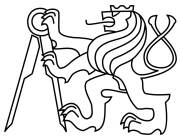
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



ASSIGNMENT OF BACHELOR'S THESIS

Title:	ElateMe - Project management and Advert server
Student:	Maksym Balatsko
Supervisor:	Ing. Jiří Chludil
Study Programme:	Informatics
Study Branch:	Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of summer semester 2017/18

Instructions

The ElateMe system is a web application with mobile clients providing functionality of crowdfunding and wishlist satisfaction of users and their friends (e.g., for Christmas, birthday, etc.).
1. Analyse/create:
 - risk list, domain model, and criteria of the project success,
 - BPMN diagrams for the wish-and-surprise ElateMe processes,
 - methodology for tagging users to target groups.
2. Design:
 - a class model for users' data and business management,
 - an AdServer (Advert Server) database model,
 - a daemon for categorizing users according to their behavior in ElateMe and the information from their Facebook profiles.
3. Implement:
 - an AdServer prototype,
 - an XML interface for importing products for wishlists,
 - a cookie collector of user devices.
4. Test the server performance and create unit tests.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague February 15, 2017

Czech Technical University in Prague
Faculty of Information Technology
Department of Software Engineering

Bachelor's thesis

ElateMe - Project management and Advert server

Maksym Balatsko

Supervisor: Ing. Jiří Chludil

15th May 2017

Acknowledgements

I would like to thank Ing. Jiří Chludil and project leader Michal Maněna for their valuable advice and cooperation in the creation of the whole project. I would also like to thank the entire ElateMe team, namely: Yevhen Kuzmovych, Georgii Solovev, Gleb Arkhipov, Yegor Terokhin. Thanks to my parents for their patience and unwavering support.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on 15th May 2017 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Balatsko, Maksym. *ElateMe - Project management and Advert server.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

Abstrakt

Tato práce je součástí komplexního startup projektu platformy crowdfundingu s elementy sociální sítě ElateMe. Hlavní myšlenkou ElateMe je, že uživatel může na svém webu zveřejnit své přání a jeho přátelé mu mohou finančně pomoci při jeho plnění. Tento projekt se skládá z mobilních aplikací pro platformy iOS a Android, webové aplikace, RESTful backend serverů a přesně té části, která je hlavním cílem této práce, Advert server.

Reklamní server je navržen tak, aby generoval odpovídající reklamní bloky podle seznamu produktů získaných od třetí strany, což uživatelu pomůže urychlit proces vytváření přání a usnadnit výběr potenciální věci, kterou potřebuje, a chce koupit. Reklamní bloky budou personalizovány, což znamená, že tyto bloky jsou generovány podle historie chování uživatele a jemu podobně v rámci aplikace ElateMe.

Hlavním cílem práce je realizace prototypu reklamního serveru. Pro tyto účely byla provedena analýza možných řešení a zvolené řešení bylo zdůrazněno. Na základě toho byla navržena aplikační architektura a popsán implementační přístup. Pro účely znovupoužitelnosti byla navržena a úspěšně implementována schéma automatizace nasazení. Z hlediska ElateMe tato práce zahrnuje analýzu hlavních business procesů a manažerské vyhodnocení projektu v rámci hodnocení rizik a kritérií úspěšnosti projektu.

Klíčová slova ElateMe, platforma crowdfundingu, systém doporučení, collaborative filtering, reklamní server, automatizace nasazení

Abstract

This thesis represents a part of a complex startup project of a crowdfunding platform with elements of social network ElateMe. The main idea of ElateMe is that the user can post his wish on the web and his friends can assist him financially to fulfill it. This project consists of mobile applications for iOS and Android platforms, web application, RESTful backend server API and exactly the part, which is the main objective of this thesis, Advert server.

Advert server is designed to generate an appropriate targeting advertisement blocks according to the products feed received from a third-party service, which will help users to speed up the process of wish creation and facilitate the choice of a potential thing they need and want to purchase. The advertisement blocks will be personalized, which means that these blocks are generated according to the behavior history of a user and users similar to him in the framework of the ElateMe application.

The main aim of this thesis is to implement an Advert server prototype. For such purposes, the analysis of possible solutions was performed and the chosen solution was highlighted. Based on that the application architecture was designed and an implementation approach of it was described. For the reusability purposes, the automation deploy scheme was designed and successfully implemented. From the ElateMe point of view, this thesis comprises the analysis of main business processes and managerial project evaluation in the framework of risks assessment and criteria of project success.

Keywords ElateMe, crowdfunding platform, recommendation system, collaborative filtering, advert server, automation deploy

Contents

Introduction **1**
What is crowdfunding 1
ElateMe 1
Aim of the thesis 2
Motivation 2

1 Analysis **3**
1.1 ElateMe platform 3
1.2 Advert server 10

2 Design **17**
2.1 Stack of technologies 17
2.2 Users' grouping 18
2.3 Recommendation system model 19
2.4 Advert server database model 20
2.5 Advert server class model 23

3 Implementation **27**
3.1 Project structure 27
3.2 Deployment 28

4 Testing **33**
4.1 Unit testing 33
4.2 Performance tests 35

5 Managerial project evaluation **37**
5.1 Risk assessment 37
5.2 Criteria of project success 38

Conclusion **39**

Future outlook 39

Bibliography **41**

A Acronyms **45**

B Installation guide **47**
B.1 Requirements 47
B.2 Dependencies installation 47
B.3 Setup 48

C Contents of enclosed CD **49**

List of Figures

1.1 ElateMe Domain model 3
1.2 ElateMe Wish lifecycle business process diagram 5
1.3 ElateMe Creating a wish business process 7
1.4 ElateMe Donating to a wish business process 8
1.5 ElateMe Wish closure business process 9
1.6 ElateMe Refundation business process 10

2.1 Users collection schema 21
2.2 Products collection schema 21
2.3 Socio-demographic groups collection schema 22
2.4 Advisor class 23
2.5 Validation class 24
2.6 Handlers classes 25

3.1 Deployment diagram 28

Introduction

What is crowdfunding

Primarily, it is necessary to figure out what the notion of crowdfunding means. Crowdfunding is a financing method that involves funding a project with relatively modest contributions from a large group of individuals, rather than seeking substantial sums from a small number of investors.[1]

Nowadays, most crowdfunding services work online, but the history of crowdfunding dates back long before the first Personal computer (PC) invention, not to mention the Internet.

Crowdfunding has a long story with several roots. Earlier one of the ways of crowdfunding was collecting the subscriptions from the population before the book was printed. If the book gained enough subscribers it was published. This procedure guaranteed the investors that the book circulation would be sold out. War bonds were theoretically a form of crowdfunding military conflicts.[2]

At first, crowdfunding on the Internet gained popularity in the USA, when the ArtistShare was launched in 2003. Brian Camelio, an American musician and software developer, was the designer of this platform. The main idea of this project was that fans could donate money to their favorite musicians to help produce their digital recordings. After a while, it turned into a platform focused not only on music but film/video and photography too.[3] Nowadays there are more than 2,000 functioning crowdfunding platforms.

ElateMe

ElateMe is a startup project, which is a crowdfunding platform with some elements of a social network. However, this project is not a crowdfunding platform in the classic sense of this term. The main idea of this project can be stated as follows: “Everybody has some desires, but not always can afford to carry them out. In this case, friends can assist them financially. The only

thing you have to do is to write, what exactly you want. Also, there is a chance to create a surprise wish for your friend, which he will not know about until the total amount of money is gathered. A birthday surprise could be a good example of such a wish". Based on this, it is clear, that ElateMe firstly aims at the personal needs of a particular person, unlike other large crowdfunding services whose main purpose is to maintain group and commercial projects.

In such a way, ElateMe will not have a commission charge for payments, which became a prerequisite to invent another method of project monetization. That is to create a special advertising server, which has to generate an appropriate products suggestions for users to create wish with, which means that it has to act as a recommendation system.

Aim of the thesis

The aim of this thesis is to analyze the options of building an Advert server for ElateMe platform, design the recommendation system model and implement an Advert server prototype. From the ElateMe point of view, the author has to define the risk list, analyze the criteria of the project success and business process of a wish lifecycle in the framework of ElateMe application.

Motivation

The goal of the author of this thesis is to analyze and learn the principles of functioning and designing the recommendation systems, structure the knowledge of developing backend applications using Python language and asynchronous web framework Tornado, improve the experience of using databases with learning an alternative to relational, namely NoSQL database MongoDB.

Chapter 1

Analysis

1.1 ElateMe platform

1.1.1 Domain model

To describe relationships and interactions between the main entities of the application it was decided to create a domain model. In software engineering, a domain model is a conceptual model of the domain that incorporates both behavior and data.[4]

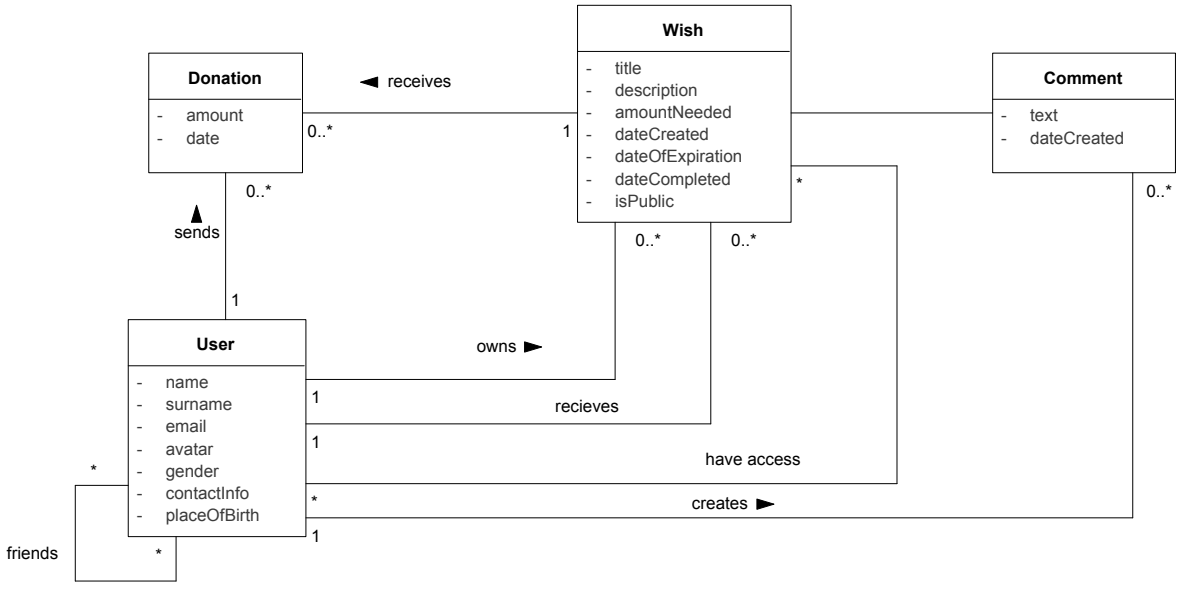


Figure 1.1: ElateMe Domain model

1.1.1.1 User

The User entity presents someone, who is registered in ElateMe platform. In ElateMe properties of the user are name, surname, email, avatar, gender, date of birth. For the reason that in our actual state of the development we have only authorization and registration to our application using Facebook, all this data is received from Facebook. As we can see on the diagram 1.1 user can have some friends. The friends are determined by the user's friends relationships on Facebook.

