UNIWERSYTET GDAŃSKI
Wydział Matematyki, Fizyki i Informatyki

Oskar Plichta
nr albumu: 195009

Budowa aplikacji modularnej do udostępniania fotografii w web 3.0

Praca magisterska na kierunku:
INFORMATYKA
Promotor:
dr W. Bzyl

Gdańsk 2015

Streszczenie

W pracy zostanie przedstawiony przyjazny dla użytkownika interfejs aplikacji internetowej w web 3.0. Interfejs zostanie zaimplementowany w programie PicDrop, który będzie spełniał założenia przyjaznego interfejsu. Celem aplikacji jest zbudowanie intuicyjnego interfejsu użytkownika do udostępniania fotografii wykonanego w *EmberJS*, *Bootstrap* oraz *Ruby on Rails*. Aplikacja ma na celu być prosta w obsłudze i ma za zadainie wyszukiwanie fotografi oraz ich łatwe udostępnianie dla wielu użytkowników jednocześnie. Ostatecznie aplikacja została wykonana zgodnie z założeniami i spełnia wyznaczone cele.

Słowa kluczowe

User Interface, Material Design, Ruby on Rails, EmberJS, PostgreSQL, RSpec, Jasmine

Spis treści

Wprowadzenie 5

1. Budowa aplikacji modularnej do udostępniania fotografii w web 3.0 . 6
- 1.1. Porównanie dostępnych rozwiązań 6
- 1.2. Możliwości zastosowania praktycznego 6

2. Projekt i analiza 7
- 2.1. Aktorzy i przypadki użycia, wymagania funkcjonalne i niefunkcjonalne 7
- 2.2. Diagram klas 7
- 2.3. Diagram modelu danych 7
- 2.4. Projekt interfejsu użytkownika w oparciu o framework Materialize 7
  - 2.4.1. Wytyczne odnośnie UI w Material Design 7

3. Implementacja aplikacji PicDrop 8
- 3.1. Architektura aplikacji PicDrop 8
- 3.2. Użyte technologie 8
- 3.3. API aplikacji 9
  - 3.3.1. Budowa API w oparciu o Ruby on Rails 9
  - 3.3.2. Połączenie z bazą danych PostgreSQL 9
  - 3.3.3. Dodawanie gemów w Ruby on Rails 9
  - 3.3.4. Połączenie z API różnych sieci społecznościowych 9
  - 3.3.5. Autoryzacja użytkowników 9
- 3.4. Front-end aplikacji 9
  - 3.4.1. Budowa Front-endu w oparciu o framework EmberJS . . 9
  - 3.4.2. Połączenie z API 9
  - 3.4.3. Opis narzędzia Ember-CLI 10
  - 3.4.4. Dodawanie wtyczek do aplikacji w EmberJS 10
- 3.5. Aplikacja mobilna na system Android 10

3.5.1. Zalety aplikacji mobilnej względem strony w mobilnej przeglądarce . . . 10
3.5.2. Tworzenie aplikacji przy użyciu narzedzia Cordova . . . 10

4. Testy . . . 11
4.1. Testowanie API przy użyciu RSpec . . . 11
4.1.1. Scenariusz testowania . . . 11
4.1.2. Raport z testów . . . 11
4.2. Testowanie Front-end przy użyciu Jasmine . . . 11
4.2.1. Scenariusz testowania . . . 11
4.2.2. Raport z testów . . . 11

5. Wkład własny . . . 12

Zakończenie . . . 13

A. Tytuł załącznika jeden . . . 14

B. Tytuł załącznika dwa . . . 15

Bibliografia . . . 16

Spis tabel . . . 17

Spis rysunków . . . 18

Oświadczenie . . . 19

Wprowadzenie

Interfejs użytkownika [1] jest podstawowym sposobem komunikacji pomiędzy człowiekiem a maszyną dlatego tak ważne jest, aby był on intuicyjny i przyjazny dla użytkownika. Postaram się pokazać na czym polega tworzenie przyjaznego UI, aby było intuicyjne dla użytkownika i pozwalało mu na wydajną pracę. Wskażę również z jakimi problemami musi się uporać *developer* aplikacji webowych, aby jego aplikacja była intuicyjna i funkcjonalna. Serwisy do wyszukiwania zdjęć, nie mają dobrego UI i dlatego postanowiłem zgłębić ten temat. Opierając się na doświadczeniach innych badaczy między innymi Roberta Hoekmana jr [1] oraz Jenifer Tidwell [2], którzy opisali swoje spostrzeżenia w ich książkach, postaram się napisać aplikacje PicDrop, która będzie miała przyjazne UI i pozwoli na łatwe udostępnianie treści. Opiszę dlaczego wybrałem *EmberJS*, *Bootstrap* oraz *Ruby on Rails* do stworzenia tej aplikacji i dlaczego te technologie uważam za najlepszy wybór.

