

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 program do udostępniania fotografii na kilka serwisów społecznościowych jednocześnie z intuicyjnym interfejsem Material Design. Dzięki temu w prosty i szybki sposób można podzielić się swoimi zdjęciami z innymi użytkownikami kilku sieci społecznościowych. Aplikacja będzie wykonana z dwóch modułów - części serwerowej i wizualnej. Część serwerowa jest oparta o *Ruby on Rails* a wizualna wykonana w *EmberJS* oraz *Materialize*. Aplikacje mobilna dla systemu iOS oraz Android wykonane zostały dzięki aplikacji Cordova, która pozwala przekształcić aplikację opartą o *EmberJS* oraz *Ember CLI* w natywną aplikację dla danego systemu mobilnego. 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	7
1.1. Porównanie dostępnych rozwiązań	7
1.2. Możliwości zastosowania praktycznego	7
2. Projekt i analiza	8
2.1. Aktorzy i przypadki użycia, wymagania funkcjonalne i нефункционалне	8
2.2. Diagram klas	8
2.3. Diagram modelu danych	8
2.4. Projekt interfejsu użytkownika w oparciu o framework Materialize	8
2.4.1. Wytyczne odnośnie UI w Material Design	9
3. Implementacja aplikacji PicDrop	10
3.1. Architektura aplikacji PicDrop	10
3.2. Użyte technologie	10
3.3. API aplikacji	11
3.3.1. Budowa API w oparciu o Ruby on Rails	11
3.3.2. Połączenie z bazą danych PostgreSQL	11
3.3.3. Dodawanie gemów w Ruby on Rails	11
3.3.4. Połączenie z API różnych sieci społecznościowych	11
3.3.5. Autoryzacja użytkowników	11
3.4. Front-end aplikacji	11
3.4.1. Budowa Front-endu w oparciu o framework EmberJS	11
3.4.2. Połączenie z API	12
3.4.3. Opis narzędzia Ember-CLI	12
3.4.4. Dodawanie wtyczek do aplikacji w EmberJS	12
3.5. Aplikacja mobilna na system Android	13

3.5.1.	Zalety aplikacji mobilnej względem strony w mobilnej przeglądarce	13
3.5.2.	Tworzenie aplikacji przy użyciu narzędzia Cordova	13
4.	Testy	14
4.1.	Testowanie API przy użyciu RSpec	14
4.1.1.	Scenariusz testowania	14
4.1.2.	Raport z testów	14
4.2.	Testowanie Front-end przy użyciu Jasmine	14
4.2.1.	Scenariusz testowania	14
4.2.2.	Raport z testów	14
5.	Wkład własny	15
	Zakończenie	16
A.	Tytuł załącznika jeden	17
B.	Tytuł załącznika dwa	18
	Bibliografia	19
	Spis tabel	20
	Spis rysunków	21
	Oświadczenie	22

