POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK:	INFORMATYKA (INF)
SPECJALNOŚĆ:	INŻYNIERIA SYSTEMÓW INFORMATYCZNYCH (INS)

PRACA DYPLOMOWA INŻYNIERSKA

A business intelligence web application for mobile application trends analysis

Aplikacja webowa do analizy trendów w aplikacjach mobilnych

AUTOR:
Jakub Pomykała

PROWADZĄCY PRACĘ:
dr hab. inż. Krzysztof Walkowiak, prof. PWr

OCENA PRACY:

WROCŁAW, 2016

Opracował: Tomasz Kubik <tomasz.kubik@pwr.edu.pl>
Data: maj 2016

Streszczenie

Celem niniejszej pracy dyplomowej było stworzenie aplikacji webowej w języku Java 8 przy użyciu frameworka Spring MVC. Frontend został stworzony z wykorzystaniem frameworka Twitter Bootstrap oraz narzędzia Thymeleaf. Klient jest w stanie wyszukać osoby potencjalnie najbardziej zainteresowane daną aplikacją mobilną. Rekomendacje aplikacji są wyliczane na podstawie recenzji napisanych przez użytkownika w sklepie Google Playna podstawie recenzji w sklepie Google Play, które napisał użytkownik. Do analizy ocen aplikacji został wykorzystany algorytm *slope one* zaprezentowany w 2005 roku przez Daniela Lemire i Annę Maclachlan [1]. Jest to algorytm rekomendacyjny przetwarzający oceny użytkowników, którego pośrednim wynikiem działania jest macierz podobieństwa pomiędzy aplikacjami. Zbudowana macierz służy do generowania rekomendacji przedmiotów, którymi, w tym przypadku, są aplikacje mobilne. Baza danych do analizy pochodzi z komentarzy pisanych na stronie sklepu z aplikacjami dla systemu Android - Google Play. Ponadto aplikacja służy również do wyszukiwania aplikacji mobilnych z wybranych sklepów (Google Play oraz Apple Store) według zadanych kryteriów takich jak: ilość ocen, liczba instalacji, nazwa aplikacji, średnia ocena aplikacji i inne. Usługi oferowane przez aplikację są płatne, dlatego za każde wyświetlone dane używające filtrowania oraz za wyszukiwanie rekomendacji naliczane są punkty. Płatności są obsługiwane przez zewnętrzny system specjalnie do tego przeznaczony - PayPal, dzięki któremu można zakupić więcej punktów niezbędnych do otrzymania informacji o aplikacjach i potencjalnych klientach. Użytkownicy systemu mają możliwość zalogowania się do aplikacji używając swojego konta Google+ lub adresu e-mail i hasła. Dane te są zapisywane w bazie danych MySQL. Natomiast pobrane dane aplikacji, użytkowników oraz wygenerowane preferencje aplikacji są zapisywane w bazie danych MongoDB. Aplikacja skierowana jest przede wszystkim do młodych twórców aplikacji mobilnych w systemie Android, którzy po wydaniu swojej aplikacji w sklepie Google Play, mogą wyszukiwać osoby potencjalnie zainteresowane ich produktem. Do konfiguracji modułów takich jak: Nginx, Tomcat, MySQL oraz MongoDB, zostało użyte narzędzie o nazwie Docker. Projekt spełnił wszystkie założenia i może zostać z powodzeniem zostać wdrożony jako w pełni działający system rekomendacji. Głównymi wyzwaniami w projekcie były:

- implementacja algorytmu *slope one*,
- wybór odpowiedniego sposobu przechowywania dużej ilości danych, który zapewni szybki dostęp do informacji podczas analizy,
- implementacja logowania przez zewnętrzny system autoryzacji - Google Plus,
- obsługa płatności.

Wykorzystywane technologie w projekcie:

- Java 8
- Spring Framework 4.3
- Thymeleaf 3.0
- MySQL 5.7
- MongoDB 3.2
- Twitter Bootstrap 3.3

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Goals of the Project

The main goal of project was to create web application to analyze trends in mobile applications. To achieve this one of the recommender algorithm has been implemented. Application is written in Java 8 using Spring Framework. Needed data to analysis, are gathered from mobile stories like Google Play and Apple Store. All information about applications are stored in database and used to filtering it according to some given criteria. It could be useful tool for many people, especially developers who wants to find proper audience for their applications. Moreover application is saving user reviews about applications from Google Play and fetching contacts if somebody register in application using Google Plus account. All this data are used to generate recommendations for those people who haven't specific application, but it could be interested for them to download it. To creating recommendations *slope one* has been used; it is very efficient and easy to implement recommendation algorithm. All features are paid, that is why application supports payment through PayPal system. Whole application is taking advantage of Model-View-Controller design pattern, which ensures clarity in project structure. Spring web applications need to be deployed to server with configured application container. In this project as application container has been used Tomcat 8.0. Project contains four modules which are needed to work on remote server as web application:

- MySQL as database to store user registration data, personal information and other,
- MongoDB as second database to store data fetched from web (reviews, person information and data necessary for *slope one* algorithm like matrix),
- Tomcat as application container where application is deployed,
- Nginx as HTTP server which in this case work as proxy server to speed up loading.

To configure all these components Docker were used. Docker is image distribution for server templates, which helps in building, shipping and running components. It allows to create module from image. Images are distributed on Docker website as repository. All framework and tools in project were free.

Chapter 2

Naming Conventions and Definitions

- User - person who express opinion about items
- Item - object which is rated by users
- Like-minded people - users who express similar opinion for same subject
- Worker - module without graphical interface
- Front-end - graphical user interface, visible part for client
- Back-end - place where business logic occurs, content which is not visible for client
- Model - unit with parameters which are trained by learning data set
- Learning data set - chosen data set to train model
- Recommendation rate - calculated prediction for user, higher value means more chance to like item by user
- Prediction rate - same as *recommendation rate*
- Similarity rate - calculated between two items or users, higher value means object are more similar
- JSON - key-value data format
- Bash - script language and UNIX shell

MVC	(*Model View Controller*) design pattern widely used in many applications
API	(*Application Programming Interface*)
ML	(*Machine Learning*)
JVM	(*Java Virtual Machine*)
CRUD	(*Create Read Update Delete*)
WAR	(*Web Application Archive*)
ORM	(*Object Relational Mapping*)
J2EE	(*Java Enterprise Edition*)
AOP	(*Aspect Oriented Programming*)
ERD	(*Entity-Relationship Diagram*)
NLP	(*Natural language processing*)
CPU	(*Central Processing Unit*)

Chapter 3

Relevant Facts and Assumptions

3.1. Relevant Facts

Slope one is the most popular and one of the most efficient recommendation algorithms for rating-based filtering. Introduced in February 2005 by Daniel Lemire and Anna Maclachlan. The biggest advantage of this algorithm is easy implementation, good performance, and accurate results. *Slope one* do not need many data to be able to predict with well accuracy recommendations for new users in system [1]. Recommender algorithms can be divided into different types:

- Collaborative filtering - which relies on earlier user behavior, gathering massive number of data sets and analyze them. Occurs "cold start problem" when algorithm cannot predict with enough accuracy and needs more data to properly recommends items [2].
- Content-Based filtering - more important is model than amount of data. Accuracy depends on item description and user preferences, predicting is based on matching keywords. Model is trained by learning data set and accuracy of algorithms largely depends on model rather than amount of data [2].
- Hybrid recommender systems [3] - combined collaborative and content-based filtering to give better results and omit weak points, like "cold start problem".

In collaborative filtering there are two subgroups:

- User-User - relies on similarity between users. User preferences are changing dynamically, it requires often rebuilding of similarity matrix between them, which is very time-consuming.
- Item-Item - relies on similarity between items. Similarity between items is more constant and more immune to errors.

Slope one belongs to collaborative filtering group, item-item subgroup. The main idea of algorithm is divided on two steps:

1. Build similarity matrix between items,
2. recommend items.

First step is very time-consuming, because it needs to calculate similarity rate between every item in a database. Performance largely depends on accessing data from the database. In second step recommendations for people were generated. Recommendations are calculated only for users who not expressed preference for given item. In this step created similarity matrix is used and user ratings for item which users express preference.

3.2. Assumptions

3.2.1. Application Requirements

The project assumes to create an application environment for the user who should be able to use all of features by web interface in any web browser. Below were listed main features which should be present in the application:

- The User should can register and sign in to the application using email or Google Plus an account. The account should be saved in MySQL database.
- User should have opportunity to show persisted applications by chosen criteria.
- User should be able to see like-mined people for given application name which has generated recommendations.
- The application should simultaneously: build similarity matrix, generate recommendations, gather data from Google Play and Apple Store.
- The application gathered data should be saved in a MongoDB database.

3.2.2. Chosen Frameworks

Programming language

Java 8 has been chosen as primary programming language J̇ava was introduced in 1995, invented by James Gosling from Sun Microsystems company and maintained by Oracle company. It is object oriented programming language complied to byte code which is executed by JVM. Every system has own implementation on JVM, that is why Java programs can be executed on any system which has installed Java Virtual Machine [4].

Web interface

Web interface has been created with the aid of Spring MVC, which is a lightweight framework for building enterprise web applications. It was introduced in 2002 and current version which is used in project was released in June 2016. Framework is written in Java and maintained by

Pivotal Software on Apache License 2.0 [5]. One of the biggest advantage of chosen framework is it's modularity, this project take advantage from many of Spring subprojects:

- Spring Data - to providing simple and consistent access to database any kind,
- Spring Security - used in authorization mechanism based on session,
- Spring AOP - allows for aspect oriented programming,
- Spring Test - helps in unit testing.

Thymeleaf has been used, to built interface in HTML language. Thymeleaf is template engine which generates HTML page on server side and transfers it to client's web browser [6]. In order to speed up of interface creation, the project is taking advantage from Bootstrap 3 framework. This framework has been created and maintained by Twitter company. Allows for easy front-end development for desktop and mobile devices [7].

Database access

Spring Data is very useful tool in modern Spring applications. The main mission of this subproject is to provide consistent interface across many supported databases [8]. This project is divided into many subprojects, but the application uses only three of them:

- Spring Data commons - utility objects for easier model data management,
- Spring Data JPA - support for MySQL database,
- Spring Data Mongo - support for MongoDB database.

Moreover to provide even more consistent and fluent API to manage two different databases, the application uses QueryDSL framework. Due to this framework can be created queries abstracted from used database. It is very good solution in applications with more than two different databases [9].

3.2.3. Architecture Overview

As previously has been mentioned, the application were shared on two server instances to provide good performance and availability:

- **Server instance 0 - web interface** - responsible for serving HTML pages as user interface. This instance is taking care of payment, user registration and authorization. However application has connection with common Mongo database to read from it,
- **Server instance 1 - calculations** - downloading applications information, user reviews and person data from web. Mission of this instance is also building similarity matrix and generating recommendations. Server has no connection with MySQL database on second server, because it is not necessary.

Both server instances has been configured using *docker-compose* tool which is part of the Docker platform [10].