1.1.1.2 Wish

A Wish presents something the User wants to collect money for. The wish can be created for the author himself or for the author's friend. Also, the user sets the access rights for the wish, that is who may see this wish in their feed, could donate money and comment. A wish has required properties, which are to be set by the user: title, description, image and needed amount of money to fulfill the wish, and an optional property - the date of the expiration, or deadline, that is the latest date to gather the needed sum of money. The date created and date closed properties are auto set on the server when the wish is created or closed respectively. The processes of wish creation and closure are described in subsection 1.1.2

1.1.1.3 Comment

User can comment on the wishes, which he has access to. Comment has only one manual property - text, which represents the body of the comment. The date is auto set according to the creation date. The commenting process is described in subsection 1.1.2

1.1.1.4 Donation

User can donate to wishes, which he has access to. User sets the amount of the Donation and payment method. The date is auto set according to the creation date. The processes of donation and refundation are described in subsection 1.1.2

1.1.2 Wish lifecycle business process

For the demonstration of essential functionality and usage of the application was decided to create business processes diagram 1.2. A business process is a series of steps performed by a group of stakeholders to achieve a concrete goal. These steps are often repeated many times, sometimes by multiple users and ideally in a standardized and optimized way.[5] As the main entity of our application is a Wish, therefore in this section the wish lifecycle business process is described.

In this case the stakeholders of this business process are:

- User/Wish Author - a user, who has created a wish and whose goal is to fulfill it
- Donator - a user, whose aim is to donate to created wish
- Commentator - a user, whose aim is to comment created wish
- Wish recipient - a user, for whom the Surprise wish is created

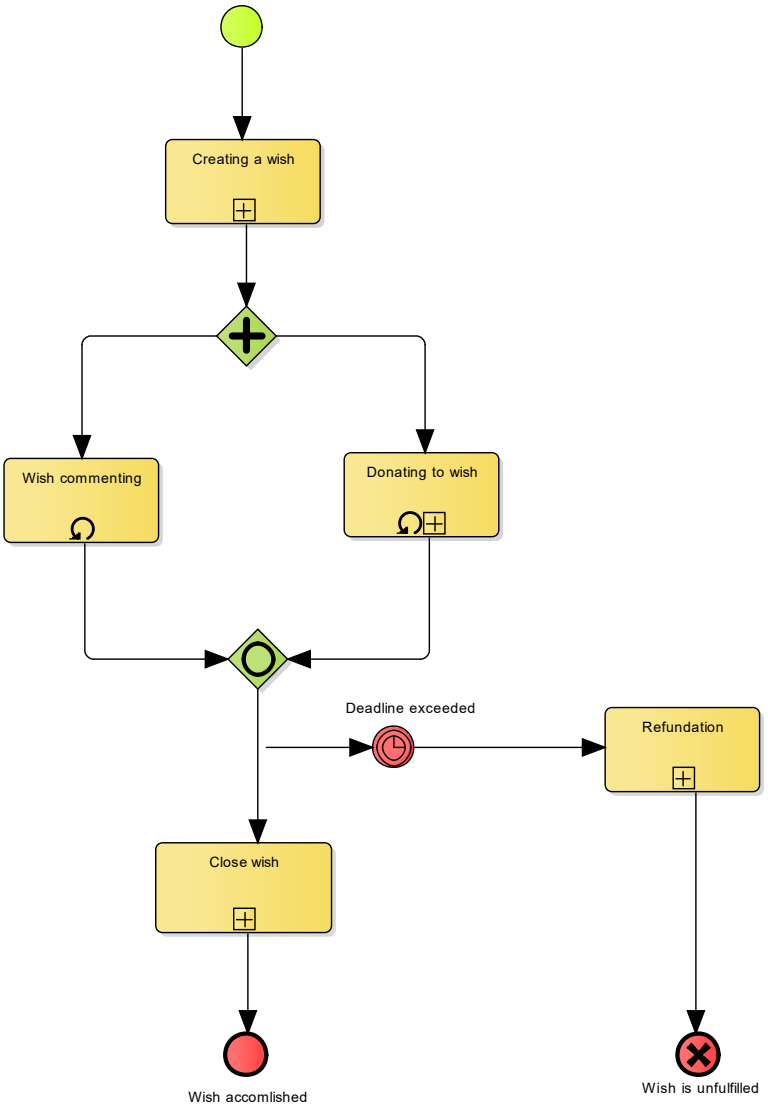


Figure 1.2: ElateMe Wish lifecycle business process diagram

1.1.2.1 Creating a wish

The initial step of the wish lifecycle business process is Creating a wish, which starts when a user performs a "create wish" action, that can be done both on the "Feed" screen and "My wishes" screen. Then the user is redirected to the "what do you wish?" screen, where he can write down the name of the preferred gift.

The application reacts to the user's input and shows the "suggestions" generated by the advert server. During the process of typing the user can choose, whether to keep the title and continue creating the wish using his own data, or to use a pregenerated wish based on the suggestion, in which most data are prefilled.

If the user has chosen one of the suggestions creating the wish, he can also choose a preferred shop to make the purchase in.

Then the user has to decide on the wish recipient choosing one of the following two options:

1. to create a wish for himself, a common wish
2. to create a Surprise wish for his friend from the friendlist.

In the first case, the user has to define the visibility level:

1. Public, everyone from his friendlist can see this wish
2. Private, only the people, who the author will grant the access to can see this wish.

In case the user has chosen the Surprise wish creation, he has to grant the access to the "donators group", which is a group of users, for whom the wish is visible and they can donate to it.

The last steps of the wish creation are identical for the both types of wishes, the user is offered to choose the deadline and the description.

After the wish is created it is displayed in the user's friends feeds according to the adjusted visibility rights, so friends begin to interact with it.

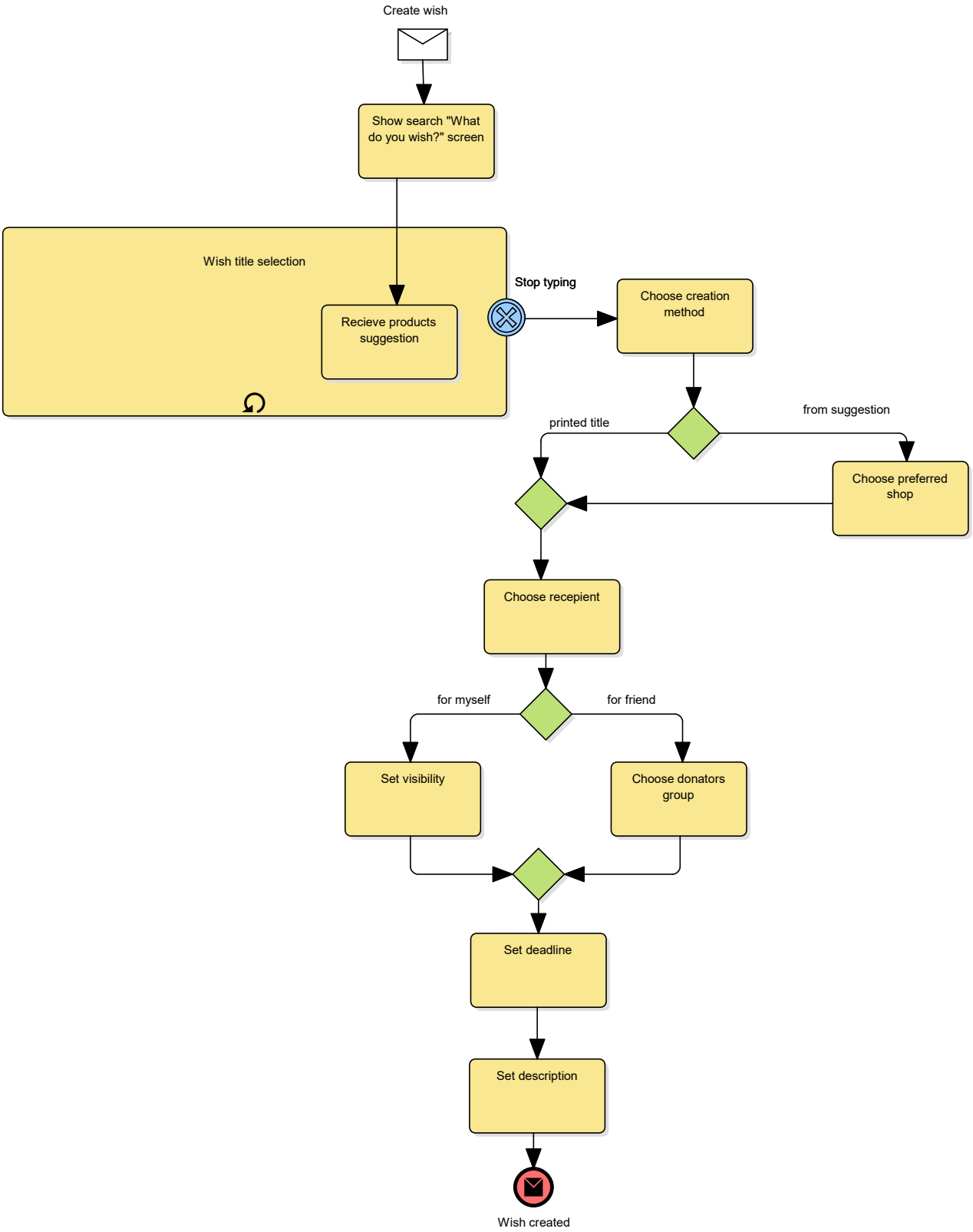


Figure 1.3: ElateMe Creating a wish business process

1.1.2.2 Wish commenting

Having enough access rights to the wish, the user's friend becomes a potential commentator. The flow of commenting the wish is straightforward and common: under the wish, the commentator has an input field, where he can write his comment and submit it. Then it will be displayed in the comments section of the wish details. A wish can be commented an unlimited number of times until it is closed.