Wprowadzenie

Fotografie są jednym z najczęstszych typów danych przesyłanych w Web 3.0. Gwałtowny rozrost sieci społecznościowych spowodował, że prawie każdy udostępnia zdjęcia aby podzielić się nimi z rodziną i przyjaciółmi. Portale takie jak Facebook, Flickr czy Twitter prześcigają się w tym aby wysyłanie zdjęć na ich serwer było jak najprostsze. Większość z nich pozwala na tzw. drag and drop¹ fotografii oraz na wysyłanie ich do innych serwisów. Zakładając, że mamy zdjęcia z wakacji i chcemy je udostępnić na Facebooku oraz umieścić na naszym koncie Flickr w celu archiwizacji musimy zalogować się na Facebooka, następnie wysłać zdjęcia na serwer Facebooka, ewentualnie dopisać opis i kliknąć w przycisk do udostępniania, po czym całość powtórzyć na serwisie Flickr. Temat wysyłania zdjęć do kilku serwisów jednocześnie, tak aby nie jeszcze raz nie powtarzać tej samej czynności lecz na innym serwisie społecznościowym, pozostaje otwarty i dlatego postanowiłem go zgłębić. Aplikacja, którą opisuje w tej pracy pozwala na jednoczesne wysyłanie zdjęć na kilka serwisów społecznościowych za pomocą kilku kliknięć. Wystarczy wybrać zdjęcia dzięki drag and drop, zalogować się na wybrane przez nas serwisy a następnie kliknąć wyślij. Wszystko przebiega szybko i sprawnie a my oszczędzamy nasz czas. Aplikacja ta jest tzw. aplikacją modułową co znaczy, że w przeciwieństwie do aplikacji monolitycznej, składa się z niezależnych od siebie części tj. serwerowej oraz wizualnej. Obie części komunikują się ze sobą za pomocą wiadomości JSON. Część serwerowa odpowiada za komunikację z bazą danych, komunikację z serwerami zewnętrznymi oraz autoryzację użytkowników i wysyłanie danych do front-endu czyli aplikacji wizualnej. Zostanie ona wykonana w języku Ruby i frameworku *Ruby on Rails*. Aplikacja wizualna, która zostanie wykonana w języku JavaScript i frameworku *EmberJS*, ma za zadanie wyświetlanie danych w przystępnej formie dla użytkownika poprzez tzw. interfejs. Omówię zagadnienie aplikacji modularnej i monolitycznej bardziej szczegółowo w jednym z kolejnych rozdziałów. Interfejs użytkownika² jest podstawowym sposobem komunikacji pomiędzy

¹ang. *drag and drop* - przeciągnij i upuść

²ang. *User Interface* - UI

człowiekiem a maszyną dlatego tak ważne jest, aby był on intuicyjny i przyjazny dla użytkownika. Postaram się pokazać dlaczego UI w mojej aplikacji jest przyjazny, intuicyjny dla użytkownika i pozwala mu na wydajną pracę a wszystko dzięki wytycznym Google Material Design, które jest nowym designem dla aplikacji od firmy Google. Więcej na temat Material Design oraz interfejsu aplikacji PicDrop w kolejnych rozdziałach.

Większość smartphonów posiada dobrej jakości aparaty, zarówno z przodu jak i z tyłu urządzenia, dlatego w każdej chwili możemy wysłać zdjęcia z dowolnego miejsca do rodziny i przyjaciół poprzez sieci społecznościowe, za pomocą natywnej aplikacji na dany system mobilny. Dzięki temu, że moja aplikacja jest modułowa, jej część serwerowa czyli API pozwala na połączenie z nią różnych części wizualnych tzw. front-end z systemów mobilnych takich jak np. iOS lub Android. Jest to możliwe dzięki aplikacji Cordova, która zamienia aplikację *EmberJS* w natywną aplikację na dany system mobilny przez co jest ona łatwiejsza w obsłudze i lepiej wykorzystuje mały wyświetlacz smartphona niż aplikacja przeglądarkowa. Opiszę dokładniej ten proces w odpowiednim rozdziale oraz podam inne zalety aplikacji natywnej. Wskażę również z jakimi problemami musi się uporać *developer* aplikacji webowych, aby jego aplikacja była intuicyjna i funkcjonalna. Wyjaśnię, które elementy mojej aplikacji testuję i w jaki sposób. 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 aplikację 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.

Budowa aplikacji modularnej do udostępniania fotografii w web 3.0

1.1. Porównanie dostępnych rozwiązań

Każdy serwis społecznościowy posiada swoje możliwości udostępniania zdjęć, które ograniczają się do jednego serwisu. Tak więc jeśli chcemy wgrać swoje zdjęcia jednocześnie na Facebooka oraz Flickr w celu archiwizacji to musimy je wgrać na jeden z tych dwóch serwisów a następnie na kolejny. Takie rozwiązanie zajmuje dużo czasu i jest niekorzystne dla użytkownika.