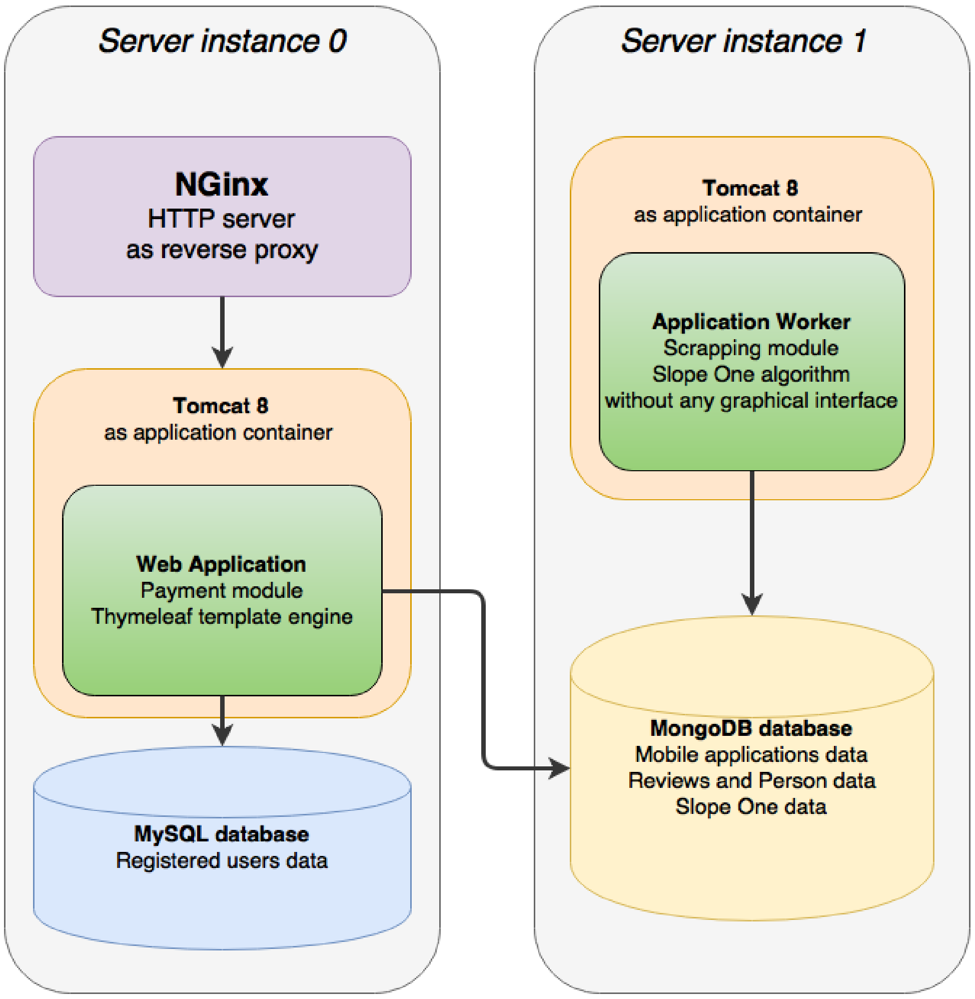


Figure 3.1: Architecture overview

3.2.4. *Slope one* Algorithm

Simplified example data set with ratings from users contains four users and four items. That kind of data are downloaded in the application. *User A* has not expressed preference for two items: *Item C* and *Item D* [11].

Table 3.1: Example user ratings table for applications

	Item A	Item B	Item C	Item D
User A	5.0	2.5	no preference	no preference
User B	5.0	4.0	4.0	2.0
User C	4.5	2.5	1.5	3.5
User D	4.0	4.0	2.0	3.5

To predict the preference for *Item C* and *Item D* for *User A*, a similarity matrix has to be build.

Building similarity matrix

The similarity matrix is built by calculating a difference between every pair of items.

Table 3.2: Difference between Item A and Item B

	Item A	Item B	Difference between *Item A* and *Item B*
User A	5.0	2.5	2.5 = 5.0 - 2.5
User B	5.0	4.0	1.0 = 5.0 - 4.0
User C	4.5	2.5	2.0 = 4.5 - 2.5
User D	4.0	4.0	0.0 = 4.0 - 4.0

An average difference equals $\frac{2.5+4.0+2.5+4.0}{4} = 1.375$. Sum of differences is divided by number of users who expressed the preference for both items. Comparison of the same objects has been omitted. In same way rest of items were calculated.

Table 3.3: Calculated differences between items

	A-B	A-C	A-D	B-C	B-D	C-D
User A	2.5	-	-	-	-	-
User B	1.0	1.0	3.0	0.0	2.0	2.0
User C	2.0	3.0	1.0	1.0	-1.0	-2.0
User D	0.0	2.0	0.5	2.0	0.5	-1.5
Sum	5.5	6.0	4.5	3.0	1.5	-1.5
Average	1.375	2	1.5	1	0.5	-0.5

The completed similarity matrix will look like below. Pairs between the same object were omitted, because there are not providing any useful information. Calculations for inverted items are also omitted, there will be negative values.

Table 3.4: Similarity matrix

	Item A	Item B	Item C	Item D
Item A	-			
Item B	1.375	-		
Item C	2.0	1.0	-	
Item D	1.5	0.5	-0.5	-

Generating recommendations

In this case, the goal is to predict *User's A* ratings for *Item C* and *Item D*, basing on his earlier opinions for *Item A*, *Item B* and values from the similarity matrix. The preferred preference for the item which has no rating is average value of all predicted values based on earlier ratings. *User A* has rated two items - *Item A* and *Item B*.

Calculating a prediction for *Item C*

- Predicted preference based on *Item A* $5.0 + 2.0 = 7.0$
- Predicted preference based on *Item B* $2.5 + 1.0 = 3.5$

The preference for *Item C* equals $\frac{7.0+3.5}{2} = 5.25$

Calculating a prediction for *Item D*

- Predicted preference based on *Item A* $5.0 + 1.5 = 6.5$
- Predicted preference based on *Item B* $2.5 + 0.5 = 3.0$

The Preference for *Item D* equals $\frac{6.5+3.0}{2} = 4, .75$ In this case *Item C* has higher prediction value, that is way this item should be recommended to *User A*.

Chapter 4

Functional requirements

Table 4.1: Functional Requirement 1

Requirement id:	1
Description:	User shall be able to sign up in system using email and password
Rationale:	User need account to see application data and like-minded people
Originator:	Anonymous user
Fit Criterion:	User has account in the application system
Dependencies:	None
History:	Created 27 November 2016

Table 4.2: Functional Requirement 2

Requirement id:	2
Description:	User shall be able to sign up in system through Google Plus account
Rationale:	User need account to see application data and like-minded people
Originator:	Anonymous user
Fit Criterion:	User has account in the application system
Dependencies:	None
History:	Created 27 November 2016

Table 4.3: Functional Requirement 3

Requirement id:	3
Description:	User shall be able to sign in in system using email and password
Rationale:	User need account to see application data and like-minded people
Originator:	User
Fit Criterion:	User is authorized in the application system and can see all data
Dependencies:	None
History:	Created 27 November 2016

Table 4.4: Functional Requirement 4

Requirement id:	4
Description:	User shall be able to sign in system through Google Plus account
Rationale:	User need account to see application data and like-minded people
Originator:	User
Fit Criterion:	User is authorized in the application system and can see all data
Dependencies:	None
History:	Created 27 November 2016

Table 4.5: Functional Requirement 5

Requirement id:	5
Description:	User shall be able to filter applications data by given criteria
Rationale:	User can purchase custom part of data set
Originator:	User
Fit Criterion:	User is able to see applications by given criteria
Dependencies:	User need to be authorized in system
History:	Created 28 November 2016

Table 4.6: Functional Requirement 6

Requirement id:	6
Description:	User shall be able to filter like-minded people by given application
Rationale:	User can purchase custom part of contact information
Originator:	User
Fit Criterion:	User is able to see like-minded people who are interested in given application
Dependencies:	User need to be authorized in the application system
History:	Created 28 November 2016

Table 4.7: Functional Requirement 7

Requirement id:	7
Description:	User shall be able change password
Rationale:	User should has ability to change password if it is necessary
Originator:	User
Fit Criterion:	User need to use new password to sign in
Dependencies:	User need to be authorized in the application system
History:	Created 28 November 2016

Table 4.8: Functional Requirement 8

Requirement id:	8
Description:	User shall be able to remember password
Rationale:	User should has chance to recover account
Originator:	User
Fit Criterion:	User is able to sign in using new password
Dependencies:	User need have account in the application system
History:	Created 28 November 2016

Table 4.9: Functional Requirement 9

Requirement id:	9
Description:	User shall be able to log out from the application system
Rationale:	User should has chance to sign in to another account
Originator:	User
Fit Criterion:	User is able to sign in to another account
Dependencies:	User need to be authorized in the application system
History:	Created 29 November 2016

Table 4.10: Functional Requirement 10

Requirement id:	10
Description:	User shall be able to filter application’s reviews by given application name
Rationale:	User can read reviews which belongs to desired application
Originator:	User
Fit Criterion:	User is able to see people reviews which belongs to desired application
Dependencies:	User need to be authorized in the application system
History:	Created 29 November 2016

Chapter 5

Nonfunctional requirements

5.1. Security Requirements

In an every computer program security should be most important aspect. Users who has trusted application and developer of this program should be well secured. User's data, especially passwords, should be properly protected. Gathered and processed data are also need protection against loss or theft. Types of protection and security has been divided into four groups.

General

An every password should be different and have more than 16 characters. The project has few components which has own authorization system:

- **Tomcat 8.0** - application container
- **MongoDB 3.2 database** - NoSQL database instance
- **MySQL 5.7 database** - relational database instance

Passwords in this modules should be changed every month to prevent not authorized access.

Preventing Data Loss

All applications which depends on gathered data rather than processing volatile content, backups of downloaded resources should be created every 6 hours and send to other server through encrypted connection. In this project all these steps are done by *Bash* script, which deletes backups older than 7 days and creates new one. In case of lack of space email should be sent to a system administrator.

Sensitive Data

The system offers paid content, all credit information is very important data. For security reasons sensitive data are not stored or even processed by application code. PayPal which is professional system will take care of whole payment process.

Secured Connection

A connection between application and client web browser should be encrypted by SSL certificate. To public access only two ports should be exposed:

- **443 port** - used to the secure and encrypted connection by *https* protocol
- **80 port** - should be used to redirect client to encrypted page on port 443

5.2. Hardware Requirements

The application has two main modules: web module with graphical interface for potential client and worker module which is responsible for the data gathering and analysis. To provide good performance and availability those two modules has been separated into two different server instances with different specifications. An important requirement for both servers is the Linux kernel, should be newer than 3.10, this springs from the Docker requirements [10]. Furthermore more disk space for the worker module need to be extendable, to keep more data in future. Below has been presented recommended server requirements for both modules.

Table 5.1: Recommended hardware requirements

	Web module	Worker module
Architecture	64 bit	64 bit
CPU	1 core	2 core
RAM	1 GB	8 GB
System	Linux (kernel >= 3.10)	Linux (kernel >= 3.10)
Free disk space	6 GB	20 GB

5.3. Aesthetic Requirements

The web application in this project should be an abstraction layer on more complicated processes which are done under hood. Whole graphical interface should be simple and clear as it can. Potential client should be able to find out main features of the application in seconds. All forms in the application should provide placeholders as hints for client. Generated recommendations need to be presented in simple way with percentage recommendation rate. The application should provide statistics using charts and tables.

5.4. Scalability and Performance Requirements

Performance of the application mostly depend on server where MongoDB database has been placed. But to provide high scalability many solutions has been applied:

- Design patterns - the most powerful solution to provide easy to scale is using a design patterns in proper way like in this project has been made. Almost every component used design pattern to increase code readability and performance.
- Tool to manage modules - main tool to manage modules and configurations was Docker. Due to this solution, any module with configuration can be easily transferred into other server instance in few minutes. Docker-compose, which is part of Docker platform supports scaling by *scale* parameter, which could be useful in future.
- Stateless application - stateless application means that application does not save any data internally, every request is treated in the same way. Stateless program gives opportunity to launch more instances of this application and use load-balancer system.
- Proper server provider - last but not least solution to provide good performance is choosing proper server provider. Main motive in choosing good provider has been easiness of upgrading server instance.

Chapter 6

Project

6.1. Off-the-shelf Software

The modern programming is building software from blocks which has been written by other developers, that is why many third-party software has been used in this project. Most important software which has been bought is Twitter Bootstrap template called *Material Admin* [12] created by Bootstrap Sale company. To authorization Google Plus service [13] has been used. Social login are very comfortable authorization mechanisms, that is way this project is taking advantage of this solution. Payment system were provided by PayPal company, that also provides Java module as Maven dependency [14]. Docker for server configuration has been used [10] and Tomcat as application container for web applications [15].

6.2. Application modules

Project has been divided into four application modules.

6.2.1. Web module

Web module relies mostly on Spring Framework and Thymeleaf as template engine. Takes care of user experience and view.

View implementation

Thymeleaf is template engine for server-side Java applications. Graphical interface in project was build in HTML and CSS, Thymeleaf injects into HTML code model objects which has been added as attribute in application side.