1.1.2.3 Donating to wish

Having enough access rights to the wish, the user's friend becomes a potential donator. This process starts when the donator performs a "donate" action. Then he is offered to enter some amount to donate. After that, he chooses the payment method to proceed the donation transaction. Our platform accepts payments via Credit card VISA/Mastercard or Bitcoin. Then the donator is to confirm the payment. If the payment succeeds, the wish author gets the notification, that someone has donated to his wish.

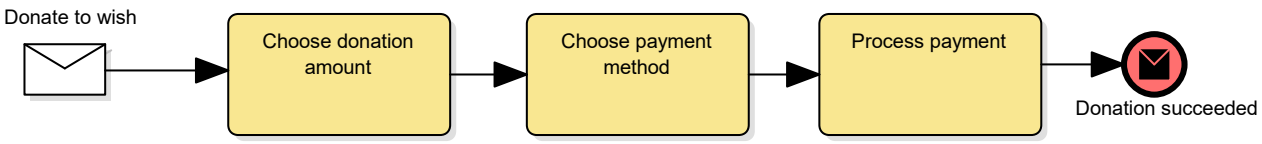


Figure 1.4: ElateMe Donating to a wish business process

1.1.2.4 Wish closure

The wish has 3 possible ways to end the lifecycle up:

1. The set deadline of the wish has exceeded. In this case the Refundation(1.1.2.5) task is performed, the wish is no more visible both to the author and his friends, and the wish is marked as unfulfilled.
2. The user closes the wish manually. In this case, the flow of closure is the same as in the previous one.
3. The wish fulfillment. This task is launched when the required amount of money is gathered.

If it is a common wish, the gathered money is transferred to the wish author bank account, the wish is marked as fulfilled and the notifications about the fulfillment are sent to the author and all the donees.

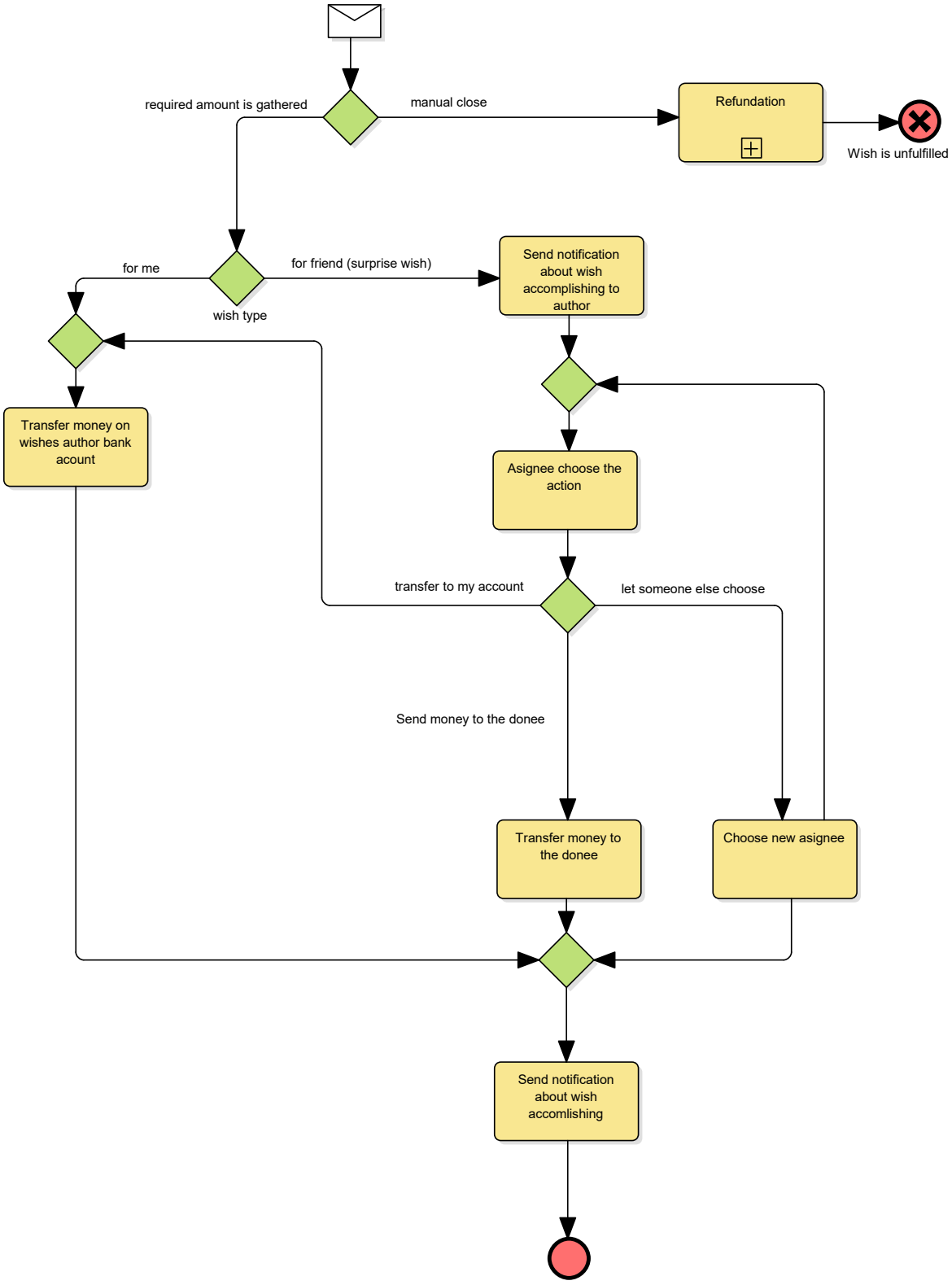


Figure 1.5: ElateMe Wish closure business process

If it is a Surprise wish, the notification about the fulfillment is sent only to the wish creator who is responsible for defining the further action flow. The first option the wish author can choose is to transfer the gathered money directly to his own bank account, the second one is to transfer the money to the wish recipient bank account, and the last one is to let someone else choose from these three possibilities. And this cycle repeats until someone chooses "money transfer". After all money transactions have successfully proceeded, the wish is marked as fulfilled, all the donators, the recipient and the author of the wish are notified that the wish has been accomplished.

1.1.2.5 Refundation

The process of refundation starts when the wish fulfillment fails. The sum of all donations is counted for each donator separately and transferred to their bank or bitcoin accounts, respectively to the donation method they had chosen during the donation creation. After that, the notification about the wish fulfillment failture is sent to all the donators.

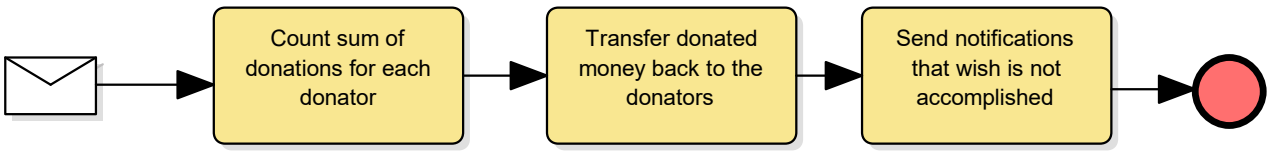


Figure 1.6: ElateMe Refundation business process

1.2 Advert server

As it is described in the introduction, ElateMe will not have a commission charge for payments, which has become a prerequisite to invent another method of project monetization. That is to create an individual advert server, which has to generate an appropriate targeting advertisement blocks according to the products feed received from a third-party service. So the advert server will represent a Recommendation system.

1.2.1 What is recommendation system

Recommendation systems are services, which try to predict what objects a user will like, according to his personal information or his behavior. Companies use recommendation systems on their websites to provide as much useful/valid information for the concrete user as possible to keep involved within the framework of their own web site, trying to make the user perform the target action they are interested in. Recommendation systems are useful both for the user - spending less time on searching and filtering himself, and the service provider - user retention on the website.

Within this bachelor thesis, recommendation systems are divided into three main categories:

- **Content-based**

  The user is recommended objects similar to the ones that he has already used. The similarity is evaluated based on the content of the object, e.g. for the film - film producer, film company, etc. However, this creates a problem of strong dependance on the subject area.

- **Collaborative filtering**

  The main idea of this method can be formulated in such a way: "Similar users like similar objects". Unlike the Content-Based methodology in this case not only the user's rating of the object is taken into account, but also the rating of other users. Therefore this method is more universal and gives us better results. Problems arising here are as follows: cold start[1], disregard to objects contents, disregard to the context of user actions.

- **Hybrid**

  This method combines the two aforementioned methods. The hybrid system is usually used to solve the problem of cold start of the collaborative filtering, it is more complicated and harder to design and implement, because of the difficulties with the compatibility of two methods. It is reasonable to build the hybrid system when the sufficient amount of data from users is captured on the fly[2], because the combination of methods is very sensitive to the data specificity.

Recommendation systems started to appear on the web quite long ago, about 25 years ago. However, the real rise in this subject area began about 7-15 years ago, when the Netflix Prize competition took place. In those times Netflix company rented out not digital copies of films but sent out Video Home System (VHS)-cassettes and Digital Versatile Disc (DVD)s. It was crucial for them to improve the quality of the recommendations. The better Netflix company recommended movies to its users, the more movies were rented. According to that, the company's profit also grew. In 2006, they launched the Netflix Prize competition. They posted the collected data into an open access: about 100 million five-point scale ratings with the Identifier (ID) of the users who put them. The participants of the competition had to predict what kind of evaluation the particular user will put to a certain film as best as possible. The quality of the prediction was measured using the Root-Mean-Square Error (RMSE) metric. Netflix already had an algorithm that predicted

[1] Cold start - new objects are recommended to nobody

[2] In relation to computer technology, "on the fly" describes activities that develop or occur dynamically rather than as the result of something that is statically predefined.[6]

user ratings with a quality of 0.9514 by the RMSE metric. The task was to improve the prediction by at least 10% - to 0.8563. The winner was promised a prize of $ 1 million. The competition lasted for about three years. For the first year, the quality was improved by 7%, then everything slowed down a little. But in the end, two teams sent their solutions each of which passed the threshold of 10% with a time difference of 20 minutes; the quality was the same to the fourth digit. These twenty minutes clinched the deal over which many teams had struggled for three years. The team, which was late (like many others who participated in the competition) were left with nothing, but the competition itself greatly spurred development in this area.[7]

1.2.2 Requirements

1.2.2.1 Functional requirements

F1 Advert server has to generate a set of recommended products for the particular user in the scope of the "What do you wish?" screen. There are two variants: common generation of products set (when the user has just entered the screen), generation of products set according to the user input, which is a search query string. Each item of the generated products set has to contain all the required data for wish creation.

