Projekt indywidualny,

Małgorzata Szwed

**Elektroniczny system obsługi siłowni: Twoja siłownia, online**

Celem projektu jest stworzenie dynamicznej aplikacji webowej we frameworku Ruby on Rails do obsługi elektronicznego systemu zarządzania siłownią.

W systemie współegzystują dwa rodzaje użytkowników:

-trenerzy

-osoby ćwiczące/kursanci

Trenerzy układają każdemu kursantowi plan treningowy (ćwiczenie/udział w zorganizowanych zajęciach) dostosowany do jego potrzeb i aktualnej kondycji. Analizują postępy kursantów. Układają dietę kursantom. Prowadzą konkretne zajęcia (plan zajęć).

Kursant – ma wgląd do planu treningowego na dany tydzień/miesiąc/okres. Monitoruje swoje postępy i urealnia termin osiąganych celów.

Taka wizja projektu daje możliwość znacznego jej rozbudowywania. Na cele uczelniane powstanie jednak tylko podstawowa funkcjonalność.

**Funkcje (w skrócie):**

-tworzenie użytkownika

-CRUD dla diet, trenerów, (użytkowników - wspomniane wcześniej), zajęć, wspisów w planie użytkowników

-trener może układać plan kursantowi

-wyświetlenia swojego planu

-zmiana danych osobowych w tym zdjęcia

-dostęp do polecanych przez siłownie diet

-dostęp do informacji o trenerach

-kontakt z trenerami

**Wymagania do uruchomienia aplikacji:**

-system (np. mac os x, linux, windows)

-interpreter języka ruby

-framework: ruby on rails

-przeglądarka internetowa (polecany Firefox lub Chrome)

-dostęp do terminala

-zainstalowane gemy z aplikacji w odpowiednich wersjach

-baza danych: sqlite (dla wersji developerskiej) oraz postgreSQL dla produkcyjnej

-(przydatne, nie wymagane) system wersjonowania git

**Potrzebne/Wykorzystane programy:**

-Git - system wersjonowania

-Rvm

-interpreter języka ruby

-rails

-postgresql - produkcyjna baza danych

-sqlite - developerska, lokalna baza danych

-rubyMine - IDE programistyczne

-vim - normalny edytor tekstu

-firefox - przeglądarka

-pentadactyl - potęrzna wtyczka do szybkiej obsługi firefoxa

-firebug - standardowa funkcjonalność FF (a także chroma itp.) pomagająca webdeveloperą w modelowaniu wyglądu strony

-Microsoft Word - do stworzenia dokumentacji

-heroku - platforma webowa do deployowania aplikacji RoR

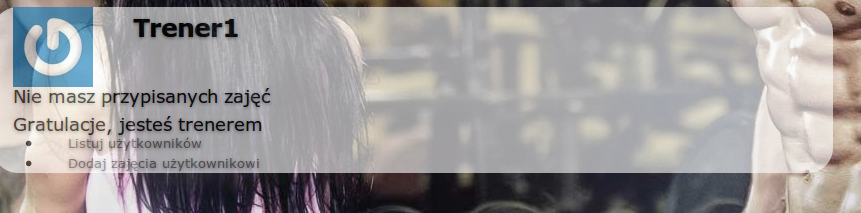
**Zastosowana platforma:**

-komputer lokalny: Ubuntu Linux (Linux lastop 3.13.0-35-generic #62-Ubuntu), 4GB ramu, ok. 2.3 GHz, 2 rdzenie, Intel Core 2 Duo, Asus LV50N