Listing 6.1: Templating example with Thymeleaf - simple-filter.html

```
<tr th:each="app␣:␣${allApps}">
        <td>
        <img th:src="${app.iconUrl}"/>
        </td>
        <td th:text="${app.companyName}"></td>
        <td th:text="${app.rating}"></td>
        <td th:text="${app.category}"></td>
</tr>
```

Thymeleaf will iterate over *allApps* which is *java.util.List* object from Java. For every *app* from list it will be created <*tr*> tag with application info.

Listing 6.2: Attribute injection example - FilterController.java

```
@RequestMapping(value = "/filter")
public String simpleFilter(Model model) {
        List<AppEntity> allApps
                = appsService.findAll(0, 10); //fetch from database
        model.addAttribute("allApps", allApps); // inject to model
        return "user/simple-filter"; // return HTML page
}
```

Hit points system

Every action in application is paid. Instead of paying for each single action project implements own currency system called hit points. Every user entity has own hit points counter stored in MySQL database. User needs to be registered and signed in system. Every time user send request to acquire filtered applications page or audience recommendations, system checks is user can afford that. If user has enough points to buy requested page, points are subtracted from account and requested content is returned. In case of lack of points, user is redirected to pricing table. Hit points can be bought using payment module which is described below. On every page hit points are displayed in top right corner. To achieve that effect *HitsInterceptor* were used which is an implementation of Spring *HandlerInterceptor* interface. Interceptor is registered on */user/*** path in web application *ServletConfig*. For any request which matches to */user/*** path pattern HitsInterceptor updates hit points amount in *HttpSession* object.

Listing 6.3: HitsInterceptor.java

```
@Override
public void afterCompletion(
            HttpServletRequest httpServletRequest,
            HttpServletResponse httpServletResponse,
            Object o,
            Exception e) throws Exception {
    Long userHits = userService
                            .getLoggedUser() //get logged user
                            .getHits(); //get user hit points

    httpServletRequest
            .getSession() //get session object from request
            .setAttribute("hits", userHits); // set attribute "hits"
}
```

Thymeleaf accesses session data in HTML view by session object *${session.hits}*.

Authorization mechanism

Application allows to registration by email or using external authorization system with OAuth 2.0 protocol like Google Plus. In authorization part many design patterns were used. One of the most important are decorator, factory and flyweight [16]. That set of programming solutions allows to build very scalable and light application. In case of changing or adding new authorization system it can be done with ease.

6.2.2. Payment module

All data which are gathered and processed in databases are very high-value content. In the course of time maintaining system will be more and more expensive. In order to ensure high system availability and continuity of the work payments are necessary. Every access to application data, recommendations or contact info to person is paid. Paying every time that client wants to display data could be awkward and annoying. Application has own currency system which are simple points that for which data can be bought. There is a possibility to buy three different packs vary in points amount and price. To buy points pack client has to be signed in, because points are associated with account. Payments are processed by PayPal company which provides Java plugin to easy implement payments [17]. In order to make payment user is redirected with necessary parameters to PayPal web application, where payment process occurs. After purchase, PayPal redirects user back to web-application with payment status. If everything worked well, system adds hit points to user's account. In other way error message is displayed.

6.2.3. Scrapping data module

Scrapping module is a Spring based worker without any graphical interface. Main task of this module is gathering data from Apple Store, Google Play and persist it in MongoDB database. There are supported three kinds of data:

Downloaded content kind

- Application data (Google Play and Apple Store) - for showing detailed information about reviewed application. It also gives a chance to offer application filtering and sorting by custom fields.
- Person's review (Google Play) - used as analysis data by *slope one* algorithm and generate application recommendations.
- Person data (Google Play)- for expose contact information for potential clients who will use this application.

To download content from website Jsoup were used as main HTML parser[18], and Jackson for casting downloaded data to Java objects [19].

Extracting data

Every time application downloads data document, it extracts application information. After that links to other applications were extracted from the same document. That approach allows to creating download application queue, minimize network usage and resource usage. If document is downloaded from Google Play it also extracts reviews and people data.

Work distribution

After start due to *@PostConstruct* annotation, worker starts four threads from *java.lang.Thread* package, using *java.util.concurrent.ExecutorService* and Java 8 lambdas. That approach is taking advantage of multi-core CPU installed in server, and accelerates tasks which can be done in concurrently [20]. Blockage is MongoDB database which uses many CPU resources while application write or read from it.

6.2.4. Algorithm module

The biggest advantage of chosen algorithm is simplicity and high effectiveness. Gathered data by scrapper module were nessesary to work algorithm.

Slope one implementation

Implementation were divided in two submodules: *PredictorService* and *SlopeOne*. This represents two steps of main slope one division. *SlopeOne* is responsible for building similarity matrix and *PredictorService* is used for generating recommendations. That split allows to work both modules simultaneously.

Building matrix

Database contains reviews with all nessesary information to build similarity matrix. Due to Mongo which is document database each review has application and user who has reviews of other applications. Main idea of implemented algorithm has been presented below.

Algorithm 1 Build difference matrix algorithm

```
 1: function buildMatrixForApplication(Application app)
 2:     R1 ← findReviewsFor app
 3:     R2 ← findAllReviewsInDatabase
 4:     for every review R1 do
 5:         for every review R2 do                          ▷ Comparing every review each other
 6:             if r1 == r1 then                            ▷ Omit comparing same review
 7:                 continue;
 8:             a1 ← get application from r1
 9:             a2 ← get application from r2
10:             U ← findUsersWhoReviewedApplications a1, a2
11:             sum, count ← set both to 0
12:             for every user U do                         ▷ Calculate average difference between reviews
13:                 if u rated r1 and r2 then
14:                     sum ← add difference between r1, r2
15:             similarityRate ← divide sum by count
16:             saveToDatabase similarityRate               ▷ Save result to database
```

Finding users in database who contains reviews for both applications (line 8) is the most time consuming task in this algorithm.

Generating recommendations

Due preprocessing part, generating recommendations is a very fast process.

Algorithm 2 Recommendation algorithm

```
 1: function predict(User u)
 2:     R1 ← get applications who were not reviewed by u
 3:     R2 ← get applications who were reviewed by u
 4:     for every review R1 do
 5:         count, sum ← sets both to 0
 6:         for every review R2 do
 7:             similarityRate ← getSimilarityRate r1, r2       ▷ From database
 8:             reviewRate ← getReviewRate u
 9:             sum += reviewRate + similarityRate
10:             count++
11:         predictionRate ← divide sum by count
12:         saveToDatabase predictionRate                       ▷ Save result to database
```

PredictionRate and *User u* are used to create *Recommendation* object and persisted in database to easily acquire and sort this data later.

6.3. Database

Operating on data is one of the biggest parts in web applications. Persisting and acquiring data are very important aspects in every application. This web application uses **Spring Data** which is on of elementary sub-project from Spring Framework. It provides very easy layer which helps to access data on any supported database engine. Moreover projects gets advantage of **QueryDSL** framework which provides high level abstraction of manipulating data by generating helper classes. QueryDSL generates helper class for every model class in project. Model classes are annotated by *@Entity* annotations from *javax.annotations* package.

6.3.1. MySQL database

Users information are very intense data, that is why there were stored in MySQL database. Keeping that kind of data in relational databases have many advantages like transnational and atomic operations. This properties are very useful when sensitive data need to be persisted.

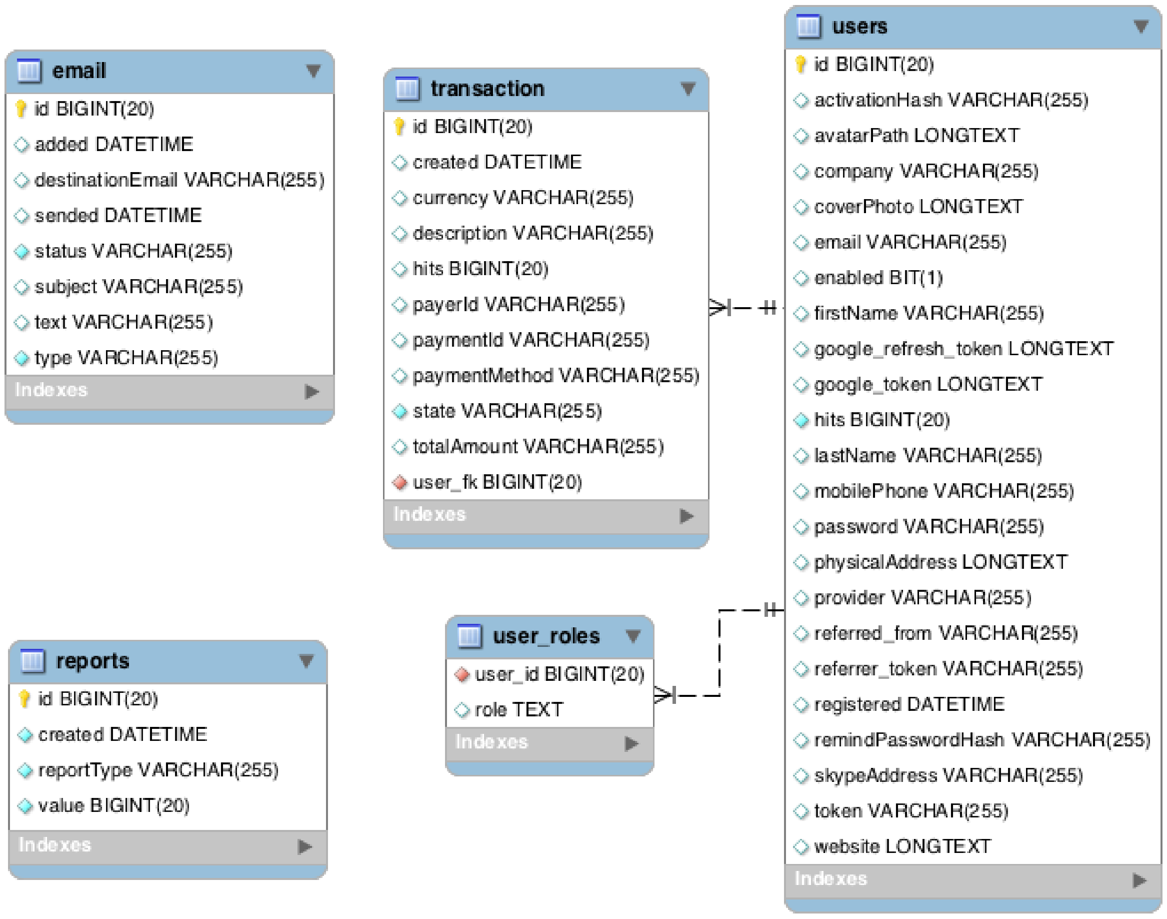
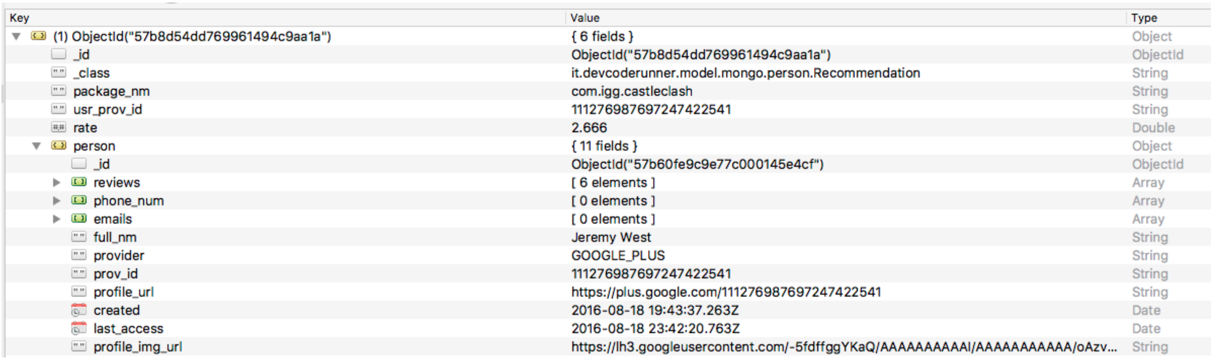


Figure 6.1: MySQL database - ERD diagram

- **Email** table - contains email queue, there are all sent emails and emails which should be send in the feature.
- **Reports** table - table used to store all historical data which allows to create statistics and charts on dashboard page. There is creation date, report type and value which should be saved.
- **Transaction** table - persists user transactions history, every transaction is tracked in case of failure at any step. It helps to check if user was able to complete the purchase.
- **User** table - all registered users were stored in this table. User passwords are stored in hashed form. Every password while register is hashed by MD5 algorithm which is included in Spring Security project.