F2 Advert server has to provide Application Programming Interface (API) to receive cookie data from client applications and save it to the data storage.

F3 Advert server has to maintain user grouping into socio-demographic groups.

F4 Advert server has to provide API for the import of products used for recommendations from the third-party services.

1.2.2.2 Non-functional requirements

N5 Products import interface has to accept the Extensible Markup Language (XML) format of imported data.

N6 Advert server has to be developed using asynchronous backend framework or asynchronous networking library.

1.2.3 Problem statement

The main aim of the Advert server is to provide the method of project monetization, whose main idea is that third-party services will provide their products to the ElateMe application. The received data is saved in the data storage and the user is suggested to create wishes using the provided products. Such a suggestion in the application is called the recommendation.

A particular pricing method for user target action performance is not yet designed. But even now it is possible to say, that the more users use the generated recommendations, the more money ElateMe service earns. So the Advert server has to produce a valid and interesting recommendation for a particular user and predict what exactly the user will like. Therefore it has to act as a Recommendation system.

First of all, we have to say that our application does not have the concrete focus on some sphere, e.g. music, filmography, etc. Consequently, a content-based recommendation system would not be appropriate for us. On the contrary, the collaborative system ideology suits our interests. Every Recommendation system needs some input data to generate a recommendation. That is why it is necessary to decide, what data we need to generate appropriate recommendations.

1.2.4 Possible solutions

To illustrate possibilities of collaborative filtering recommendation system realization, let us consider the following input data: a multiplicity of users $u \in U$, a multiplicity of products $p \in P$ and a multiplicity of events $(r_{up}, u, p) \in E$. All the events are set with the user u that performs the event, the product p, over which the event is performed and the event's result r_{up}. Let us call the result of the performed event *rating*. We are to do:

- **Predict user preferences**
  $r_{up} = Predict(u, p)$
- **Generate a personal recommendation**
  $u \longmapsto (p_1, ..., p_k) = Recommend(u)$
- **Identify the similarity between the users**
  $sim(u_n, u_k)$

Let us consider possible prediction algorithms. Proceeding the idea that: "Similar users like similar objects", first thing we can do is to divide the users into several target groups according to their similarity. Such an algorithm in machine learning is called clustering, and the result groups are called clusters.

So the first method, which may be used is as follows: Supposing that we have found the function to determine the similarity of users, who then have been divided into clusters so that the similar users are in the same cluster. The user's product rating r_{up} will be predicted as a mean of all users' ratings of the product in this cluster:

$$r_{up} = \frac{1}{|C|} \sum_{v \in C} r_{vp}, \text{ where C is cluster}$$

Problems arising in this algorithm are as follows:

- Nothing is to be recommended for new/"non-typical" users.
- An untypical user is hard to be placed in a suitable cluster.
- If nobody in a cluster has placed the products rating, the prediction can not be performed.

How can this algorithm be improved? The author offers to give up rigid clustering, applying one of two methods: User-based or Item-based.

In the User-based method, the formula of rating prediction r_{up}, for the user u and the product p can be formulated in such a way:

$$r_{up} = \overline{r_u} + \frac{\sum_{v \in U_p} sim(u,v)(r_{vp} - \overline{r_v})}{\sum_{v \in U_p} sim(u,v)}$$

Here r_u represents the average rating of products, with which user u has already interacted. This value describes the accustomed behavior of a particular user. The fraction in this formula represents weighted arithmetic mean, where the function $sim(u, v)$, similarity between the users, is weight, the $(r_{vp} - r_v)$ tells us if the user v rated product p more or less than on the average. Arithmetic mean computed by considering relative importance of each item is called weighted arithmetic mean.[8] To summarize, this method takes into account the user's common behavior of rating the products and corrects the output with the averaged rating collected from the similar users, based on the main idea of the collaborating filtering that the similar users like similar objects. The problems arising here are the cold start and the lack of recommendations to new/"non-typical" users.

The item-based method is a symmetrical algorithm to the User-based method. Unlike User-based ideology, this algorithm is based on the idea that a user will like the object if he likes objects similar to it. The formula of rating prediction r_{up}, for the user u and the product p can be formulated in such a way:

$$r_{up} = \overline{r_p} + \frac{\sum_{k \in P_u} sim(u,v)(r_{uk} - \overline{r_k})}{\sum_{k \in P_u} sim(u,v)}$$

Here we use the average rating of the product r_p and correcting the prediction with weighted arithmetic mean, where the weight is function $sim(p, k)$ that calculates the similarity between products p and k. One of the problems of this method is the same as in User-based one, it is the problem of the cold start. Another issue arising here is "trivial" recommendations. Trivial, in this case, means, first of all, that the recommendations for the user will be non-personalized and that the bigger part of recommendations will consist of the objects that most users like.

These two methods have at least two common problems: cold start, as it was aforewritten, and the resource intensity of calculations with the high number of users, because every time it is necessary to calculate the prediction using the ratings of all the users or all the ratings of all the products respectively.[7][9]

The second thing we have to consider is metrics of distance between objects, or in our case similarity degree. Within the framework of this bachelor thesis, the main frequently used variants of calculation of this metrics will be highlighted:

- **Euclidean distance**

  The most common function of distance calculation. It represents the distance between vectors x, y in n-dimensional vector space.

$$\rho(x, y) = \sqrt{\sum_i^n (x_i - y_i)^2}$$

- **Manhattan distance**

  This function represents the sum of vectors x, y coordinates differences in n-dimensional vector space.

$$\rho(x, y) = \sum_i^n |x_i - y_i|$$

- **Chebyshev distance**

  This metric is calculated as the maximum of modulus difference between the coordinates of vectors x,y in n-dimensional vector space.

$$\rho(x, y) = max(x_i - y_i)$$

- **Pearson correlation coefficient**

  This metric is a measure of linear correlation between two variables X, Y in a dataset. In our case the eventual values of variable X and Y will be presented in vectors x, y in n-dimensional vector space.

$$\rho(x,y) = \frac{\sum_i^n (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_i^n (x_i - \overline{x})^2}\sqrt{\sum_i^n (y_i - \overline{y})^2}}$$

  [9][10][11]

1.2.5 Future outlook and possible improvements

The problem of the resource intensity of calculations with the high number of users in future will be resolved, so that instead of the multiplicity of all users that have rated the product, only the $k \in N$ nearest neighbors that have rated the product will be used. This method will standardize and speed up the time of the prediction computation, on account of the use of a fewer amount of users for the calculations.

ElateMe application does not provide an opportunity for a qualitative evaluation of the product by a user. The first thing to be introduced is the system of likes/dislikes or just likes. In future the author puts forward the idea either to create some notification mechanism offering the user to rate the product he received, which can appear in one or several weeks to make sure that the user has already tried to use the product and formed an opinion about it, or just leave the opportunity for the user to rate the product any time after the wish is fulfilled.

Chapter 2

Design

2.1 Stack of technologies

2.1.1 Python

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.[12] The author has chosen to use Python as the main programming language guided primarily by his two-year experience of work with this language. Moreover, there are a lot of developed libraries on Python to work with data analysis and processing.

2.1.2 Tornado Web Server

Tornado is a Python web framework and asynchronous networking library, originally developed at FriendFeed. Tornado application could be easily scaled with increasing number of users. Traditional synchronous web frameworks, for example, Django, uses a thread per client methodology, which could be very expensive in high loaded applications. Unlike that Tornado is supposed to handle all the opened connections using only one thread. It becomes possible though using a single-threaded event loop. Event loop is a programming construct that waits for and dispatches events or messages in a program.[13] And all that means that the whole application has to be asynchronous and non-blocking because only one operation can be active at a time. An asynchronous function returns before it is finished, and generally causes some work

to happen in the background before triggering some future action in the application. A function blocks when it waits for something to happen before returning.[14] Due to the scalability and asyncrony Tornado was chosen as a primary framework for developing the ElateMe Advert server.

2.1.3 MongoDB

MongoDB was chosen as data storage for ElateMe Advert server. MongoDB is an open-source, document database designed for ease of development and scaling.[15] The key features of that technology used in the framework of this thesis are described in 2.4

2.1.4 nginx

nginx [engine x] is an Hypertext Transfer Protocol (HTTP) and reverse proxy server, a mail proxy server, and a generic Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) proxy server. For a long time, it has been running on many heavily loaded Russian sites including Yandex, Mail.Ru, VK, and Rambler. According to Netcraft, nginx served or proxied 28.72% busiest sites in April 2017. Here are some of the success stories: Netflix, Wordpress.com, FastMail.FM.[16] nginx usage in the Advert server application is described in 3.2

2.1.5 Ansible

Ansible is an Information technology (IT) automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates.

Ansible's main goals are simplicity and ease-of-use. It also has a strong focus on security and reliability, featuring a minimum of moving parts, usage of OpenSSH for transport (with an accelerated socket mode and pull modes as alternatives), and a language that is designed around auditability by humans–even those not familiar with the program. [17] In ElateMe advert server, Ansible will be used as an automation deploy tool. The details of usage are described in 3.2.1

2.2 Users' grouping

Users in ElateMe Advert server are to be grouped into socio-demographic groups. Every group will be allotted a set of rules pledged on available user properties, e.g. age, native country, gender, etc. On the user creation action or profile update action, the user will be regrouped according to the new data. The structure of rules is described in 2.4

2.3 Recommendation system model

ElateMe Advert server will be built as the User-based collaborative filtering system. The main reason is that our service has to generate personalized products recommendations. This means that the author had given up the idea to group the users according to their similarity, the concept of clustering. To identify the correct method of similarity function calculations, it is necessary to examine a dataset of real data obtained from users. As ElateMe is now in the development phase, the author is unable to perform that task. Therefore, the Euclidean distance function for ElateMe recommendation system was selected as the most shared and universal method of the similarity calculation. In our case, the Euclidean distance should be inverted because the further from each other the objects are located the less the value of similarity function it should acquire. So as the similarity function the difference between the maximum Euclidean distance in the current n-dimensional vector space and the value of the Euclidean distance between objects will be used.