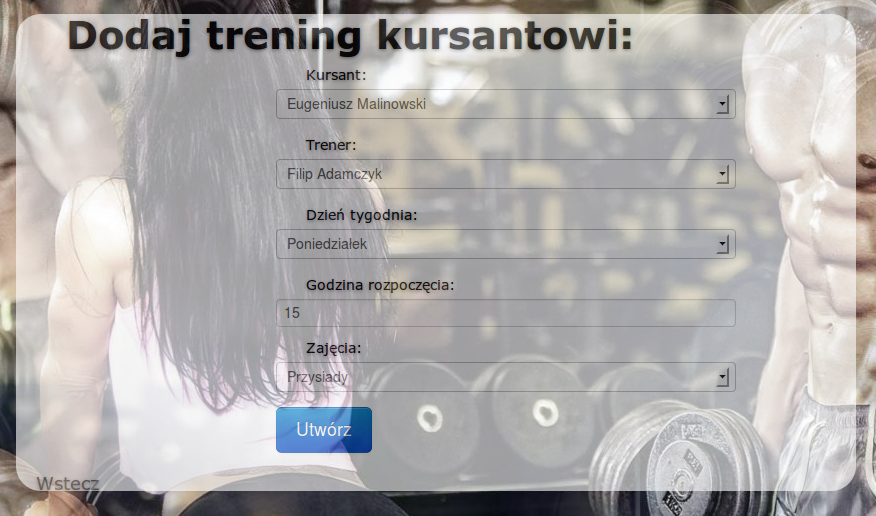
-platforma webowa: heroku

**Zrzuty ekranu:**

Niektóre mogę nie być w 100% aktualne, jako że aplikacja ciągle była udoskonalana. Widzimy natomiast, iż różnice są raczej kosmetyczne.



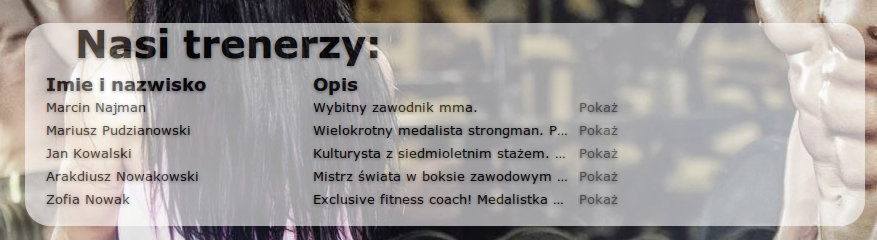
(panel trenera)



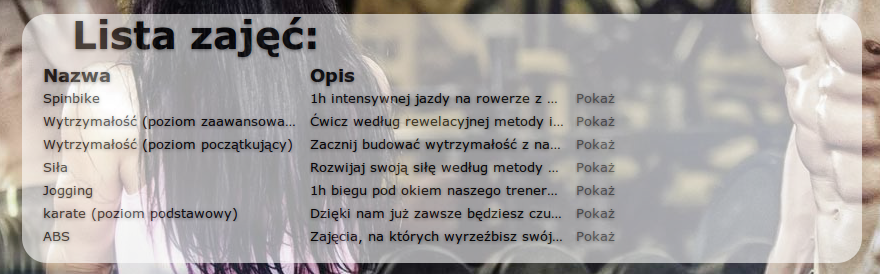
(trener -> dodawanie nowych zajęć dla kursanta; w oparciu o CRUD)



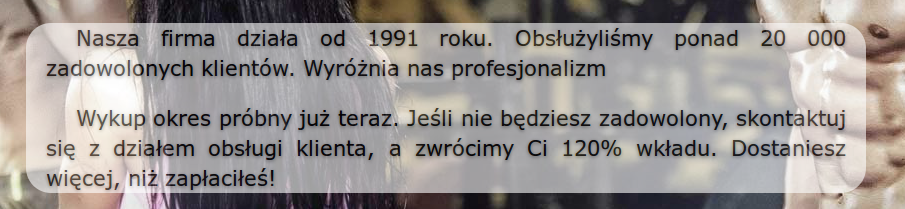
(diety polecane przez siłownie - dane wyświetlane dynamicznie z bazy danych)



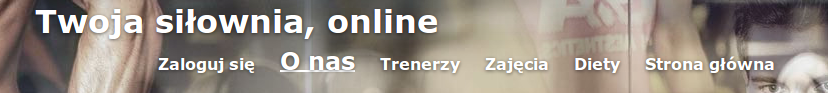
(licencjonowani trenerzy naszej siłowni wraz z opisem)