[1] ang. *User Interface* - UI

ROZDZIAŁ 1

Budowa aplikacji modularnej do udostępniania fotografii w web 3.0

1.1. Porównanie dostępnych rozwiązań

1.2. Możliwości zastosowania praktycznego

Głównym celem aplikacji PicDrop jest proste i intuicyjne udostępnianie fotografii. Użytkownik może wgrać własne fotografie lub wyszukac poprzez wbudowaną wyszukiwarkę. Następnie wystarczy, że użytkownik przeciągnie wybrane zdjęcia do bocznego paska i kliknie guzik z symbolem portalu gdzie chce udostępnić swoje fotografie. To wszystko. Obsługa jest szybka i bezproblemowa.

ROZDZIAŁ 2

Projekt i analiza

2.1. Aktorzy i przypadki użycia, wymagania funkcjonalne i niefunkcjonalne

Aplikacja wczytuje zdjęcia z bazy danych PostgreSQL, wysyła je przez JSON do klienta i tam EmberJS odpowiednio obrabia dane pokazując je w formie przyjaznej użytkownikowi. Następnie, gdy chcemy udostępnić jakiś plik to jest to obsługiwane także przez EmberJS, tak aby komunikacja była szybka i niezawodna. Jednakże jeśli użytkownik wczytuje własne zdjęcia, które chce udostępnić to są one zamieniane na ciąg znaków w standardzie Base64Image i wysyłane wiadomością JSON do API. Tam back-end łączy się z wybranym portalem społecznościowym i przesyła do niego zdjęcia.

2.2. Diagram klas

2.3. Diagram modelu danych

2.4. Projekt interfejsu użytkownika w oparciu o framework Materialize

2.4.1. Wytyczne odnośnie UI w Material Design

ROZDZIAŁ 3

Implementacja aplikacji PicDrop

3.1. Architektura aplikacji PicDrop

Aplikacja składa się z dwóch części. Pierwszą z nich jest serwer API, które zostało stworzone w technologi Ruby on Rails w oparciu o gem rails-api. API jest pośrednikiem między bazą danych PostgreSQL a drugą częścią aplikacji czyli klientem stworzonym w EmberJS, który jest odpowiedzialny za warstwę wizualną aplikacji. Obie części komunikują się poprzez JSON. Dzięki takiemu rozwiązaniu można stworzyć dodatkowych klientów, na przykład do aplikacji mobilnych.

3.2. Użyte technologie

Spośród mnóstwa technologii do tworzenia interfejsów użytkownika, najbardziej przodujące są oparte te na języku *JavaScript* takie jak *Bootstrap*, *jQuery* czy *AngularJS*. Opierając się na artykułach [] oraz [] postanowiłem wybrać *EmberJS*, *Materialize* oraz *Ruby on Rails*. UI zostanie wykonane w *EmberJS*, jest to biblioteka *open-source* języka *JavaScript* stworzona przez Yehuda Katz'a Tom Dale'a, która jest cały czas usprawniana przez społeczność. Posiada ona szereg mechanizmów ułatwiających developerom tworzenie UI na jej podstawie, ma także czytelną i przejrzystą dokumentację. UI aplikacji zostanie dodatkowo upiększone poprzez *framework Materialize*, a *Jasmine* pozwala na testowanie kodu w JavaScript. Na serwer wybrałem sprawdzonego *Ruby on Rails*, które pozwala na szybkie tworzenie aplikacji internetowych oraz dzięki narzędziu *RSpec* na łatwe testowanie kodu.

3.3. API aplikacji

3.3.1. Budowa API w oparciu o Ruby on Rails

3.3.2. Połączenie z bazą danych PostgreSQL

3.3.3. Dodawanie gemów w Ruby on Rails

3.3.4. Połączenie z API różnych sieci społecznościowych

3.3.5. Autoryzacja użytkowników

3.4. Front-end aplikacji

3.4.1. Budowa Front-endu w oparciu o framework EmberJS

Warstwa wizualna aplikacji czyli tak zwany front-end został wykonany przy użyciu frameworka EmberJS. Dzięki temu łatwiej zarządzać tą cześcią aplikacji. EmberJS opiera się na wzorcu Model-View-Controler (pol. Model-Widok-Kontroler) co oznacza, że składa się z trzech podstawowych części opowiedzialnych za różne akcje. Model jest pewną reprezentacją problemu bądź logiki aplikacji. W naszym przypadku model jest szablonem z opisanymi typami danych danego zasobu np. zdjęcia. Model również komunikuje się z Ember Data czyli warstwą aplikacji bezpośrednio komunikującą się z serwerem. Widok opisuje, jak wyświetlić pewną część modelu w ramach interfejsu użytkownika. W EmberJS widok składa się z tak zwanych templates czyli kodem HTML z aktywnie zmieniającymi się częściami. Kontroller przyjmuje dane wejściowe od użytkownika i reaguje na jego akcje, zarządzając aktualizacje modelu oraz odświeżenie widoków. Kontroler w EmberJS pobiera dane z modelu oraz zarządza akcjami w widoku.