The problem of cold start and the lack of recommendations will be resolved so that after the products are imported into our system, our specialist will manually set up a target socio-demographic group, to which the product is intended. As ElateMe is now in the development phase, the problem of the resource intensity of calculations will not be resolved for now. But a possible solution of this problem is highlighted in the subsection 1.2.5

Presently, ElateMe does not have the direct possibility for the user to rate the gift he received. Thus the author has to define the list of target actions that will simulate user rating action. As the exact rating scale does not exist, the author decided that rating value will be in the interval $[0; 1]$, also such an interval lets omit data normalization. The author poses the following list with target actions and its valuations:

- User hides wish - 0.1
- User collected $\frac{1}{4}$ of the needed amount of money for his wish - 0.25
- User closes the wish manually - 0.5
- User collected $\frac{1}{2}$ of the needed amount of money for his wish - 0.5
- User collected $\frac{3}{4}$ of the needed amount of money for his wish - 0.75
- User donates to the wish - 0.8
- User collected the full amount of money for his wish - 1

When one of these actions is performed, advert server receives the notification from the main server of ElateMe and assigns the corresponding value of the action as the user's product rating, if the product has already been rated by this user, the maximum value between the current and a new one is chosen.

To summarize, the final model for ElateMe Adver server looks like this:

U - users, P-products, E-events, r_{up} - rating of product p placed by user u

$$predict(u,p) = r_u + \frac{\sum_{v \in U_p} similar(u,v)(r_{vp} - \overline{r_v})}{\sum_{v \in U_p} similar(u,v)},\ u \in U,\ p \in P$$

$$similar(u,v) = \sqrt{|P_{uv}|} - \sqrt{\sum_{p \in P_{uv}} (r_{up} - r_{vp})^2}$$

2.4 Advert server database model

As was above written the main data storage in the Advert server application will be MongoDB, which is a document-based not only Sequence Query Language (NoSQL) database. Data in MongoDB is stored in collections, which are the analog of tables in a traditional relational database. The collection consists of documents. A document is an analog of a row in a table in a relational database and represents itself a BSON, which is a binary representation of JavaScript Object Notation (JSON), though it supports extended data types. MongoDB collection by itself is schemaless; therefore the format of data storing is defined on the database client. The only constraint imposed by Mongo is the uniqueness of the pre-reserved field _id, it the analog of a primary key in a relational database. In our Advert server application prototype the database will consist of 3 main collections: *users* collection, *products* collection, socio-demographic groups (*soc_demo_groups*) collection.

Users collection

A User in *users* collection has some basic attributes same as in the main ElateMe application database. The integer identifying attribute is *_id*, which is autogenerated in the main database, this field is required and nonupdatable, that means it is set on create action. The optional attributes represent the data received from users' Facebook accounts, at first logging in. So far Facebook gives the access only to *gender*(enum: M, F), *place of birth*(string) and *date of birth*(date) attributes to our application. So in future, this list could expand.

The *avg_rating*(float) field represents the average between the ratings that user has assigned to the particular products. This field is recounted dynamically on the runtime. The *ratings* field represents an array of users ratings, which consists of an *id*(integer) field - products identifier and *value*(float) field which is the value of rating, what was assigned in the result of the performance of a target action by the user.

The *soc_demo_groups* field is an array of identifiers(integers) of the socio-demographic groups the User was placed in, during users grouping.

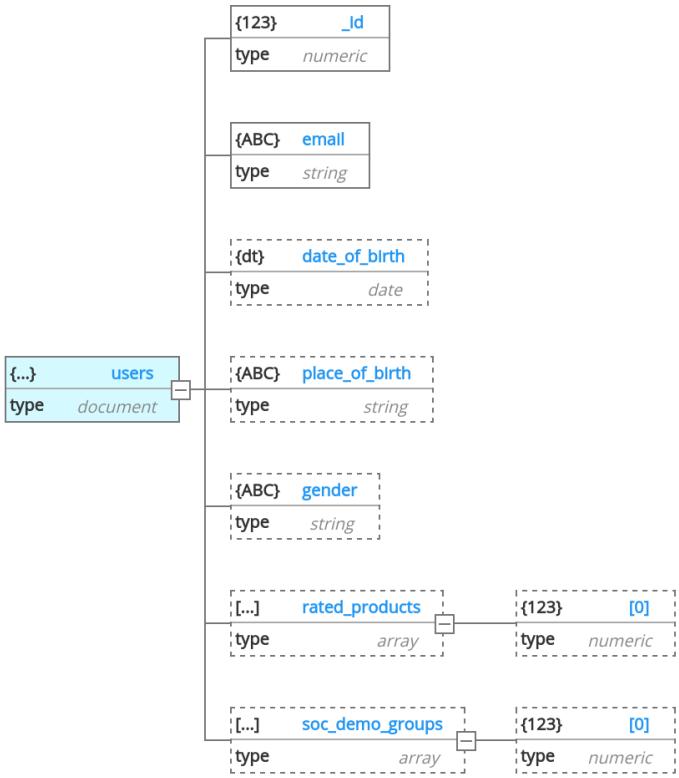


Figure 2.1: Users collection schema

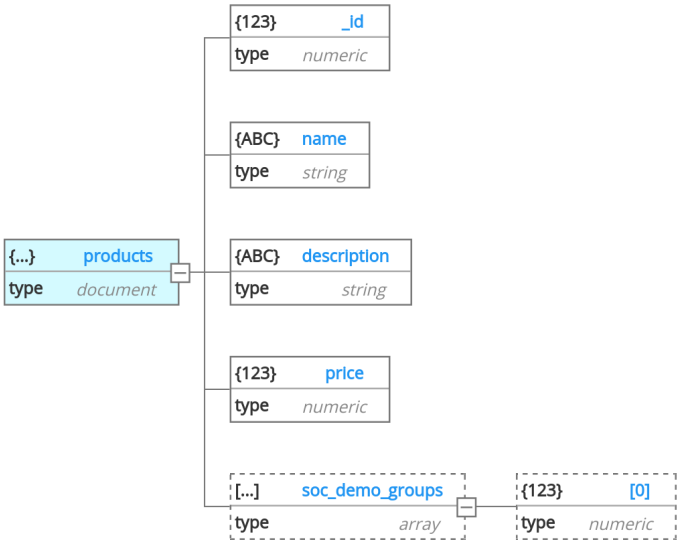


Figure 2.2: Products collection schema

Products collection

A Product in *products* collection is identified by autogenerated _id (integer) field. Also the *image*(string), *name*(string), *description*(string) and *price*(float) field are defined and they are required. All these attributes are set on products import action. The *soc_demo_groups* field is an array of socio-demographic groups identifiers a product was placed in. This field is intended to avoid the lack of recommendations problem and is set manually.

Socio-demographic groups collection

A socio-demographic group in *soc_demo_groups* collection is identified by autogenerated *_id*(integer) field. Also the group has got a *name* and a set of predefined *rules*. The rules are imposed on available user properties, for now they are *age* - an array of intervals with *from* (integer) and *to* (integer) attributes, *gender* - array of possible gender values, *place_of_birth* - an array of possible users native countries. These attributes are optional, but at least one of them has to be defined.

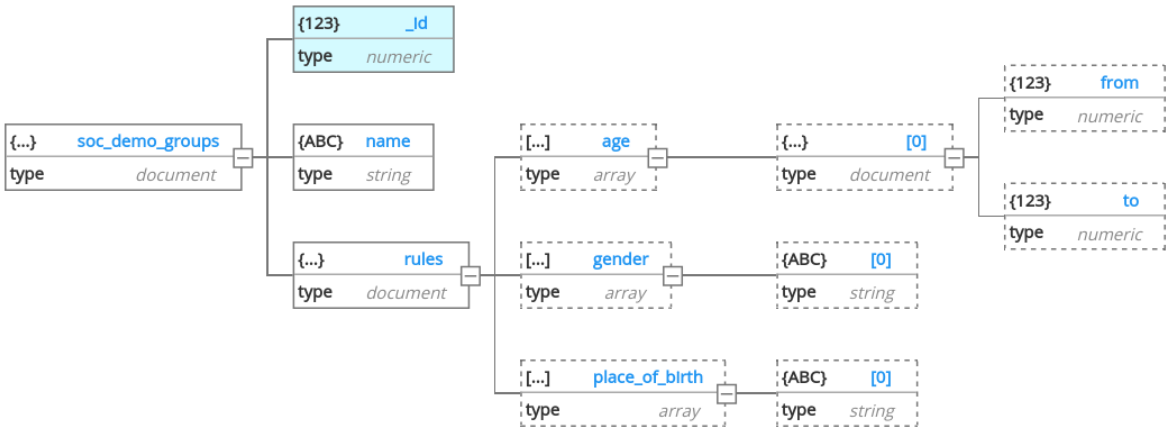


Figure 2.3: Socio-demographic groups collection schema

By the reason that the products are also to be used in recommendations with the search query, multiple text index for product *description* and *name* attributes was created. That is what MongoDB provides exactly for such purposes. Also one of the requirements from the project leader was to evaluate products and socio-demographic group identifiers in autogenerating sequenced format. MongoDB itself does not maintain such a mechanism, by default the *_id* value is generated as an ObjectId. Therefore, to simulate it, it was decided to use common approach, i.e. to create an additional collection of sequences for the collections identifiers - *seqs*. At the time of an insert action the client retrieves the related value of id from the seqs collection and after that increments it. This solution has to be changed in future because in a high loaded system it will cause significant inhibition of the system, native ObjectIds are recommended for use.

2.5 Advert server class model

2.5.1 Advisor class

Advisor class, in fact, implements the model, which is described in 2.3. The instance of this class is initialized for the particular user in *RecommendationHandler*, while the request is being processed. The only public method is *recommend*, which returns the list of the first *limit* recommendations ordered by the prediction value, where the *limit* is integer method parameter. And there are two auxilliary methods: *_predict* method calculates the prediction(float) for the user rating for the product(int), *_similarity* method calculates the similarity between the users using the Euclidean distance (float).