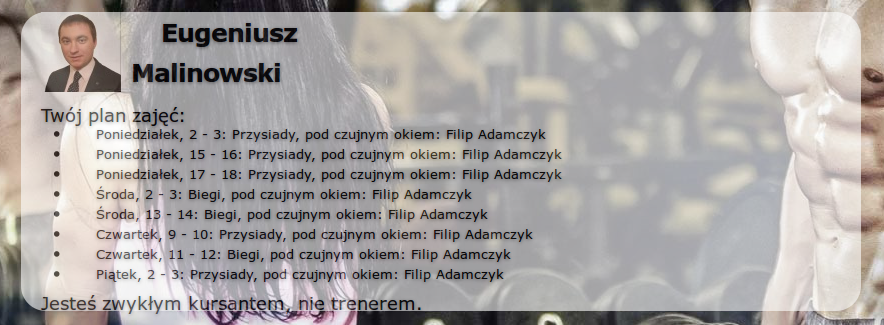
(przykładowe zajęcia oferowane przez siłownie)



(krótki opis siłowni)



(menu główne w formie bloku z dynamicznie zmieniającym się “wskazaniem” na aktualne podstronę)



(widok kursanta z aktualnym, posortowanym po dacie i godzinie planie treningowym oraz zdjęciem - obsługa zdjęcia przez serwis zewnętrzny: graviator)

**Szczegółowa informacja jak jest zbudowana aplikacja**

jest to typowa aplikacja we frameworku ruby on rails. Struktura katalogów jest standardowa, więc nie widzę sensu jej dogłębnego opisywania (informacje na ten temat można znaleźć np. tutaj: https://www.railstutorial.org/book/beginning#sec-the\_first\_application). Z rzeczy, na które należy zwrócić uwagę, gdyż zostały wygenerowane w trakcie tworzenia projektu (oczywiscie przy użyciu narzędzia rails generate scaffold/model/controller):

-scaffold (model + controller + widoki) dla:

--użytkownika (user)

--diet (diet)

--dni tygodnia (weekday) - prosta tabela zawierające polskie nazwy dni tygodnia

--trenera (coach)

--zajęcia (activity) - np. bieganie z krótkim opisem

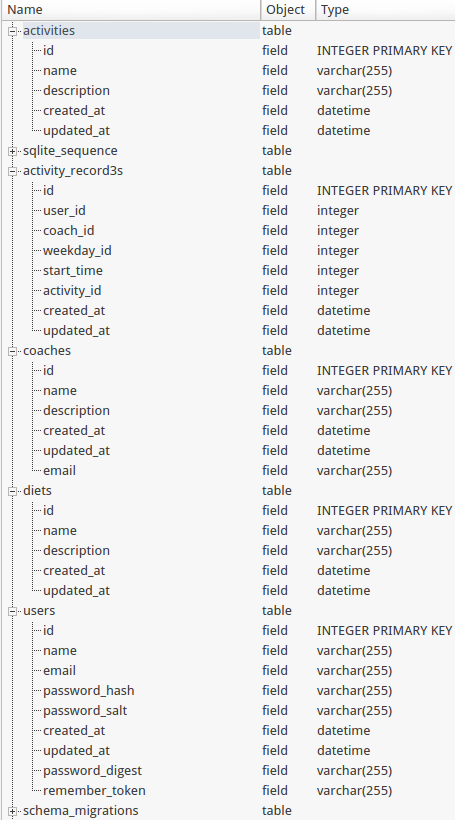
--zajęcia wpisane w planie (activity\_record3) - tabela łącząca id użytkownika, trenera oraz dnia tygodnia wraz z wpisem godzinowym

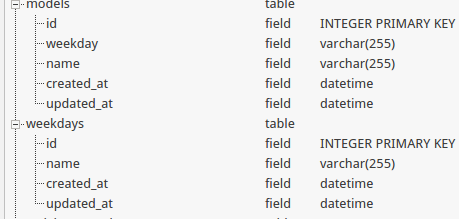
-controller

--sessions (sesje) - do obsługi system logowania

Oczywiście, w skończonej aplikacji nie zawsze wszystkie wygenerowane przez funkcje scaffold funkcjonalności będą używane, a wiele jest zmodyfikowanych.

**Opis modeli w bazie danych:**





Przy czym kolumny created\_at oraz updated\_at są stworzone automatycznie przez RoR i są transparentne dla programisty.

Wycinki ekranu są zrobione przez program shutter i pochodzą z programu SQLLite, który wyświetla developerską strukturę bazy danych. Produkcyjna baza stoi na postreSQL.

**Jak działa aplikacja?**

Ogólnie działanie opiera się na dobrze poznanym i szeroko stosowanym wzorcu projektowym: model-widok-kontroler.