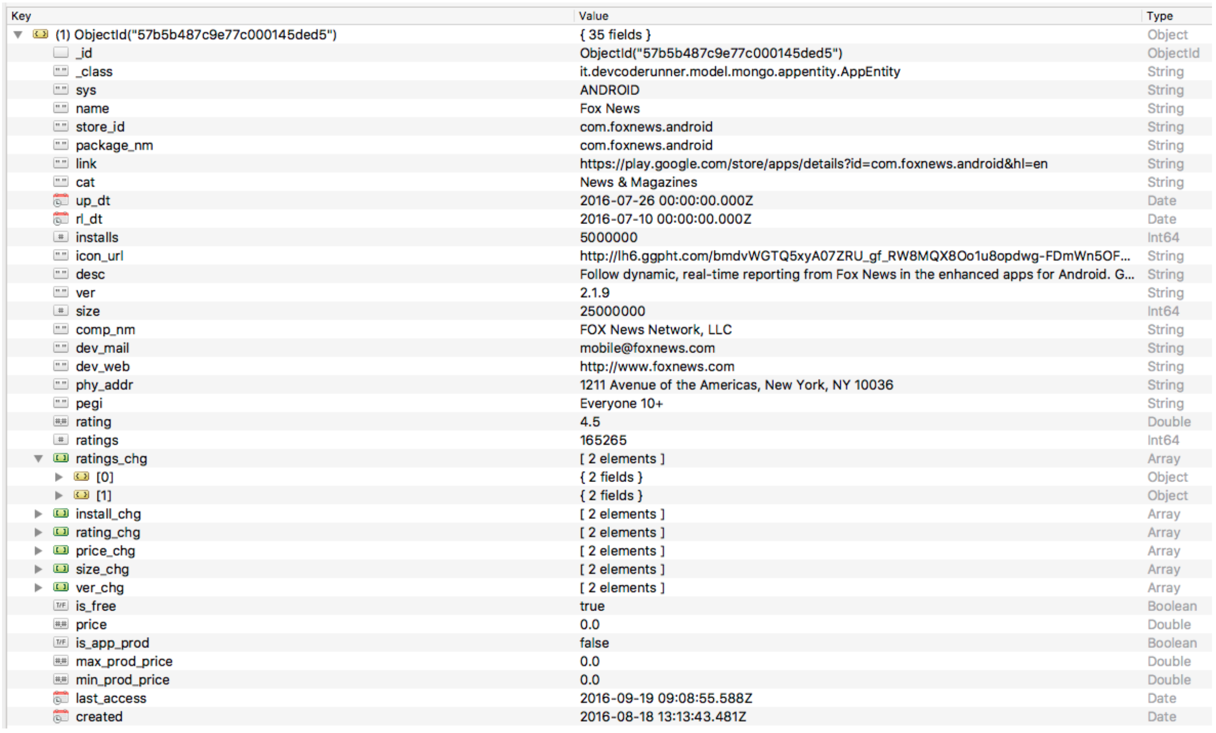
6.3.2. MongoDB database

All data which was downloaded from web are stored in MongoDB which is NoSQL database engine. One of the main argument for choosing that kind of database was high write performance and horizontal scaling. To improve performance in case of slowdown, there can be added an extra node with MongoDB instance. This solution is easier and cheaper to do than providing extra power to existing server-node.



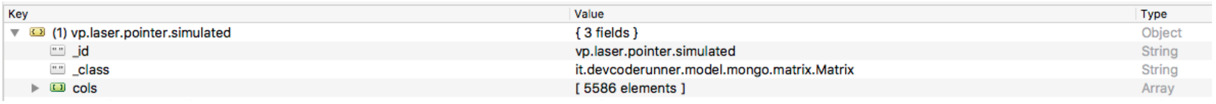
Key	Value	Type
(1) ObjectId("57b8d54dd769961494c9aa1a")	{ 6 fields }	Object
_id	ObjectId("57b8d54dd769961494c9aa1a")	ObjectId
_class	it.devcoderunner.model.mongo.person.Recommendation	String
package_nm	com.igg.castleclash	String
usr_prov_id	111276987697247422541	String
rate	2.666	Double
person	{ 11 fields }	Object
_id	ObjectId("57b60fe9c9e77c000145e4cf")	ObjectId
reviews	[6 elements]	Array
phone_num	[0 elements]	Array
emails	[0 elements]	Array
full_nm	Jeremy West	String
provider	GOOGLE_PLUS	String
prov_id	111276987697247422541	String
profile_url	https://plus.google.com/111276987697247422541	String
created	2016-08-18 19:43:37.263Z	Date
last_access	2016-08-18 23:42:20.763Z	Date
profile_img_url	https://lh3.googleusercontent.com/-5fdffggYKaQ/AAAAAAAAAAI/AAAAAAAAAAA/oAzv...	String

Figure 6.2: Single recommendation schema in Mongo database



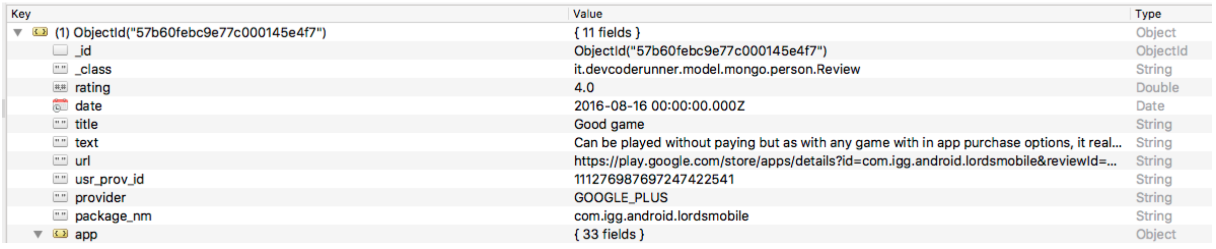
Key	Value	Type
(1) ObjectId("57b5b487c9e77c000145ded5")	{ 35 fields }	Object
_id	ObjectId("57b5b487c9e77c000145ded5")	ObjectId
_class	it.devcoderunner.model.mongo.appentity.AppEntity	String
sys	ANDROID	String
name	Fox News	String
store_id	com.foxnews.android	String
package_nm	com.foxnews.android	String
link	https://play.google.com/store/apps/details?id=com.foxnews.android&hl=en	String
cat	News & Magazines	String
up_dt	2016-07-26 00:00:00.000Z	Date
rl_dt	2016-07-10 00:00:00.000Z	Date
installs	5000000	Int64
icon_url	http://lh6.ggpht.com/bmdvWGTQ5xyA07ZRU_gf_RW8MQX8Oo1u8opdwg-FDmWn5OF...	String
desc	Follow dynamic, real-time reporting from Fox News in the enhanced apps for Android. G...	String
ver	2.1.9	String
size	25000000	Int64
comp_nm	FOX News Network, LLC	String
dev_mail	mobile@foxnews.com	String
dev_web	http://www.foxnews.com	String
phy_addr	1211 Avenue of the Americas, New York, NY 10036	String
pegi	Everyone 10+	String
rating	4.5	Double
ratings	165265	Int64
ratings_chg	[2 elements]	Array
[0]	{ 2 fields }	Object
[1]	{ 2 fields }	Object
install_chg	[2 elements]	Array
rating_chg	[2 elements]	Array
price_chg	[2 elements]	Array
size_chg	[2 elements]	Array
ver_chg	[2 elements]	Array
is_free	true	Boolean
price	0.0	Double
is_app_prod	false	Boolean
max_prod_price	0.0	Double
min_prod_price	0.0	Double
last_access	2016-09-19 09:08:55.588Z	Date
created	2016-08-18 13:13:43.481Z	Date

Figure 6.3: Single application schema in Mongo database



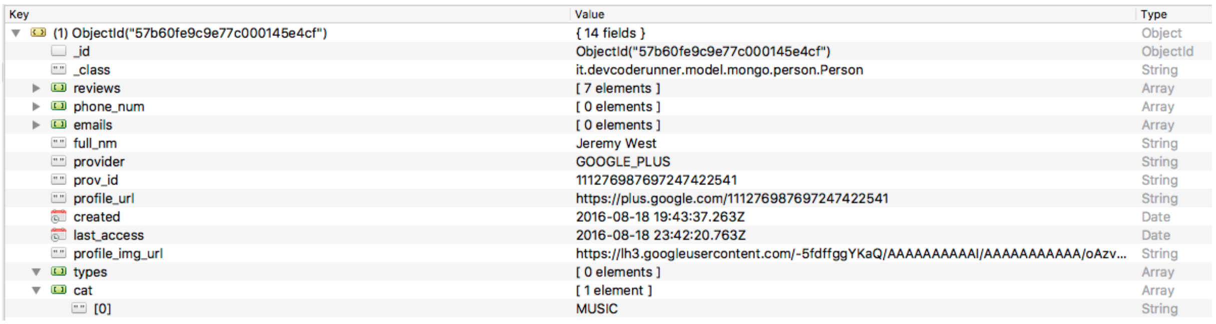
Key	Value	Type
(1) vp.laser.pointer.simulated	{ 3 fields }	Object
_id	vp.laser.pointer.simulated	String
_class	it.devcoderunner.model.mongo.matrix.Matrix	String
cols	[5586 elements]	Array

Figure 6.4: Matrix row schema in Mongo database



Key	Value	Type
(1) ObjectId("57b60febc9e77c000145e4f7")	{ 11 fields }	Object
_id	ObjectId("57b60febc9e77c000145e4f7")	ObjectId
_class	it.devcoderunner.model.mongo.person.Review	String
rating	4.0	Double
date	2016-08-16 00:00:00.000Z	Date
title	Good game	String
text	Can be played without paying but as with any game with in app purchase options, it real...	String
url	https://play.google.com/store/apps/details?id=com.igg.android.lordsmobile&reviewId=...	String
usr_prov_id	111276987697247422541	String
provider	GOOGLE_PLUS	String
package_nm	com.igg.android.lordsmobile	String
app	{ 33 fields }	Object