3.4.2. Połączenie z API

Połączenie z serwerem aplikacji jest jedną z najważniejszych rzeczy w całej aplikacji. To dzięki niemu możemy się komunikować z serwerem oraz przetwarzac dane z bazy danych. EmberJS do pobierania i przetwarzania danych korzysta z biblioteki

Ember Data. Ma ona kilka wbudowanych tzw. adapterów do połączenia z różnymi typami serwerów napisanych w różnych językach i technologiach. W mojej aplikacji używam ActiveModelAdapter, który działa bezproblemowo z ActiveModel w Ruby on Rails przeprowadzając serializacje i deserializację danych z EmberJS do wiadomości JSON.

3.4.3. Opis narzędzia Ember-CLI

Ember-CLI (Command Line Interface) to program narzędziowy, który zarządza aplikacją napisaną we frameworku EmberJS. Głównymi zaletami korzystania z tego narzędzia jest zarządzanie plikami, zależnościami, proste dodawanie wtyczek, uruchamianie serwera, generowanie konkretnych części programu jak np. model lub kontroler.

3.4.4. Dodawanie wtyczek do aplikacji w EmberJS

Wtyczki są to programy dodające jakąś funkcjonalność do naszego programu, dzięki czemu nie musimy wszystkich elementów pisać od podstaw. Wystarczy jak poszukamy wtyczki zapewniającej nam funkcjonalność, której szukamy np. t17-ember-upload pozwala na wgrywanie zdjęć do naszej aplikacji porzez proste Drag & Drop (pol. Przeciągnij i upuść). Aby dodać wtyczkę wystarczy zainstalować ją przy użyciu NPM lub bowera po czym dodać app.import w pliku Brocfile.js. Następnie należy zaimportować konkretny moduł z wtyczki do naszej części aplikacji i już można używac widoku oraz akcji z wtyczki.

3.5. Aplikacja mobilna na system Android

3.5.1. Zalety aplikacji mobilnej względem strony w mobilnej przeglądarce

3.5.2. Tworzenie aplikacji przy użyciu narzedzia Cordova

ROZDZIAŁ 4

Testy

4.1. Testowanie API przy użyciu RSpec

4.1.1. Scenariusz testowania

4.1.2. Raport z testów

4.2. Testowanie Front-end przy użyciu Jasmine

4.2.1. Scenariusz testowania

4.2.2. Raport z testów

ROZDZIAŁ 5

Wkład własny

Zakończenie

DODATEK A

Tytuł załącznika jeden

Treść załącznika jeden.

DODATEK B

Tytuł załącznika dwa

Treść załącznika dwa.

Bibliografia

[1] Robert Hoekman jr. *Magia interfejsu. Praktyczne metody projektowania aplikacji internetowych*. Helion, 2010.

[2] Jenifer Tidwell. *Projektowanie interfejsów. Sprawdzone wzorce projektowe*. Helion, 2012.

[3] Don Norman. *The Design of Everyday Things*. Basic Books, 2002.

[4] Steve Krug. *Nie każ mi myśleć! O życiowym podejściu do funkcjonalności stron internetowych*. Helion, 2012.

[5] DouglasCrockford. *JavaScript - Mocne Strony*. Helion, 2011.

[6] Joe Fiorini. User Interface Thinking in Rails: An Example. 2012.

[7] Rolf Hennicker Nora Koch. Modeling the User Interface of Web Applications. 2001.

[8] Rspec Docs dostęp 2015-05-12.

[9] Ruby on Rails Docs dostęp 2015-05-12.

[10] EmberJs Docs dostęp 2015-05-12.

[11] Ember CLI Docs dostęp 2015-05-12.

[12] Ember Rails dostęp 2015-05-12.

[13] EmberJS Tutorial dostęp 2015-05-12.

[14] Introduction to Ember dostęp 2015-05-12.

Spis tabel

Spis rysunków

Oświadczenie

Ja, niżej podpisany(a) oświadczam, iż przedłożona praca dyplomowa została wykonana przeze mnie samodzielnie, nie narusza praw autorskich, interesów prawnych i materialnych innych osób.

.....	.....
data	podpis