Przykładowy opis dla przykładowej aplikacji w RoR: https://www.railstutorial.org/book/demo\_app#sec-mvc\_in\_action

Ogólnie, u nas:

kiedy użytkownik pierwszy raz otwiera stronę, aplikacja szuka w pliku konfiguracyjnym routes.rb, jaki kontroler i jaka funkcja (a poprzez funkcje także i widok) odpowiada danej ścieżce (w tym przypadku ścieżce głównej, domyślnej). Ta sama procedura tyczy się każdej innej ścieżki (istotnie, wszystkie dozwolone ścieżki wraz z metodami dostępu są wyszczegółowione w pliku routes.rb).

Następnie akcja zostaje przekazana do odpowiedniego kontrolera i odpowiedniej funkcji. Rozpoczyna się kalkulacja niezbędnych parakterów do wyświetlenia strony (charakterystyczny mechanizm np. dla django albo RoR) - kalkulowanie logiki podstrony aplikacji. Po zakończeniu, zostajemy przekierowani do konkretnego widoku, który może kożystać z wyliczonych wcześniej zmeinnych.

Istnieje jeden główny schemat wyglądu widoków (applications.html.erb) - podmieniane są tylko bloki treści/kontekstu (ramka po środku strony) oraz tytuł.

**Które cześći kodu pochodzą z innych źródeł?**

Trudno mi się właściwie oddnieś do tego pytania.

Część kodu została wygenerowana automatycznie (patrz: rails generate ...).

Wspierałam się także książką dostępna na railstutorial.org (możliwe, że można znaleźć trochę podobieństw do zaprezentowanej tam aplikacji) oraz stackoverflow. Nie mogę jednak powiedzieć, jakoby jakieś fragmenty były bezmyślnie kopiowane. Cały kod (nie wygenerowany automatycznie) napisałam własnoręcznie w oparciu o powyższe materiały.

Grafika tła pochodzi z (została przeze mnie zmodyfikowana w GIMPie):

http://www.gymaesthetics.com/

Uzyskałam mailową zgodę serwisu na wykożystanie grafiki.

**“﻿Pliki kodu źródłowego nie zawierają wymaganej informacji kto jest ich autorem, proszę takie informacje (jako komentarz) umieścić we wszystkich plikach tworzonych przez panią”**

Nie rozumiem, skąd takie polecenie. Wszystkie pliki, które nie są wygenerowane automatycznie przez narzedzia dostarczone do RoR (a więc pliki standardowe - o których wspominałam wcześniej przy okazji standardowej struktury aplikacji) zostały utworzone przeze mnie. Ponadto, modyfikowałam większość plików standardowych, aby dostosować je pod moją aplikacje.

**Szczegółowa dokumentacja:**

Etap 1

Nauka rubyiego I frameworku ruby on rails.

<http://www.railstutorial.org/book>

Instalacja oraz konfiguracja niezbędnego oprogramowania

Etap 2

Wizualizacja początkowa projektu; wstępny zarys; decyzje architektoniczne.

Etap 3

Skrótowy opis działań:

-stworzenie pustej aplikacji railsowej (rails new gym)

-skonfigurowanie Gemfile

-założenie repozytorium git I powiązanie z githubem

-dostosowanie gitignore

-ustawienie frameworków do testów automaczynych (rspec, spork, guard) oraz TDD/BDD (Capybara, Cucumber)

-powiązanie repozytorium z heroku, wysłanie strony na publiczny serwer

-stworzenie stron statycznych (główna, o nas)

-dostosowanie wyglądu (scss, twitter bootstrap), dobranie obrazów

-napisanie testów do stron statycznych sprawdzających zawartość

-wygenerowanie plików konfiguracyjnych (modelu, kotrolera, …) dla diet, zajęć oraz trenerów (rails generate scaffold …)

-dostosowanie praw dostępu do wygenerowanych stron (routes.rb)

-wpisanie przykładowych informacji na publiczny serwer heroku do bazy postgresql

-zablokowanie praw edycji od strony anonimowego użytkownika

-obsługa logowania

-obsługa gravataru

-obsługa planu zajęć

**Wnioski:**

Po długiej pracy udało się stworzyć portal dokładnie według założeń projektowych.

**Pozdrawiam,**

Małgorzata Szwed