Figure 6.5: Single review schema in Mongo database

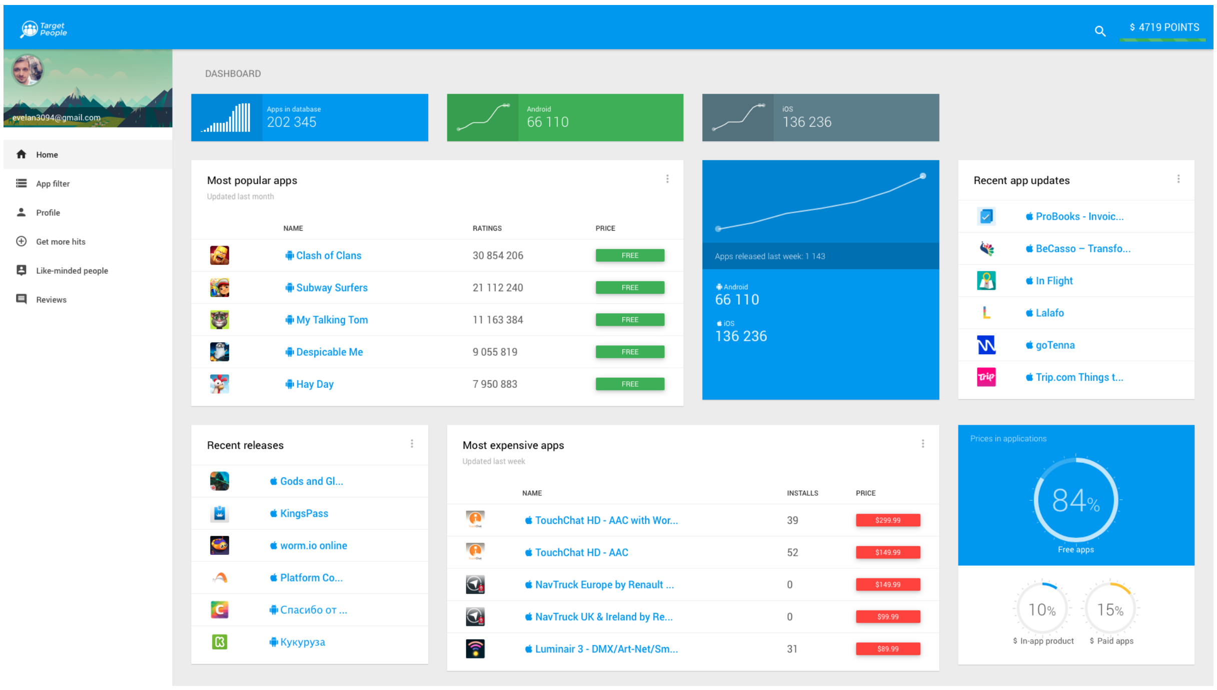


Key	Value	Type
(1) ObjectId("57b60fe9c9e77c000145e4cf")	{ 14 fields }	Object
_id	ObjectId("57b60fe9c9e77c000145e4cf")	ObjectId
_class	it.devcoderunner.model.mongo.person.Person	String
reviews	[7 elements]	Array
phone_num	[0 elements]	Array
emails	[0 elements]	Array
full_nm	Jeremy West	String
provider	GOOGLE_PLUS	String
prov_id	111276987697247422541	String
profile_url	https://plus.google.com/111276987697247422541	String
created	2016-08-18 19:43:37.263Z	Date
last_access	2016-08-18 23:42:20.763Z	Date
profile_img_url	https://lh3.googleusercontent.com/-5fdffggYKaQ/AAAAAAAAAAI/AAAAAAAAAAA/oAzv...	String
types	[0 elements]	Array
cat	[1 element]	Array
[0]	MUSIC	String

Figure 6.6: Person schema in Mongo database

6.4. UI/UX

Important assumption is simplicity of user expirience. In order to maximize easy of use, Bootstrap Framework has been used.



Apps in database 202 345

Android 66 110

iOS 136 236

Most popular apps
Updated last month

	NAME	RATINGS	PRICE
	Clash of Clans	30 854 206	FREE
	Subway Surfers	21 112 240	FREE
	My Talking Tom	11 163 384	FREE
	Despicable Me	9 055 819	FREE
	Hay Day	7 950 883	FREE

Apps released last week: 1 143

Android 66 110

iOS 136 236

Recent app updates

- ProBooks - Invoic...
- BeCasso – Transfo...
- In Flight
- Lalafo
- goTenna
- Trip.com Things t...

Recent releases

- Gods and Gl...
- KingsPass
- worm.io online
- Platform Co...
- Спасибо от ...
- Кукуруза

Most expensive apps
Updated last week

	NAME	INSTALLS	PRICE
	TouchChat HD - AAC with Wor...	39	$299.99
	TouchChat HD - AAC	52	$149.99
	NavTruck Europe by Renault ...	0	$149.99
	NavTruck UK & Ireland by Re...	0	$99.99
	Luminair 3 - DMX/Art-Net/Sm...	31	$89.99

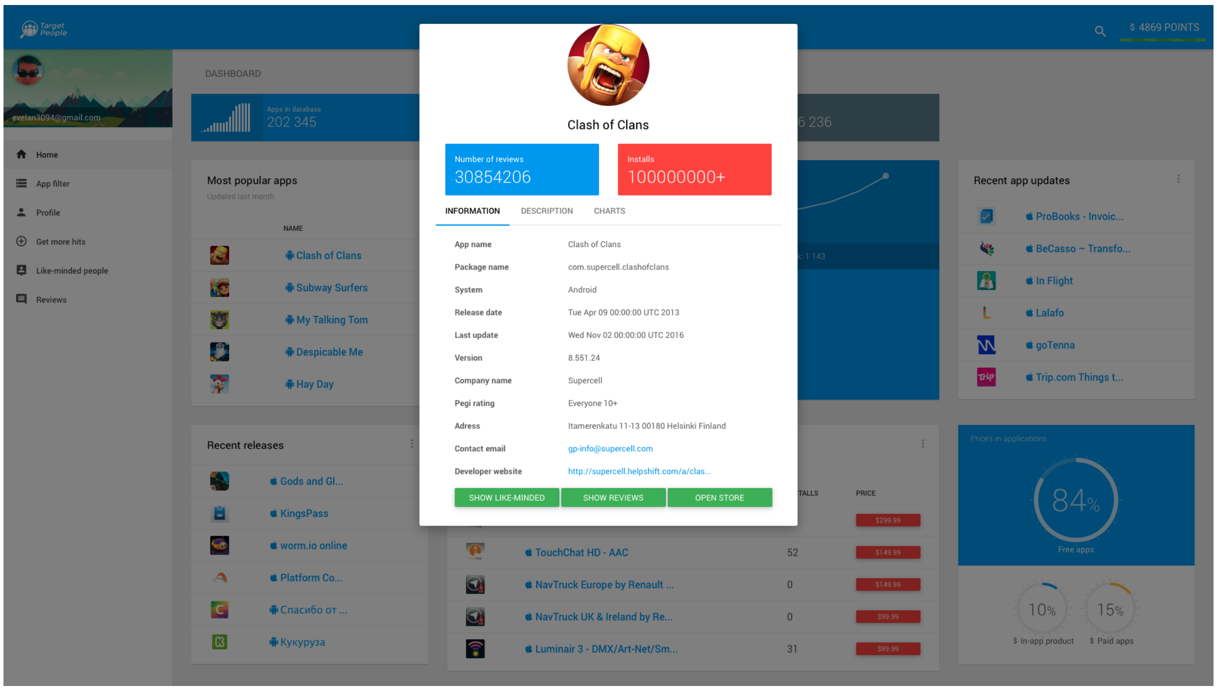
Prices in applications

84% Free apps

10% $ In-app product

15% $ Paid apps

Figure 6.7: Application homepage



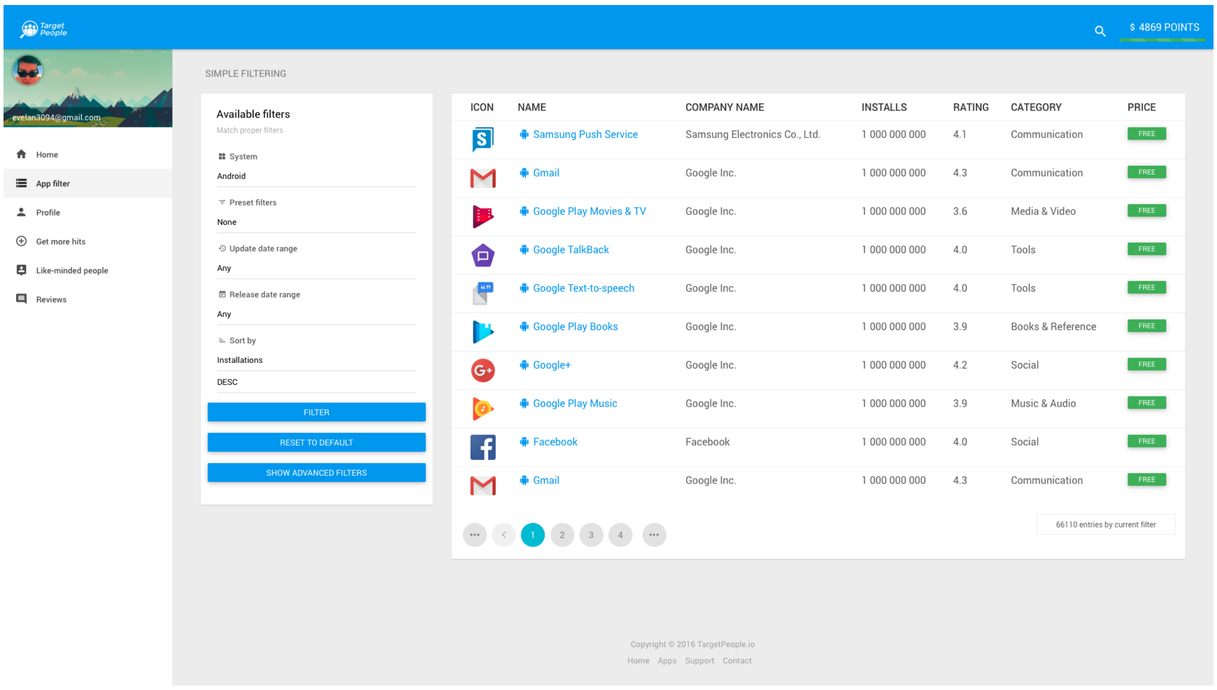
Clash of Clans

Number of reviews 30854206

Installs 100000000+

App name	Clash of Clans
Package name	com.supercell.clashofclans
System	Android
Release date	Tue Apr 09 00:00:00 UTC 2013
Last update	Wed Nov 02 00:00:00 UTC 2016
Version	8.551.24
Company name	Supercell
Pegi rating	Everyone 10+
Adress	Itamerenkatu 11-13 00180 Helsinki Finland
Contact email	gp-info@supercell.com
Developer website	http://supercell.helpshift.com/a/clas...

Figure 6.8: Single view of mobile application from database



SIMPLE FILTERING

Available filters
Match proper filters

System
Android

Preset filters
None

Update date range
Any

Release date range
Any

Sort by
Installations
DESC

FILTER
RESET TO DEFAULT
SHOW ADVANCED FILTERS

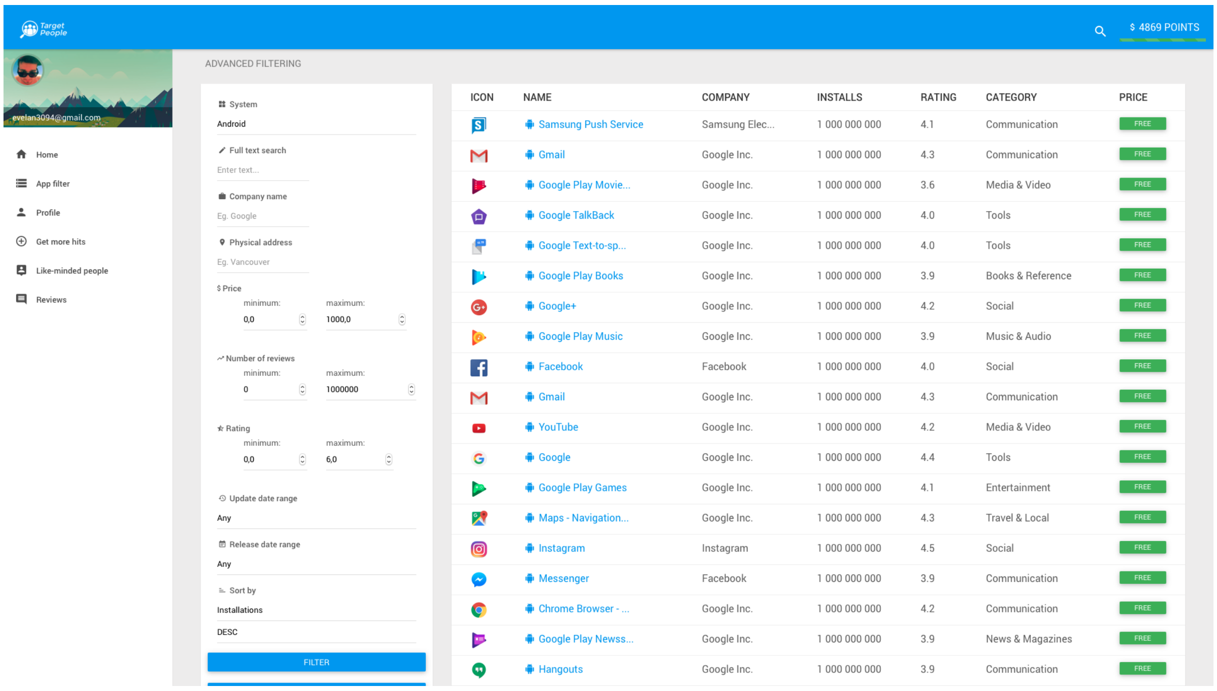
ICON	NAME	COMPANY NAME	INSTALLS	RATING	CATEGORY	PRICE
	Samsung Push Service	Samsung Electronics Co., Ltd.	1 000 000 000	4.1	Communication	FREE
	Gmail	Google Inc.	1 000 000 000	4.3	Communication	FREE
	Google Play Movies & TV	Google Inc.	1 000 000 000	3.6	Media & Video	FREE
	Google TalkBack	Google Inc.	1 000 000 000	4.0	Tools	FREE
	Google Text-to-speech	Google Inc.	1 000 000 000	4.0	Tools	FREE
	Google Play Books	Google Inc.	1 000 000 000	3.9	Books & Reference	FREE
	Google+	Google Inc.	1 000 000 000	4.2	Social	FREE
	Google Play Music	Google Inc.	1 000 000 000	3.9	Music & Audio	FREE
	Facebook	Facebook	1 000 000 000	4.0	Social	FREE
	Gmail	Google Inc.	1 000 000 000	4.3	Communication	FREE

··· < 1 2 3 4 ···

66110 entries by current filter

Copyright © 2016 TargetPeople.io
Home Apps Support Contact

Figure 6.9: Simple filters with predefined settings



ADVANCED FILTERING

System
Android

Full text search
Enter text...