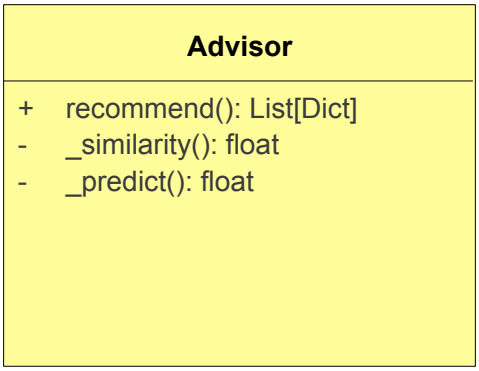


Figure 2.4: Advisor class

2.5.2 Request handlers

tornado.web module provides a simple web framework with asynchronous features that allow it to scale to large numbers of open connections. *RequestHandler* is base class for HTTP request handlers.[18] As Advert server prototype is going to be developed, it is desirable to generate user-friendly response errors. By default, Tornado responds with a simple Hypertext Markup Language (HTML) page with the only error code, when the *HTTPError* is raised. *RequestHandler* has special method *write_error* that is triggered when the *HTTPError* is raised. The author decided to create a base class for the ElateMe Advert server which is called *HttpRequestHandler*, which overrides the *write_error* method and responds with the error code and exception message in JSON format.

Tornado's *RequestHandler* provides methods to process requests with particular HTTP methods, their names are lowercase representation of used request HTTP method. By default, this methods raise *HTTPError* with status code 405 Method not allowed. As it is depicted on 2.6 all the Advert Server handlers implement at least one of these methods. **ListHandler* classes are

used to get the list of particular objects from the database on HTTP method GET or to create a new object in the database on HTTP method POST. **Handler* classes are used to get a standalone object by the identifier on HTTP method GET or to partially update the object in the database by the identifier on HTTP method PATCH, or to delete the object from the database by the identifier on HTTP method DELETE. The exception of such rules are only *RecommendationHandler* and *ActionHandler*, which will be described below. The specification of on request methods behavior was designed according to Representational state transfer (RESTful) specifications.[19] *RecommendationHandler* class is used to handle requests for recommendations for the user by the user identifier (integer) and it responds with the list of product, ordered by the predicted rating. *ActionHandler* class is used to handle the notification from the main ElateMe backend server about the commission of a target action (string) by the user (integer) on the product (integer).

2.5.3 Request body validation

Tornado does not support out-of-box[3] request body validation by the defined schema. Therefore, it was decided to design a *RequestValidator* class that will be responsible for the validation of XML and JSON documents according to the defined enum of JSON schemas (*JSONSchema*) and XML schemas (*XMLSchema*).

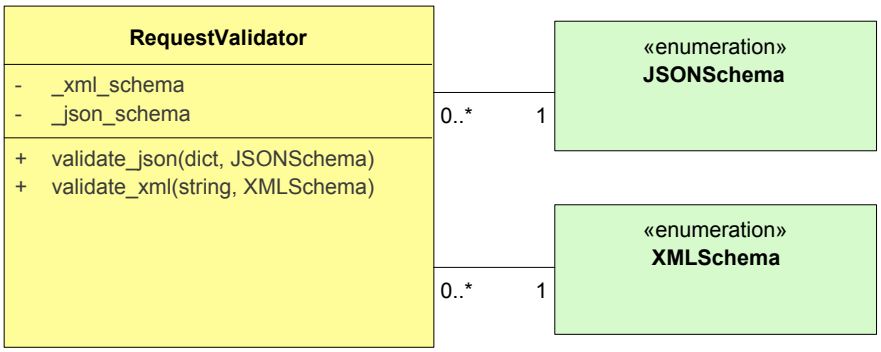


Figure 2.5: Validation class

[3]An out-of-the-box feature or functionality (also called OOTB or off the shelf), particularly in software, is a feature or functionality of a product that works immediately after installation without any configuration or modification.[20]

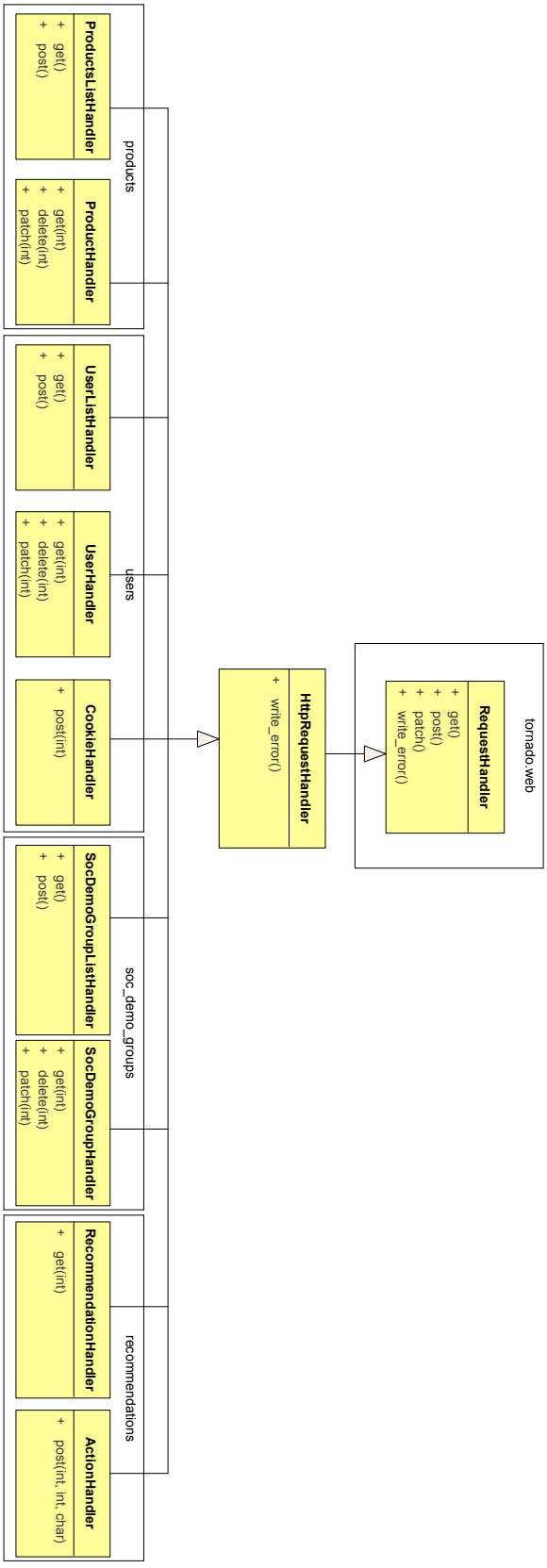


Figure 2.6: Handlers classes

Chapter 3

Implementation

3.1 Project structure

As the ElateMe Advert server is a long-term and potentially volume project, and in the framework of this thesis only the prototype of this application is going to be developed, it is very important to define concrete flexible project structure, that could be easily extended at a certain moment and in which the developer could get quickly oriented. Tornado documentation, unfortunately, does not provide project structure best practices. Therefore it was decided to use Python Django style of project structure.

The file launching the application is situated on the top level of Tornado project and is called *manage.py*, which is the analogue of *manage.py* file in Django.

The whole project is divided into modules called applications. The application serves one certain purpose, for example, user management or recommendations generating. The application is required to contain *urls.py* file, where Uniform Resource Locator (URL) configuration is specified and *handlers.py* file where the application handlers are situated.

On the top level of the project, the app directory is situated. This folder contains the *settings.py* file with the whole project configurations, such as database connection configuration, debug mode, and *urls.py* file where root URL specification is situated. In fact, the root URL specification is responsible for joining all URL specification of all the applications in one full-fledged URL configuration.

```
manage.py ..... file launching the application
app ..... project configurations directory
    settings.py ..... project main configuration
    urls.py ..... project root URL configuration
some_app ..... some_app application
    handlers.py ..... some_app handlers
    urls.py ..... some_app URL configuration
```

3.2 Deployment

The Advert server will be deployed on the Virtual Private Server (VPS) with Ubuntu 14.04 Operating System (OS). nginx on our server will serve as a HTTP proxy server and will be configured to run in secure mode. To enable Hypertext Transfer Protocol Secure (HTTPS) on the website you have to get a certificate from Certification Authority (CA). We have chosen Let's Encrypt CA to get the certificate from. There were two main reasons for that: the certificates are free, and the certificate renewal can be automated. Let's Encrypt also provides their own Software (SW) for certificate generation and maintenance, which is well documented, so it is another point in favor of it.

Tornado server will be launched on localhost and communicate with nginx and MongoDB.

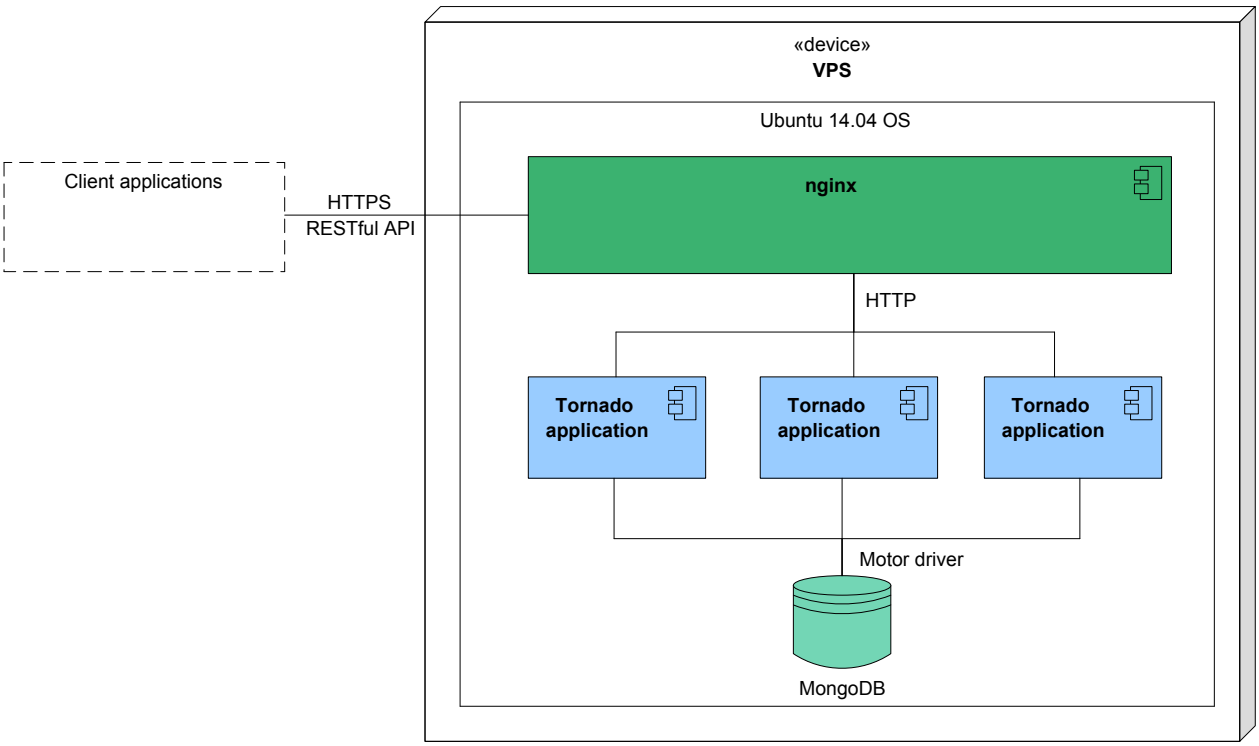


Figure 3.1: Deployment diagram

The communication with nginx will be conducted by HTTP protocol. nginx in ElateMe Advert server application will be used as a load balancer to distribute traffic to several application servers and to improve performance, scalability, and reliability of the applications.

