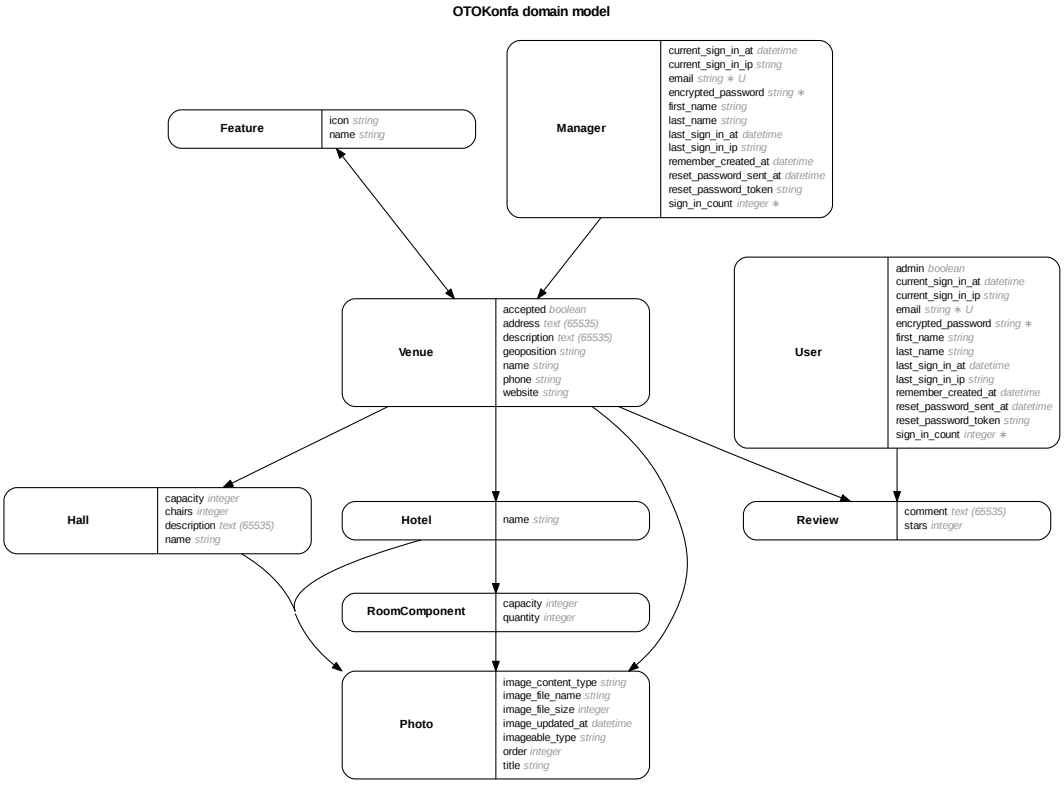
11. **jBuilder**
    Zależność ułatwiająca budowanie JSON'ów w aplikacji

12. **Sass-rails gem**
    Do kompilacji Sass na CSS

13. **Paperclip**
    Biblioteka do załączania zdjęć

2. Opis bazy danych

2.1. Diagram ERD



Rys. 2.1. Diagram ERD (zobacz pełny)

2.2. Krótki opis

Struktura bazy danych jest automatycznie generowana przez technologię w której wykonany został projekt (Ruby on Rails, a konkretnie jego komponent Active Record). Domyślnie projekt jest tworzony z użyciem adaptera do baz MySQL, jednak technologia oferuje łatwą możliwość podmiany adaptera na inny (PostgreSQL, SQLite i inne), bez konieczności obszernych zmian w kodzie.