Company name
Eg. Google

Physical address
Eg. Vancouver

$ Price
minimum: 0,0
maximum: 1000,0

Number of reviews
minimum: 0
maximum: 1000000

Rating
minimum: 0,0
maximum: 6,0

Update date range
Any

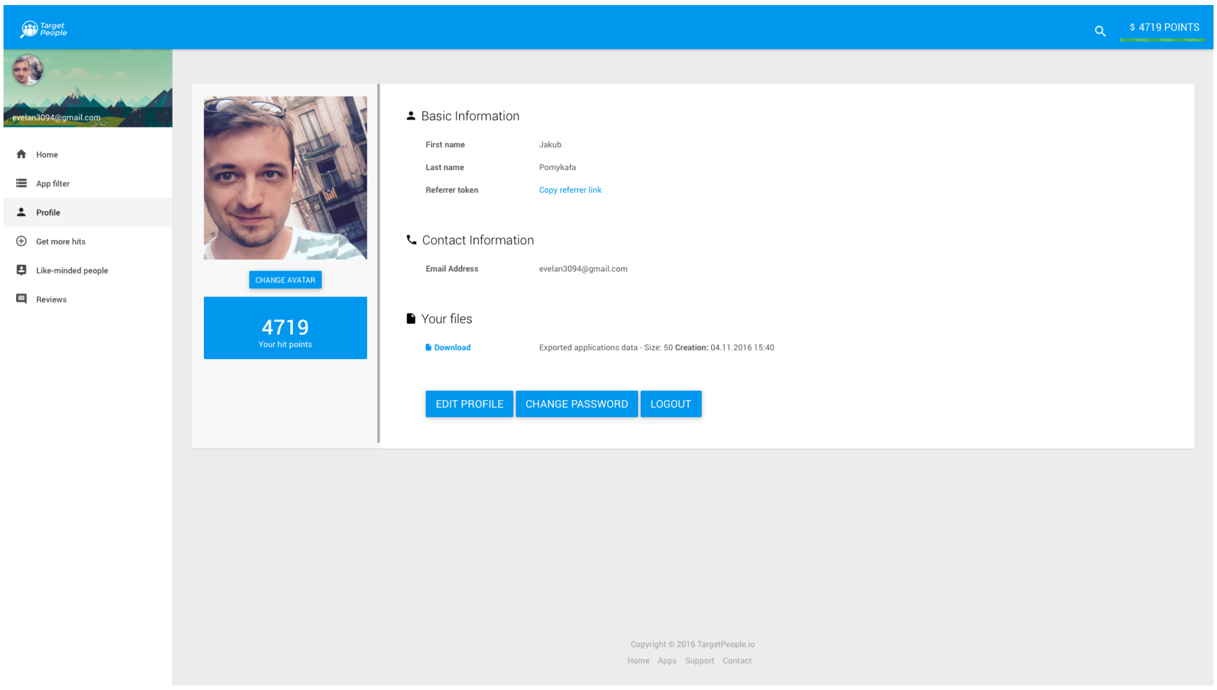
Release date range
Any

Sort by
Installations
DESC

FILTER

ICON	NAME	COMPANY	INSTALLS	RATING	CATEGORY	PRICE
	Samsung Push Service	Samsung Elec...	1 000 000 000	4.1	Communication	FREE
	Gmail	Google Inc.	1 000 000 000	4.3	Communication	FREE
	Google Play Movie...	Google Inc.	1 000 000 000	3.6	Media & Video	FREE
	Google TalkBack	Google Inc.	1 000 000 000	4.0	Tools	FREE
	Google Text-to-sp...	Google Inc.	1 000 000 000	4.0	Tools	FREE
	Google Play Books	Google Inc.	1 000 000 000	3.9	Books & Reference	FREE
	Google+	Google Inc.	1 000 000 000	4.2	Social	FREE
	Google Play Music	Google Inc.	1 000 000 000	3.9	Music & Audio	FREE
	Facebook	Facebook	1 000 000 000	4.0	Social	FREE
	Gmail	Google Inc.	1 000 000 000	4.3	Communication	FREE
	YouTube	Google Inc.	1 000 000 000	4.2	Media & Video	FREE
	Google	Google Inc.	1 000 000 000	4.4	Tools	FREE
	Google Play Games	Google Inc.	1 000 000 000	4.1	Entertainment	FREE
	Maps - Navigation...	Google Inc.	1 000 000 000	4.3	Travel & Local	FREE
	Instagram	Instagram	1 000 000 000	4.5	Social	FREE
	Messenger	Facebook	1 000 000 000	3.9	Communication	FREE
	Chrome Browser - ...	Google Inc.	1 000 000 000	4.2	Communication	FREE
	Google Play Newss...	Google Inc.	1 000 000 000	3.9	News & Magazines	FREE
	Hangouts	Google Inc.	1 000 000 000	3.9	Communication	FREE

Figure 6.10: Advanced filters with all features



4719
Your hit points

Basic Information

First name	Jakub
Last name	Pomykała
Referrer token	Copy referrer link

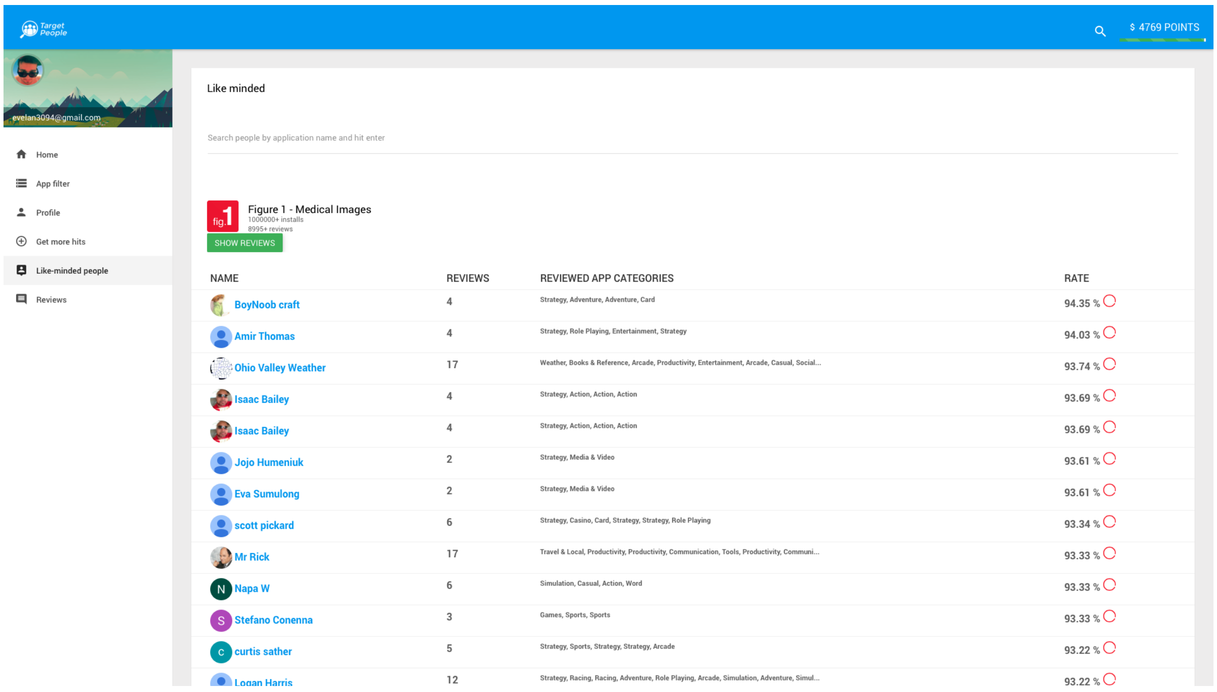
Contact Information

Email Address	evelan3094@gmail.com

Your files

Download	Exported applications data - Size: 50 **Creation:** 04.11.2016 15:40

Figure 6.11: Registered user profile



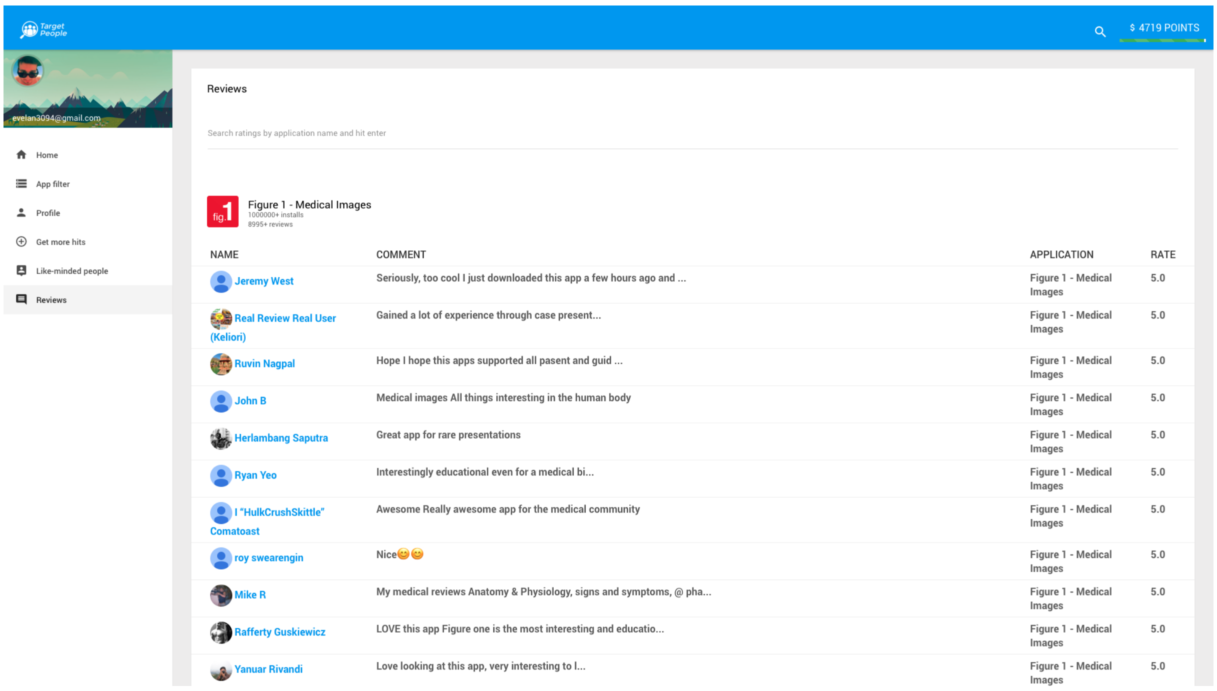
Like minded

Search people by application name and hit enter

Figure 1 - Medical Images
1000000+ installs
8995+ reviews

NAME	REVIEWS	REVIEWED APP CATEGORIES	RATE
BoyNoob craft	4	Strategy, Adventure, Adventure, Card	94.35 %
Amir Thomas	4	Strategy, Role Playing, Entertainment, Strategy	94.03 %
Ohio Valley Weather	17	Weather, Books & Reference, Arcade, Productivity, Entertainment, Arcade, Casual, Social...	93.74 %
Isaac Bailey	4	Strategy, Action, Action, Action	93.69 %
Isaac Bailey	4	Strategy, Action, Action, Action	93.69 %
Jojo Humeniuk	2	Strategy, Media & Video	93.61 %
Eva Sumulong	2	Strategy, Media & Video	93.61 %
scott pickard	6	Strategy, Casino, Card, Strategy, Strategy, Role Playing	93.34 %
Mr Rick	17	Travel & Local, Productivity, Productivity, Communication, Tools, Productivity, Communi...	93.33 %
Napa W	6	Simulation, Casual, Action, Word	93.33 %
Stefano Conenna	3	Games, Sports, Sports	93.33 %
curtis sather	5	Strategy, Sports, Strategy, Strategy, Arcade	93.22 %
Logan Harris	12	Strategy, Racing, Racing, Adventure, Role Playing, Arcade, Simulation, Adventure, Simul...	93.22 %