1.2. Możliwości zastosowania praktycznego

Głównym celem aplikacji PicDrop jest proste i intuicyjne udostępnianie fotografii. Użytkownik może wgrać własne fotografie poprzez przeciągnięcie do okna przeglądarki. Następnie wystarczy, że użytkownik kliknie guzik z symbolem portalu gdzie chce udostępnić swoje fotografie i zaloguje się na wybrany portal społecznościowy. To wszystko. Obsługa jest szybka i bezproblemowa a my oszczędzamy nasz czas.

ROZDZIAŁ 2

Projekt i analiza

2.1. Aktorzy i przypadki użycia, wymagania funkcjonalne i нефункционалне

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

Interfejs użytkownika musi być czytelny i łatwy w obsłudze a jednocześnie pozwalać na wydajną pracę. W aplikacji PicDrop główny nacisk położyłem na intuicyjność i szybkość działania. Te założenia spełnia wybieranie zdjęć przez drag and drop dzięki czemu możemy szybko umieścić fotografie, na których nam najbardziej zależy w oknie aplikacji.

2.4.1. Wytyczne odnośnie UI w Material Design

Google Material Design został zaprezentowany pierwszy raz na Google I/O w 2014 roku wraz z systemem Android i jego wersją Lollipop. Mimo, że od tamtego wydarzenia minęło już trochę czasu to jest on używany przez Google w kolejnych edycjach systemu Android. Głównymi założeniami tego designu jest to że składa on się z kilku nałożonych na siebie warstw papieru w różnych kolorach. Dzięki temu mamy wrażenie głębi a wszystko jest przejrzyste i intuicyjne. Dodatkowo animacje w Material Design dopasowują się do kolejnej wytycznej czyli do takiego animowania elementów na ekranie, żeby użytkownik miał najważniejsze elementy cały czas na widoku a nie, żeby one pojawiały się znikąd.

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 technologii 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

Część serwerowa aplikacji może zostać wykonana w wielu językach programowania takich jak Ruby, Python, JavaScript czy PHP. Wybrałem język Ruby, ponieważ jest on szybkim językiem skryptowym oraz posiada stabilny i sprawdzony framework *Ruby on Rails*, który pozwala na szybkie tworzenie aplikacji internetowych oraz dzięki narzędziu *RSpec* na łatwe testowanie kodu. 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 [1] oraz [2] 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 upiękkszzone poprzez framework *Materialize*, a *Jasmine* pozwala na testowanie kodu w *JavaScript*.

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

Jedną z największych zalet Ruby on Rails jest dopasowanie się frameworka do naszych potrzeb za pomocą minibibliotek tzw. gemów, tworzonych przez developerów z całego świata. W sieci można znaleźć tysiące różnych gemów, a każdy z nich posiada przeważnie tylko jedno specyficzne zadanie. Gemy, które są w ciągłym użytku są na bieżąco aktualizowane, dzięki czemu są poprawiane błędy i dodawane nowe funkcje. Przykładowo jeżeli chcemy testować naszą aplikację, możemy to zrobić dzięki gemowi RSpec. Pozwala on na testowanie różnorodnych elementów naszej aplikacji od walidacji pól w formularzach do kontrolerów i zapisywania danych w bazie.

Aby dodać nowy gem do naszej aplikacji wystarczy dodać jego nazwę do pliku Gemfile w głównym katalogu naszej aplikacji go zapisać. Następnie wystarczy w terminalu otworzyć folder z naszą aplikacją i wpisać komendę "bundle install". Ruby on Rails automatycznie zainstaluje najnowszą wersję danego gemu, chyba, że wpisaliśmy konkretną wersję w pliku Gemfile. Całość przebiega szybko i sprawnie.

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 częścią aplikacji. EmberJS opiera się na wzorcu Model-View-Controller (pol. Model-Widok-Kontroler)

co oznacza, że składa się z trzech podstawowych części odpowiedzialnych 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. Kontroler 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 przetwarzać 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 serializację 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 poprzez 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 używać 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 narzędzia 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