3. Moduł wyszukiwania

Moduł odpowiedzialny za wyszukiwanie ośrodków spełniających kryteria użytkownika. Wywoływany przez akcję użytkownika na stronie internetowej, dostaje preferencje użytkownika (lokalizacja, ocena, atrybuty). Komunikuje się z modułem wyświetlania (przekazuje listę obiektów spełniających kryteria do wyświetlenia) oraz modułem dostępu do bazy danych (pobiera obiekty spełniające kryteria).

index() - sprawdza czy dany ośrodek spełnia kryteria wyszukiwania oraz wysyła go do modułu wyświetlania

Dane wejściowe (jeden lub więcej z poniższych):

- name (String) – nazwa ośrodka
- description (String) – opis ośrodka
- hall_size (String) – rozmiar sali (liczba potrzebnych miejsc)
- hall_count (String) – liczba sal
- average_rating (String) – ocena ośrodka
- attributes (String) – atrybuty ośrodka
- location (String) – lokalizacja ośrodka

Dane wyjściowe:

- venue (venue) - ośrodek spełniający kryteria

4. Moduł porównywania

Moduł odpowiedzialny za porównywanie dwóch lub więcej obiektów z bazy w przejrzysty sposób, odpowiednio podkreślając różnice między nimi. Komunikuje się z modułem dostępu do bazy danych (pobieranie danych) oraz z modułem wyświetlania (przedstawienie danych).

compare() - sprawdza czy ośrodek ma zostać dodany do porównywania oraz przekazuje go do modułu wyświetlania

Dane wejściowe:

- venue (venue) - sprawdzany ośrodek

Dane wyjściowe: brak

5. Moduł autentykacji

Moduł odpowiedzialny za rejestrację, logowanie oraz resetowanie hasła do konta. Komunikuje się z modułem dostępu do bazy danych (pobieranie i wysyłanie danych) oraz modułem obsługi maila (przekazuje wiadomości o utworzeniu konta/resetowaniu hasła do wysyłania). Moduł korzysta z gema „devise", który służy do autentykacji (https://rubygems.org/gems/devise/)

sign_in() - wywołuje Devise::SessionsController.new

sign_out() - wywołuje Devise::SessionsController.destroy

sign_up() - wywołuje Devise::RegistrationsController.new

reset_password() - wywołuje Devise::PasswordsController.edit

Wszystkie konieczne informacje znajdują się w dokumentacji:
https://github.com/plataformatec/devise

6. Moduł obiektu

Moduł odpowiedzialny za udostępnienie zalogowanemu menadżerowi na utworzenie nowego lub edycję dodanego przez niego ośrodka. Komunikuje się z modułem dostępu do bazy danych (pobiera/aktualizuje dane obiektu) oraz z modułem autentykacji (sprawdza czy użytkownik jest zalogowanym menadżerem).

check_manager_ownership!() - sprawdza czy użytkownik próbujący edytować ośrodek jest menadżerem tego ośrodka
Dane wejściowe:

- current_manager (manager) – obecnie zalogowany menadżer

Dane wyjściowe:

- check (boolean) – zmienna logiczna: prawda jeśli jest menadżerem ośrodka; fałsz jeśli nie jest

create() - tworzenie nowego ośrodka
Dane wejściowe:

- name (String) – nazwa ośrodka
- description (String) – opis ośrodka
- address (String) – adres ośrodka
- geoposition (String) – współrzędne geograficzne ośrodka
- phone (String) – numer kontaktowy ośrodka
- email (String) – email ośrodka

Dane wyjściowe: brak

destroy() - usuwa dany obiekt

Dane wejściowe: brak

Dane wyjściowe: brak

update() - uaktualnienie danych ośrodka
Dane wejściowe:

- name (String) – nazwa ośrodka
- description (String) – opis ośrodka
- address (String) – adres ośrodka
- geoposition (String) – współrzędne geograficzne ośrodka
- phone (String) – numer kontaktowy ośrodka
- email (String) – email ośrodka

Dane wyjściowe: brak

7. Moduł oceniania

Moduł pozwala zalogowanym użytkownikom na ocenę ośrodka wraz z dodaniem komentarza (opcjonalnego) oraz wyświetlenie listy ocen. Komunikuje się z modułem dostępu do bazy danych (pobiera/dodaje oceny) oraz z modułem obiektu (pobiera identyfikator danego obiektu).