Figure 6.12: Like-minded people view



Reviews

Search ratings by application name and hit enter

Figure 1 - Medical Images
1000000+ installs
8995+ reviews

NAME	COMMENT	APPLICATION	RATE
Jeremy West	Seriously, too cool I just downloaded this app a few hours ago and ...	Figure 1 - Medical Images	5.0
Real Review Real User (Keliori)	Gained a lot of experience through case present...	Figure 1 - Medical Images	5.0
Ruvin Nagpal	Hope I hope this apps supported all pasent and guid ...	Figure 1 - Medical Images	5.0
John B	Medical images All things interesting in the human body	Figure 1 - Medical Images	5.0
Herlambang Saputra	Great app for rare presentations	Figure 1 - Medical Images	5.0
Ryan Yeo	Interestingly educational even for a medical bi...	Figure 1 - Medical Images	5.0
I "HulkCrushSkittle" Comatoast	Awesome Really awesome app for the medical community	Figure 1 - Medical Images	5.0
roy swearengin	Nice😊😊	Figure 1 - Medical Images	5.0
Mike R	My medical reviews Anatomy & Physiology, signs and symptoms, @ pha...	Figure 1 - Medical Images	5.0
Rafferty Guskiewicz	LOVE this app Figure one is the most interesting and educatio...	Figure 1 - Medical Images	5.0
Yanuar Rivandi	Love looking at this app, very interesting to l...	Figure 1 - Medical Images	5.0

Figure 6.13: Mobile application reviews

6.5. User guide

User guide assumes that application has been deployed on local machine with default project settings.

6.5.1. How to login or register using Google Plus account

1. Open http://localhost:9000/login in web browser.

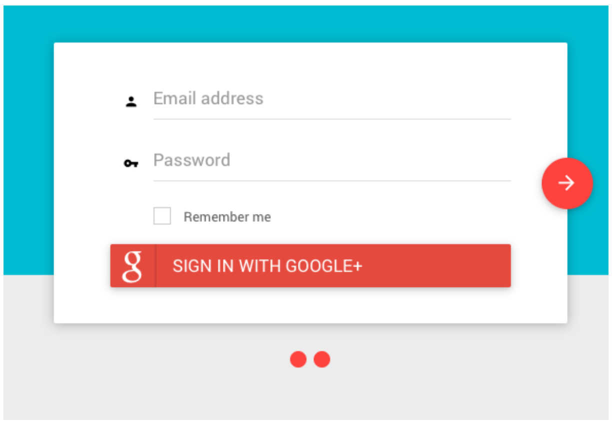


Figure 6.14: Application Login Page

2. Choose "Sign in with Google+" button, application will redirect user to Google authorization service. 6.14
3. Login to Google account. 6.15

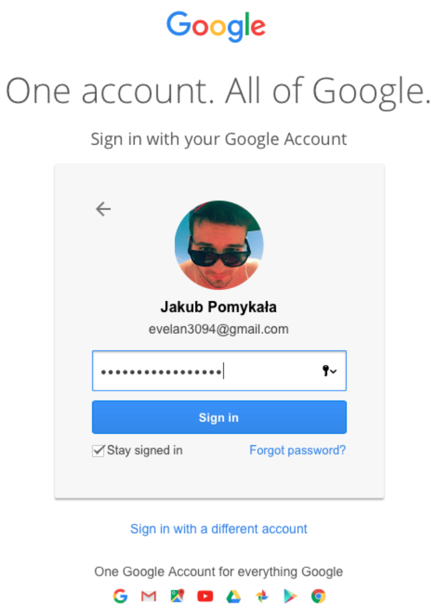

Google
One account. All of Google.
Sign in with your Google Account
Jakub Pomykała
evelan3094@gmail.com
Sign in
Stay signed in
Forgot password?
Sign in with a different account
One Google Account for everything Google

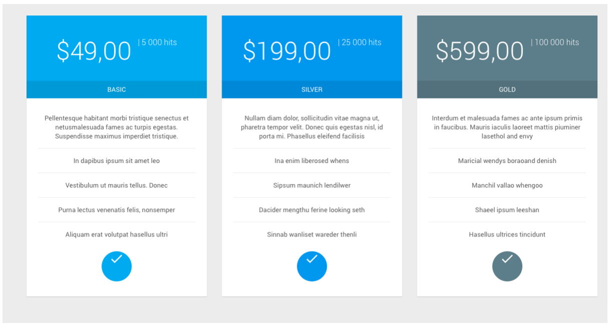
Figure 6.15: Google authorization page

6.5.2. How to purchase hit points

To purchase points, user has to be signed in application.

1. Open http://localhost:9000/user in web browser.
2. Choose "Get more points" from menu on left side.

3. Choose one of pricing plans. 6.16
4. Process payment on PayPal page. 6.17

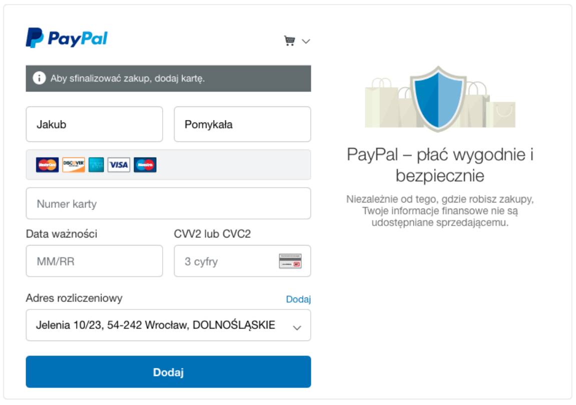


$49,00 | 5 000 hits
BASIC
Pellentesque habitant morbi tristique senectus et netusmalesuada fames ac turpis egestas. Suspendisse maximus imperdiet tristique.
In dapibus ipsum sit amet leo
Vestibulum ut mauris tellus. Donec
Purna lectus venenatis felis, nonsemper
Aliquam erat volutpat hasellus ultri

$199,00 | 25 000 hits
SILVER
Nullam diam dolor, sollicitudin vitae magna ut, pharetra tempor velit. Donec quis egestas nisl, id porta mi. Phasellus eleifend facilisis
Ina enim liberosed whens
Sipsum maunich lendilwer
Dacider mengthu ferine looking seth
Sinnab wanliset wareder thenli

$599,00 | 100 000 hits
GOLD
Interdum et malesuada fames ac ante ipsum primis in faucibus. Mauris iaculis laoreet mattis piuminer lasethol and envy
Maricial wendys boraoand denish
Manchil vallao whengoo
Shaeel ipsum leeshan
Hasellus ultrices tincidunt

Figure 6.16: Pricing page



Aby sfinalizować zakup, dodaj kartę.

Jakub | Pomykała

Numer karty

Data ważności: MM/RR

CVV2 lub CVC2: 3 cyfry

Adres rozliczeniowy — Dodaj

Jelenia 10/23, 54-242 Wrocław, DOLNOŚLĄSKIE

Dodaj

PayPal – płać wygodnie i bezpiecznie

Niezależnie od tego, gdzie robisz zakupy, Twoje informacje finansowe nie są udostępniane sprzedającemu.

Figure 6.17: Purchase process on PayPal page

5. On successful result, user will be redirected to application and hit points will be added.

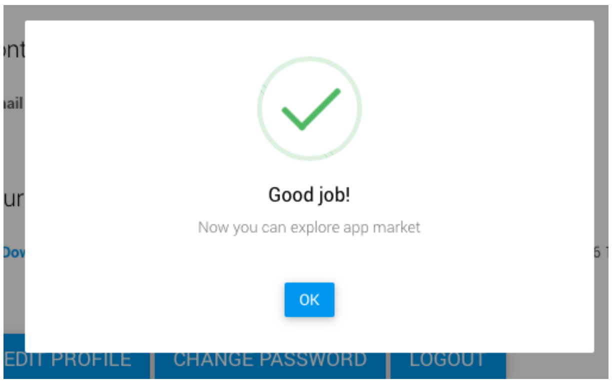
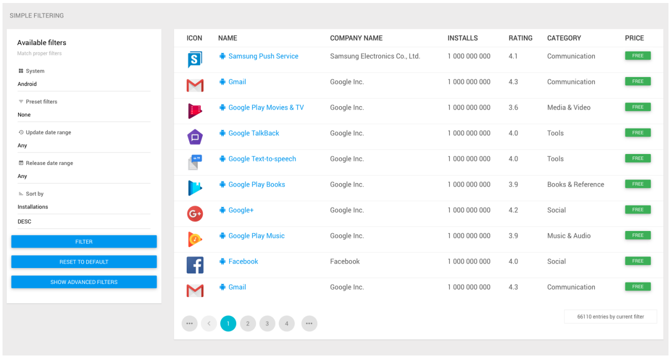


Figure 6.18: Purchase status

6.5.3. How to find application with given criteria

1. Open http://localhost:9000/user in web browser.
2. Choose "Apps" from menu on left side.
3. Choose "Advanced filter" on bottom-left side 6.19



SIMPLE FILTERING

Available filters
Match proper filters

System
Android

Preset filters
None

Update date range
Any

Release date range
Any

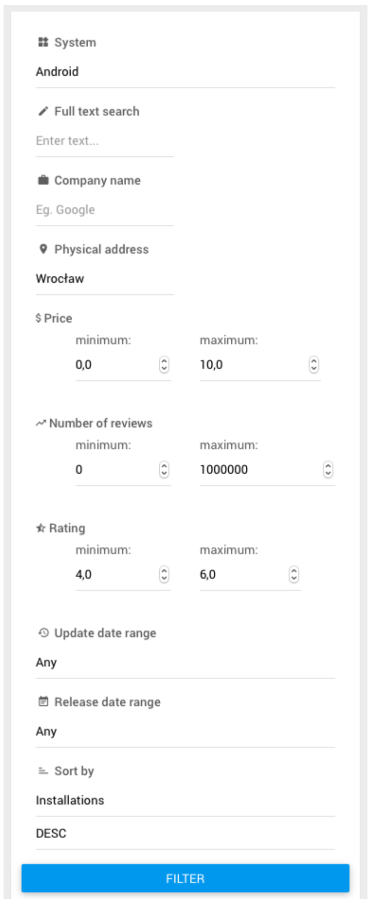
Sort by
Installations
DESC

ICON	NAME	COMPANY NAME	INSTALLS	RATING	CATEGORY	PRICE
	Samsung Push Service	Samsung Electronics Co., Ltd.	1 000 000 000	4.1	Communication	FREE
	Gmail	Google Inc.	1 000 000 000	4.3	Communication	FREE
	Google Play Movies & TV	Google Inc.	1 000 000 000	3.6	Media & Video	FREE
	Google TalkBack	Google Inc.	1 000 000 000	4.0	Tools	FREE
	Google Text-to-speech	Google Inc.	1 000 000 000	4.0	Tools	FREE
	Google Play Books	Google Inc.	1 000 000 000	3.9	Books & Reference	FREE
	Google+	Google Inc.	1 000 000 000	4.2	Social	FREE
	Google Play Music	Google Inc.	1 000 000 000	3.9	Music & Audio	FREE
	Facebook	Facebook	1 000 000 000	4.0	Social	FREE
	Gmail	Google Inc.	1 000 000 000	4.3	Communication	FREE

66110 entries by current filter

Figure 6.19: Simple filters with predefined criteria

4. Enter criteria 6.20



System
Android

Full text search
Enter text...