The following mechanisms of load balancing are supported in nginx:

- **round-robin** — requests to the application servers are distributed in a round-robin fashion.
- **least-connected** — next request is assigned to the server with the least number of active connections.
- **ip-hash** — a hash-function is used to determine what server should be selected for the next request (based on the client's IP address).

In our application, *least-connected* mechanism of load balancing will be used.

The connection to MongoDB will be conducted by the *Motor driver*. The official Python driver for MongoDB is called *pymongo*, but its problem is that it is synchronous, which causes blocking the Tornado main thread every time it queries data from the database, which in turn breaks asynchrony of Tornado framework. *Motor driver* is an asynchronous alternative to *pymongo*. It provides practically the same interface as *pymongo* does for communication with a database. Motor does not support multithreading and forking, and was developed to be used in a single-threaded Tornado application.[21]

3.2.1 Automation deploy

Nowadays, unification and automation of different tasks in IT sphere are gaining popularity. This tendency in the author's opinion is very progressive and effective. The task of automation is especially effective in large projects, where several developers are working on the same application part. In the framework of our application, the task of deploy automating will be used.

3.2.1.1 Fabric

Fabric is a Python (2.5-2.7) library and command-line tool for streamlining the use of Secure Shell (SSH) for application deployment or systems administration tasks.[22] In our application Fabric is used for the server provisioning[4]. The server will be configured in the following way:

- Server dependencies will be updated and upgraded.
- Ansible dependencies will be installed.
- As the security measures root SSHing will be turned off, instead of it a new sudoer user deployer will be created. Password authentication will be turned off, instead of it, the SSH keys will be used.

[4]Server provisioning - initial server setting up and configuration to prepare it for using in a network.

- The SSH keys for deployer user will be generated and copied to the remote server.

Fabric was originally developed for Python (2.5-2.7), however, as Python 3.6 is used in the Advert server application, it was necessary to find an alternative, which is a fork of Fabric called Fabric3, whose main aim is to provide the compatibility with Python 3.4+ and to stay 100% compatible with the original Fabric.[23]

3.2.1.2 Ansible

Ansible is a modern and powerful tool that is developed for the routine tasks automation. As it was written above, Ansible will be used for deployment in our application. Let us describe the functioning principles of Ansible.

Ansible uses the SSH protocol to execute tasks on the remote server. One of the Ansible main features is idempotence of modules, meaning that if the task result has not changed since the last execution, Ansible will not change the current state of task-affected files and in such a way the task is executed effectively. The task always uses one of Ansible modules to be performed.

Tasks could be combined into playbooks, which are written in Yet Another Markup Language (YAML) format. As for the author's opinion, YAML is one of the most well-arranged formats of configuration because it does not contain syntactic garbage such as, for example, closing tags in XML interfering the perception ease. The formating in YAML is in a Python-like style using indentation levels. Therefore YAML is good for configuration files, but not effective for data transfer, because it could not be minified.

A playbook also has to define hosts to connect to and the remote user that will perform the tasks execution. Ansible also, maintains privileges escalation on a single task or for the whole playbook.

Ansible uses Jinja template syntax for template rendering or using defined variables in tasks. A variable could be defined as in the playbook so in the separate file.

Ansible playbook could have the defined handlers. Handlers are the list of tasks that could be notified by the notifying tasks if something has changed in the result of notifying task completion. The example of usage could be the update of a configuration file of nginx, which has to be followed by the nginx service restart to take effect. In this case, a handler could be the task of nginx service restart, and the notifying task is the one, that updates the configuration. So nginx service will be restarted if the configuration has changed, if not, nothing happens.

Here is an example of a simple playbook, that installs nginx on host 8.8.8.8:

```
- name: Simple playbook
  hosts: 8.8.8.8
  user: root
  tasks:
    - name: Install nginx
      apt: name=nginx state=latest
      become: yes
```

So what if we have a large number of tasks? A standalone playbook will be enormous and difficult to understand. For purposes of playbooks splitting, Ansible also maintains including files with the tasks or the tasks parts. It can be performed with just the include expression or by using roles. Roles are ways of automatically loading certain variables files, tasks, and handlers based on a known file structure. Ansible expects such project structure for using roles:

```
site.yml ..... some playbook
roles ..... roles directory
  common ..... role called common
    files ..... role files directory
    templates ..... role templates directory
    tasks ..... role tasks directory
    handlers ..... role handlers directory
    vars ..... role variables directory
```

The example of using roles with the illustrated project structure:

```
- name: Simple playbook
  hosts: 8.8.8.8
  user: root
  roles:
    - common
```

Hosts or the hosts groups could be placed in a separate file, which is called ‘inventory’. This file is written in the format similar to the .ini files.[17]

In the Advert server project, it was decided to carve up the process of deployment in such parts(roles), executed in the following order:

- **Provision**

  This step is not the same to the described in 3.2. If the initial server provision is used for the individual VPS settings update and installation of basic packages and dependencies, this role is responsible for the installation of additional dependencies, which were used during the development. Project git repository will also be cloned or updated(pulled) to a newer version in this role. Git is a version control system used in this project.

- **Database**

  This step is responsible for installation and configuration of MongoDB. After that the database structure and indexes are updated using the pre-written scripts.

- **nginx**

  This step is responsible for installation and configuration of nginx. Also, the configuration of our server is placed to enabled sites to turn on listening connections from network.

- **Web**

  This step begins with generating python virtual environment in the project or if it exists, the dependencies are updated according to the requirements. Virtual environment is a special tool, whose main aim is to isolate standalone project dependencies from globally installed dependencies in a python project to avoid version conflicts. Then the supervisor jobs are configured and launched for the project. Supervisor is a service which allows us to launch different processes called *programs* and control them(start, stop, restart) from a comfortable interface, which is called supervisorctl.

Chapter 4

Testing

4.1 Unit testing

During the development, the examination of the correct functioning of the particular parts of ElateMe Advert server application was performed by using unit tests. Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.[24]

Tornado framework uses class based unit testing, for HTTP servers the class *AsyncHTTPTestCase* is used. Every test case class should implement *get_app* method that has to return tornado application instance that has to be tested. Optionally you can override methods:

- *setUp* - method, which is called before each test method, may be used for needed for testing data initialization
- *tearDown* - method, which is called after each test method, may be used for cleanup.

Every test method name has to be started with *test_*.

While the development of the test cases the author got to the problem that in case of using the Motor connection to MongoDB like in an original application, the application and Motor used different IO loops, which indeed led to errors. To solve this problem the connection to MongoDB using Motor has to be established directly in the test case class.

It was decided to allot an abstract test case class for the Adver server application, which solves this problem and is called BaseHTTPTestCase:

```
from tornado.testing import AsyncHTTPTestCase
from tornado.ioloop import IOLoop
from tornado import web
from app.settings import settings


class BaseHTTPTestCase(AsyncHTTPTestCase):
    def get_app(self):
        import motor.motor_tornado
        client = motor.motor_tornado.MotorClient('localhost', 27017)
        settings['db'] = client.elatemedb
        return web.Application(**settings)

    def get_new_ioloop(self):
        return IOLoop.instance()
```

Here is an example of a simple products details test:

```
    class ProductsDetailsTests(BaseHTTPTestCase):

    def test_products_details(self):
        products_json_list = [
            {
                "image": "some.url/imgae.jpg",
                "name": "Iphone",
                "description": "Lorem",
                "price": 10000.0
            }
        ]

        response = self.fetch('/products/', method='POST',
                              body=json.dumps(products_json_list))
        print(response.body)
        resultlist = json.loads(response.body)

        for item in resultlist:
            product_id = int(item['_id'])
            response = self.fetch(f'/products/{product_id}/')
            print(response.body)
            product = json.loads(response.body)
            self.assertEquals(item['name'], product['name'])
            self.assertEqual(response.code, 200)
```

```
def test_user_does_not_exist(self):
    response = self.fetch('/products/0/')
    self.assertEquals(response.code, 404)
```

4.2 Performance tests

The performance testing of the Advert server application was conducted using ApacheBench(*ab*), which is a tool for benchmarking your HTTP server.

The main endpoints, in which performance is to be tested, are the common recommendation list generation and recommendations with the search query. *ab* has two main options that were used while testing: *-c concurrency*, meaning the number of multiple requests to perform at a time and *-n requests*, meaning the number of requests to perform for the benchmarking session. The ab output is the report about the performed test. The main metrics from the report that the project leader was interested in are the number of requests per second and the mean, across all concurrent requests, time per request.

The results that the author received with the settings described in 3.2 for 100 requests and 100 simultaneous connections are the following:

- Recommendation without a search query

  Requests per second: 100 req/s

  Time per request: 10.531 ms (mean, across all concurrent requests)

- Recommendation with a search query

  Requests per second: 500 req/s

  Time per request: 2.198 ms (mean, across all concurrent requests)

The project leader marked the results of this testing as acceptable.