create() - tworzy nową ocenę ośrodka
Dane wejściowe:

- stars (String) – ocena ośrodka w skali 1-5
- comment (String) – komentarz użytkownika

Dane wyjściowe: brak

show() - pokazuje listę ocen
Dane wejściowe:

- curren_tvenue (venue) – identyfikator obecnego ośrodka

Dane wyjściowe: brak

8. Moduł kontaktowy

Moduł odpowiedzialny za udostępnienie użytkownikowi formularzu, dzięki któremu może się on skontaktować z właścicielem interesującego go obiektu. Komunikuje się z modułem dostępu do bazy danych (pobiera dane) oraz modułem obsługi maila.

contact() - pobiera dane potrzebne do wysłania maila (podane poniżej) i przesyła je do modułu obsługi maila
Dane wejściowe:

- email (String) – adres email ośrodka
- name (String) – imię i nazwisko użytkownika
- phone (String) – numer telefonu użytkownika
- message (String) – wiadomość napisana przez użytkownika

Dane wyjściowe:

- email (String) – adres email ośrodka
- name (String) – imię i nazwisko użytkownika
- phone (String) – numer telefonu użytkownika
- message (String) – wiadomość napisana przez użytkownika

9. Moduł obsługi maila

Moduł odpowiedzialny za wysyłanie maili: od użytkowników do ośrodków oraz w wypadku resetowania hasła. Komunikuje się z modułem kontaktowym lub modułem autentykacji, od których dostaje potrzebne dane.

contact() - wysyła maila korzystając z danych wejściowych
Dane wejściowe:

- email (String) - adres email, na który ma zostać wysłany email
- name (String) - imię i nazwisko użytkownika
- phone (String) – numer telefonu użytkownika
- message (String) – wysyłana wiadomość

Dane wyjściowe: brak

Moduł oparty na ActionMailer z API Ruby on Rails (http://api.rubyonrails.org/classes/ActionMailer/Base.html).

10. Moduł dostępu do bazy danych

Moduł odpowiedzialny za komunikację z bazą danych: tworzenie, edytowanie, pobieranie, usuwanie danych.
Moduł jest w całości oparty na gemie "mysql2"(https://rubygems.org/gems/mysql2/) - oznacza to, że po stworzeniu i skonfigurowaniu bazy danych dalsza implementacja nie była potrzebna.

Wszystkie konieczne informacje znajdują się w dokumentacji: http://www.rubydoc.info/gems/mysql2/0.4.4

11. Moduł wgrywania plików

Moduł umożliwia dołączenie plików do systemu. Komunikuje się z modułem dostępu do bazy danych oraz systemem plików.

add() - dodaje plik do systemu
Dane wejściowe:

- path() - ścieżka do pliku

Dane wyjściowe: brak

delete() - usuwa wybrany plik z systemu
Dane wejściowe:

- title() - nazwa pliku

Dane wyjściowe: brak

12. Moduł galerii

Umożliwia edycję oraz prezentację zdjęć obiektu. Wykorzystuje moduł wgrywania plików do dodawania zdjęć do galerii.

edit() - edytuje nazwę zdjecia
Dane wejściowe:

- title() - nowa nazwa zdjecia

Dane wyjściowe: brak

add() - dodaje zdjęcie do galerii, przekazuje do modułu wgrywania plików
Dane wejściowe:

- path() - ścieżka do zdjęcia

Dane wyjściowe: brak

delete() - usuwa wybrane zdjęcie z galerii, przekazuje do modułu wgrywania plików
Dane wejściowe:

- title() - nazwa zdjęcia

Dane wyjściowe: brak

A. Diagramy ERD

Powrót do diagramów komunikacji w tekście...

OTOKonfa domain model