Company name
Eg. Google

Physical address
Wrocław

$ Price
minimum: 0,0
maximum: 10,0

Number of reviews
minimum: 0
maximum: 1000000

Rating
minimum: 4,0
maximum: 6,0

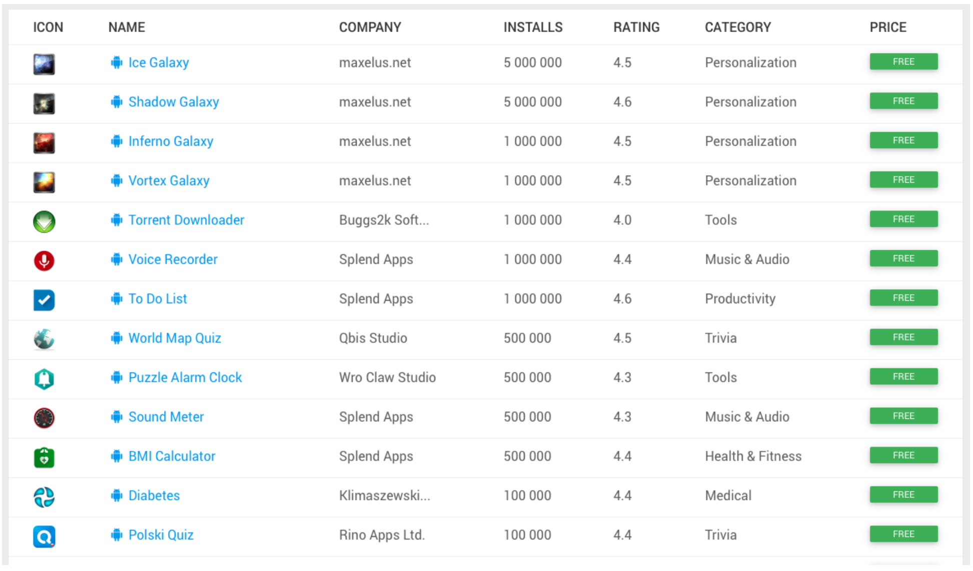
Update date range
Any

Release date range
Any

Sort by
Installations
DESC

Figure 6.20: All available criteria

5. Click "Filter" button.
6. Filtered applications will be shown on list.

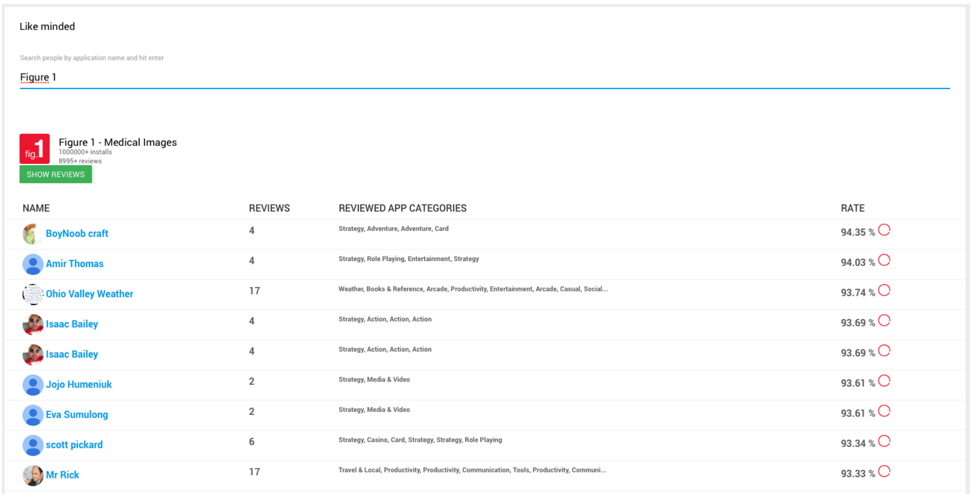


ICON	NAME	COMPANY	INSTALLS	RATING	CATEGORY	PRICE
	Ice Galaxy	maxelus.net	5 000 000	4.5	Personalization	FREE
	Shadow Galaxy	maxelus.net	5 000 000	4.6	Personalization	FREE
	Inferno Galaxy	maxelus.net	1 000 000	4.5	Personalization	FREE
	Vortex Galaxy	maxelus.net	1 000 000	4.5	Personalization	FREE
	Torrent Downloader	Buggs2k Soft...	1 000 000	4.0	Tools	FREE
	Voice Recorder	Splend Apps	1 000 000	4.4	Music & Audio	FREE
	To Do List	Splend Apps	1 000 000	4.6	Productivity	FREE
	World Map Quiz	Qbis Studio	500 000	4.5	Trivia	FREE
	Puzzle Alarm Clock	Wro Claw Studio	500 000	4.3	Tools	FREE
	Sound Meter	Splend Apps	500 000	4.3	Music & Audio	FREE
	BMI Calculator	Splend Apps	500 000	4.4	Health & Fitness	FREE
	Diabetes	Klimaszewski...	100 000	4.4	Medical	FREE
	Polski Quiz	Rino Apps Ltd.	100 000	4.4	Trivia	FREE

Figure 6.21: Filtered applications by given criteria

6.5.4. How to find like-minded people for given application

1. Open http://localhost:9000/user in web browser.
2. Choose "Like-minded people" from menu on left side.
3. Enter application name and click *Enter* button on keyboard 6.22



Like minded

Search people by application name and hit enter

Figure 1

Figure 1 - Medical Images
1000000+ installs
8995+ reviews

SHOW REVIEWS

NAME	REVIEWS	REVIEWED APP CATEGORIES	RATE
BoyNoob craft	4	Strategy, Adventure, Adventure, Card	94.35 %
Amir Thomas	4	Strategy, Role Playing, Entertainment, Strategy	94.03 %
Ohio Valley Weather	17	Weather, Books & Reference, Arcade, Productivity, Entertainment, Arcade, Casual, Social...	93.74 %
Isaac Bailey	4	Strategy, Action, Action, Action	93.69 %
Isaac Bailey	4	Strategy, Action, Action, Action	93.69 %
Jojo Humeniuk	2	Strategy, Media & Video	93.61 %
Eva Sumulong	2	Strategy, Media & Video	93.61 %
scott pickard	6	Strategy, Casino, Card, Strategy, Strategy, Role Playing	93.34 %
Mr Rick	17	Travel & Local, Productivity, Productivity, Communication, Tools, Productivity, Communi...	93.33 %

Figure 6.22: Sorted recommendations for all people from database

Chapter 7

Conclusions

The project was mostly concentrated on the *slope one* algorithm because of several reasons. People are creating content in the Internet every moment. In growing amount of data it is more and more difficult to find useful information. These days tagging and categorizing content is not enough to keep order in the Internet resources. That is why recommender algorithms was invented. Computer programs help us in highlighting most interesting content for single user or group of users. Increasing demand for data scientists in IT segment, shows how big data problem we were facing with. Many large companies like Google or Amazon are trying to improve user experience of their products by using machine learning. Between 2006 and 2009, Netflix was hosting competition [1] for best algorithm to predict movies for users.

The created application is good start for implementing more complicated prediction algorithms. Right now, that huge amount of gathered data is excellent environment for any machine learning system. Database contains thousands of people reviews with text and integer value of rate. That could be used for using natural language processing, which can extract most common phrases and used by developers for advertising. Web module as like as worker module could be modified in many ways to provide better and better results.

The strength of project is it's modularity which was achieved by using design patterns in proper way and Spring framework which can be extended by other plugins or frameworks. Docker platform provide scalability and portability, project can be deployed on another server instance in seconds by using Docker configuration files.

Java, as primary language, was weak a point of project. Better results could give usage of Scala or Kotlin. Both languages are supported by Spring Framework and both are JVM languages. The cost is a slower execution time because of surplus byte code which is generated during compilation. Profit is less consuming development time and more readable code. Scala is combined object oriented paradigm with functional paradigm, the function part is very useful while writing algorithm. The main focus is concentrated on that how something is made instead of creating abstraction level for acquiring necessary data from objects. Persisting the similarity matrix in MongoDB is worth to consider in case of further development. Probably better result will give any in-memory database, which could be saved on hard drive every some amount of time as backup. That solution could accelerate similarity matrix creation as well as prediction phase. Besides that two exceptions, used tools were enough in this project. All of requirements were achieved and main project goal has been executed.

[1] E. Buskirk, "How the Netflix Prize Was Won", 22.09.2009,
https://www.wired.com/2009/09/how-the-netflix-prize-was-won/ (15.11.2016)

Bibliography

[1] Daniel Lemire and Anna Maclachlan. Slope One Predictors for Online Rating-Based Collaborative Filtering. 2005.

[2] Lior Rokach, Francesco Ricci, and Bracha Shapira. *Recommender Systems Handbook*. Springer US, 1 edition, 2010. ISBN 978-0-387-85820-3. doi: 10.1007/978-0-387-85820-3.

[3] Dietmar Jannach, Tu Dortmund, and Gerhard Friedrich. *Recommender Systems*. 2013.

[4] Bruce Eckel. *Thinking in Java*. Prentice Hall, 4 edition, 1998. ISBN 978-0131872486.

[5] Rod Johnson, Juergen Hoeller, Keith Donald, Colin Sampaleanu, Rob Harrop, Thomas Risberg, Alef Arendsen, Darren Davison, Dmitriy Kopylenko, Mark Pollack, Thierry Templier, Erwin Vervaet, Portia Tung, Ben Hale, Adrian Colyer, John Lewis, Costin Leau, Mark Fisher, Sam Brannen, Ramnivas Laddad, Arjen Poutsma, Chris Beams, Tareq Abedrabbo, Andy Clement, Dave Syer, Oliver Gierke, Rossen Stoyanchev, Phillip Webb, Rob Winch, Brian Clozel, Stephane Nicoll, and Sebastien Deleuze. Spring Framework Reference Documentation 4.3.4.RELEASE, 2016.

[6] Daniel Fernández. Tutorial: Using Thymeleaf, 2016. URL `http://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html`.

[7] Twitter Bootstrap reference, 2016. URL `http://getbootstrap.com`.

[8] Craig Walls. *Spring in Action*. Manning Publications, 4 edition, 2014. ISBN 9781617291203.

[9] Timo Westkämper, Samppa Saarela, Vesa Marttila, Lassi Immonen, Ruben Dijkstra, John Tims, and Robert Bain. Querydsl Reference Guide, 2016. URL `http://www.querydsl.com/static/querydsl/4.1.3/reference/html{_}single/`.

[10] Docker Reference. URL `https://docs.docker.com`.

[11] David Grossman and Yuval Merhav. *Item Based Recommenders*. 2010.

[12] Material Admin Template, 2016. URL `https://wrapbootstrap.com/theme/material-admin-responsive-angularjs-WB011H985`.

[13] Google+ API Reference, 2016. URL `https://developers.google.com/+/web/signin/`.

[14] Maven Reference, 2016. URL `https://maven.apache.org`.

[15] Tomcat Reference, 2016. URL `http://tomcat.apache.org`.

[16] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1 edition, 1994.

[17] PayPal REST API Java SDK, 2016. URL `https://github.com/paypal/PayPal-Java-SDK`.

[18] Jsoup API reference, 2016. URL `https://jsoup.org/apidocs/`.

[19] Jackson Documentation, 2016. URL `http://wiki.fasterxml.com/JacksonDocumentation`.

[20] Tim Peierls, Brian Goetz, Joshua Bloch, Joseph Bowbeer, Doug Lea, and David Holmes. *Java Concurrency in Practice*. Addison-Wesley Professional, 1 edition, 2006. URL `https://www.amazon.com/gp/product/B004V9OA84`.

Appendix A

CD/DVD content

- Application code
- Docker configuration files