Chapter 5

Managerial project evaluation

5.1 Risk assessment

ElateMe has been developed for more than a year, and during this process, different risks have arisen and disappeared. The author will describe only considerable risks that lasted the longest or are currently relevant.

The first one is the failure to meet the timeframe. The first reason of it is that the design of mobile application was constantly changing, which caused the delays in the mobile applications development. The second reason is that one of the developers of Android application left the project in winter 2017. However, the mobile developers worked well, and they were managed to fulfill the set tasks in their diplomas with some exceptions. So, to conclude, this risk was not justified, and now the functionality of iOS and Android application is in the same state.

The second arisen risk was that banks could decline to work with a crowdfunding platform. Initially, the project was planned to establish the payments system on Fio banka basis, but this bank did not respond to our request for more than a month. Therefore, it was decided to try to establish cooperation with the UniCredit Bank, which unequivocally declined to work with crowdfunding. After a long time, nevertheless, our project leader got to contact with the Fio banka manager and got the first pack of technical documentation, but not all the issues are settled yet. As the result payments will be implemented neither in ElateMe main server nor the mobile applications.

The last risk, which is a risk of all startup projects, is that people will not like the main idea of ElateMe and will not use our service. But on 2.05.2017 Usability testing was held by our project leader and a team of mobile developers, in which testers had to fill the questionnaire after testing. In this questionnaire one of the questions was "Would you use ElateMe in the future?" and 70% of participants had chosen "Maybe", 30% had chosen the option "Yes" and 0% had chosen the option "No". This data is not indicative, but it tells us that people will probably like the idea of the ElateMe applic-

ation. The details of these tests are described in the bachelor thesis of Gleb Arkhipov.[25]

5.2 Criteria of project success

The project was not directly defined by the budget, but by the time that is allocated for the bachelor thesis. It is apparent that with such a budget it is not possible to implement a high-quality application.

However, to be able to verify the success of the project, it is necessary to specify the primary objectives. This is not only a summary of application requirements but also ones that cover the entire project and concern not only the technical aspects of the project.

Despite the problems that have been described in 5.1, team members have submitted their final work. Developers successfully finished prototypes of the client and server applications and held the usability testing, where defined at the start of the project functionality was tested. The details of these tests are described in the bachelor thesis of Gleb Arkhipov.[25]

From a relationship point of view, the team was cooperative, and everyone tried to contribute and help with the smooth running of the project; no quarrels happened.

Last but not least, is an indication of the success of a positive evaluation in defending the bachelor thesis by individual team members. But this aspect will manifest later.

Conclusion

The author devoted a greater part of this thesis to the Advert server application and carried out the analysis of three possible solutions, the most appropriate of which was chosen. The selected mathematical model was adapted to the requirements, and the limitations of this model were compensated by integrating socio-demographic users grouping to the application. Based on that, database model and application architecture were designed. The implementation was performed in such a way as to make all the parts of applications reusable and easily extendable. Another argument for is that in this project the automation deploy scenarios are designed and implemented. The particular parts of the implemented prototype were tested using the unit tests, all of which finally passed successfully. Advert server performance was tested using ApacheBench, and the results were highlighted. All the aforementioned says that the Advert server part of the bachelor thesis tasks was successfully accomplished.

From the ElateMe point of view, the author analyzed relationships between the main entities in the aforementioned platform and business process of the wish lifecycle, which is a key flow of the application. The output of this analysis was used as an input for the analysis and design of different parts of the application, such as API server and mobile applications. The managerial evaluation of the project in the framework of this thesis was outlined only by the risk assessment and the criteria of the project success, which were briefly described and the analysis of these issues was supported by the facts and persuasive argumentation. So the ElateMe part of the bachelor thesis was successfully fulfilled.

Future outlook

The first thing that has to be mentioned is that the Advert server for the moment of writing is integrated neither with the API server nor with the mobile applications. The determining cause of it is that the developers, who

are responsible for these parts of the application did not have the task of such an integration in their tasks for the bachelor thesis. So the integration is to be done in the nearest future.

Also as it was described in 1.2.5, ElateMe application is not fully ready for the purposes of the correct recommendation system functioning. To resolve this problem before the release, it is very important to give the user an opportunity for a qualitative evaluation of the product; the possible improvements are described in 1.2.5.

The used recommendation algorithm was chosen on the basis of the experience of the other services, and this algorithm requires testing of the correctness of functioning after the release using real users' data. It is obvious that such algorithms require fine-tuning depending on the specificities of business; therefore the mathematical model will be probably corrected on-the-fly.

Bibliography

[1] Reference. Crowdfunding [online]. [Cited 2017-04-02]. Available from: http://whatis.techtarget.com/definition/crowdfunding

[2] Reference. The Statue of Liberty and America's crowdfunding pioneer [online]. [Cited 2017-04-02]. Available from: http://www.bbc.com/news/magazine-21932675

[3] Reference. A Brief History of Crowdfunding [online]. [Cited 2017-04-02]. Available from: http://www.freedman-chicago.com/ec4i/History-of-Crowdfunding.pdf

[4] FOWLER, Martin. *Patterns of Enterprise Application Architecture*. Addison Wesley, 2003.

[5] Reference. Business Process [online]. [Cited 2017-04-05]. Available from: http://www.pnmsoft.com/resources/bpm-tutorial/business-process/

[6] Reference. On-the-fly [online]. [Cited 2017-04-24]. Available from: http://whatis.techtarget.com/definition/on-the-fly

[7] Reference. Recommendation systems and factorization models [online]. [Cited 2017-04-10]. Available from: https://www.youtube.com/watch?v=Us4KJkJiYrM

[8] Reference. Weighted Arithmetic Mean [online]. [Cited 2017-04-10]. Available from: http://www.emathzone.com/tutorials/basic-statistics/weighted-arithmetic-mean.html

[9] Reference. Recommendation systems: user-based and item-based [online]. [Cited 2017-04-12]. Available from: http://www.emathzone.com/tutorials/basic-statistics/weighted-arithmetic-mean.html

[10] Reference. Review of Algorithms of data clustering [online]. [Cited 2017-04-10]. Available from: https://habrahabr.ru/post/101338/

[11] Reference. Coefficient of correlation [online]. [Cited 2017-04-15]. Available from: http://statistica.ru/theory/koeffitsient-korrelyatsii/

[12] Reference. The Python Tutorial [online]. [Cited 2017-04-23]. Available from: https://docs.python.org/3/tutorial/index.html

[13] ELMASTRI, Ramez. *Operating Systems*. Cram101 Textbook Reviews, 2014, ISBN 9781467277341.

[14] Reference. Tornado [online]. [Cited 2017-04-23]. Available from: http://www.tornadoweb.org/en/stable/

[15] Reference. The MongoDB 3.4 Manual [online]. [Cited 2017-04-10]. Available from: https://docs.mongodb.com/manual/

[16] Reference. nginx [online]. [Cited 2017-04-25]. Available from: https://nginx.org/en/

[17] Reference. Ansible Documentation [online]. [Cited 2017-05-02]. Available from: http://docs.ansible.com/ansible/index.html

[18] Reference. tornado.web — RequestHandler and Application classes [online]. [Cited 2017-05-01]. Available from: http://www.tornadoweb.org/en/stable/web.html

[19] Reference. Using HTTP Methods for RESTful Services [online]. [Cited 2017-05-01]. Available from: http://www.restapitutorial.com/lessons/httpmethods.html

[20] Reference. Out of the box [online]. [Cited 2017-04-28]. Available from: http://searchcio.techtarget.com/definition/out-of-the-box

[21] Reference. Differences between Motor and PyMongo [online]. [Cited 2017-05-02]. Available from: http://motor.readthedocs.io/en/stable/differences.html

[22] Reference. Welcome to Fabric! [online]. [Cited 2017-05-01]. Available from: http://www.fabfile.org/

[23] Reference. Fabric3 [online]. [Cited 2017-05-1]. Available from: https://pypi.python.org/pypi/Fabric3

[24] KOLAWA, Adam. *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Press, 2007, ISBN 0-470-04212-5.

[25] ARKHIPOV, Gleb. *ElateMe - iOS client II*. Czech Technical University in Prague, Faculty of Information Technology, 2017.

Appendix A

Acronyms

PC Personal computer

VHS Video Home System

DVD Digital Versatile Disc

ID Identifier

RMSE Root-Mean-Square Error

API Application Programming Interface

XML Extensible Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

TCP Transmission Control Protocol

UDP User Datagram Protocol

IT Information technology

NoSQL not only Sequence Query Language

JSON JavaScript Object Notation

HTML Hypertext Markup Language

RESTful Representational state transfer

URL Uniform Resource Locator

VPS Virtual Private Server

OS Operating System

CA Certification Authority

SW Software

SSH Secure Shell

YAML Yet Another Markup Language

Appendix B

Installation guide

Advert server installation guide on Ubuntu 14.04 OS. The project contains the same guide in the .md format.

B.1 Requirements

To be able to setup and run project you need:

- MongoDB 3.4
- Python 3.6
- pip
- virtualenv

B.2 Dependencies installation

MongoDB 3.4

```
sudo apt-key adv --keyserver
            hkp://keyserver.ubuntu.com:80
            --recv 0C49F3730359A14518585931BC711F9BA15703C6

echo "deb [ arch=amd64 ] http://repo.mongodb.org/apt/ubuntu
     trusty/mongodb-org/3.4 multiverse" |
     sudo tee /etc/apt/sources.list.d/mongodb-org-3.4.list

sudo apt-get update
sudo apt-get install mongodb-org
```

Python 3.6

```
sudo add-apt-repository ppa:fkrull/deadsnakes
sudo apt-get update
sudo apt-get install python3.6
```

pip

```
sudo apt-get install python3-pip
```

virtualenv

```
sudo pip install virtualenv
```

B.3 Setup

Syncronize database indexes and sequences:

```
cd mongo_scripts/
./syncdb.sh
```

Go to *app* folder and create virtual environment:

```
cd ../app
virtualenv -p /usr/bin/python3.6 venv
```

To begin using the virtual environment, it needs to be activated:

```
source venv/bin/activate
```

Install requirements inside virtual environment:

```
pip install -r requirements.txt
```

Now you should be able to run project locally:

```
./manage.py
```

or

```
./manage.py --port=<port_to_run_app_on>
```

Default port is 5555.
Server should be running on localhost:5555

Appendix C

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... the directory of source codes
  - `advertserver` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `BP_Balatsko_Maksym_2017.pdf` ..... the thesis text in PDF format