International Journal of Aquatic Science
ISSN: 2008-8019
Vol 12, Issue 02, 2021

Mobile Application Based Intelligent Role of Information Technologies in Tourism Sector

Kotsyuba Igor[1], Kudriashov Alexander[2], Galperin Mikhail[3], Agapova Catherine[4], Bishwajeet Pandey[5]

[1]Associate Professor, doctorate of Technical Sciences, Saint Petersburg national research university of information technologies, mechanics and optics, Russia
[2]student, Saint Petersburg national research university of information technologies, mechanics and optics, Russia
[3]student, Saint Petersburg national research university of information technologies, mechanics and optics, Russia
[4]student, Saint Petersburg national research university of information technologies, mechanics and optics, Russia
[5]Gyancity Research Consultancy, Motihari, India

E-mail: [1]igor.kotciuba@gmail.com, [2]225238@niuitmo.ru, [3]197317@niuitmo.ru, [4]katena.agapova.01@mail.ru, [5]gyancity@gyancity.com

Abstract
Purpose:
This paper presents the intellectual information system for personalized tourism consisting of the server app that serves data for the client mobile app.
Design/methodology/approach:
The proposed solution is implemented with full cycle stages in mind including data collection, data mining, information system's design, conceptual, logic and physical design as well as formatting, configuring data models for client apps and designing an acceptable user experience level of the mobile application and results of testing.
Findings:
Further deployment is connected with using new math models and patterns of system's realiza- tion. Presented intellectual system's use allows to significantly reduce costs at the stages of a complex of routes' constructing according to individual preferences, collecting all the necessary information of the route's subject and the ability to automatically take into account and correct the limits of time and financial resources.
Originality/value:
Decision-making math models for personalized travel routes' generation and optimization are presented. Both the theoretical explanations of decision-making models the development process are described.

Keywords: tourism, mobile applications in tourism, intelligent systems.

Paper Type: Research Paper

1. INTRODUCTION

Modern tourism is a booming trend characterized by appearance of numerous new technologies that are aimed to increase the attractiveness of places of interest. However, there is a different way, which implies developing a personalized approach to the creation of tourist routes. This is a result of a tourism global growth, as well as tourism economic potential and its influence on the development and improvement of public services in various regions.

Tourism in Russia is growing extensively due to country's expansion to the international tourist market and significant economic effect. It evolves both in big cities and in smaller regions. A lot of local cultural objects are being rediscovered thanks to the popularization of social and economic potential of Russian regions as well as to efforts made by regional scientific community which provides a lot of diverse strategies for calling in tourist flows to the local cultural centers.

One of the key directions of tourism developing is a reinforced individualization of tourist routes planning which is being implemented in a form of weakly structured indices of a tourist profile. This also leads to increasing Russian tourist routes competitive advantages. The most popular routes now are those which satisfy modern typical visiting programs requirements. Automatization tools provide notably valuable opportunities for tourism organization support. They help building individual tourist routes which take care of dynamically changing user preferences like system [1]. Thus, developing such a flexible tool for making tourist programs gains especial relevance level.

Background

The key problems that are currently being solved with e-Tourism technologies include: supporting up-to-date information updates about popular tourist destinations; providing helpful and valuable touristic guides [2]; hotels selection for long-stay city tourists [3] an opportunity to share the stories about places visited and tourist experience in general [4].

These days, the requirements related to the tourism information technology development mostly concern searching for a methodical, algorithmic, and software tools to create diverse and realistic personalized tourist routes [5]. However, it is still a challenge for tourists to create a route that combines multiple tourist destinations into one trip [6], considering financial, timely, and other limits simultaneously. The major difficulties also arise when choosing the route length and duration for both individual tourists and groups, especially when there is a conflict of interests between group members [5]. Moreover, the travel industry is oriented on analyzing the data to classify tourists on the base of the places of interest that they select and their travelling habits [7].

Another development direction of information technology in tourism is its technological aspect. Not only the content offered to a tourist has a meaningful value but also information technologies which allow big amounts of data markup and give the end user a convenient instrument for communication purposes. It is mentioned in work [8] that information quality metrics, source trustworthy level, interactivity and availability meanings positively affect the satisfaction level while deciding whether to go for a trip or not. Researches [9] postulate that information technologies in tourism have stopped being only a market tool and constantly transform into an instrument for making knowledgebase using new devices like smartphones, drones, wearable devices. This also happens due to new opportunities of connection and big data technologies such as UGL (user generated content), transaction information and device information (GPS, mobile roaming data, Bluetooth) [11].

Works dedicated to tourism digitalization should be marked separately. The work [12] describes advantages of e-WOM communication using positive relations discovery between informational social impact, normative social impact and entertaining content of self-expression. In the work [13] a concept model of recommendations in tourism is presented which is based on studying tourists needs (digital services and marketing, intellectual analysis of data and online community). The work [14] shows up the potential of immersion technologies such as augmented and virtual reality with heritage control in the tourist digital experience area through an integration of historical facts and alternative scenarios.

The possible solutions of the aforementioned problems include: Pareto optimization heuristics with non-dominated sorting [5]; multi-criteria optimization by popularity, cost and number of attractions [6]; route locations grouping using clusterization methods for daily subtasks search and taking users ratings and feedback into consideration [2]; genetic algorithm for variable neighborhood search and memetic search in differential evolution [5]; motivation-based routes matching with existing tourism topologies [7]; k-means clustering; trip buddy with recommendations based on user web surfing behavior [15] and other methods.

The usage of intellectual systems provides opportunities to combine user preferences with effective automated methods for complex computational tasks solving and provides better understanding of tourist behavior [16] to make decisions with higher quality and speed [12] Nevertheless, the search of new methods to solve a problem of personalized tourist routes creation has no complete solution, and research in this area remains relevant.

The development of a recommendation system that provides users with unique personalized walking routes on their mobile devices has the following goals:

• cover the majority of potential interests,

• represent cultural and historical information about Saint Petersburg in conformity with user preferences,

• achieve the advantage of being able to generate an endless number of routes containing all possible combinations of existing locations which have been picked and combined using the recommendation algorithm.

2. MATERIALS AND METHODS

Data Preparation Task

Every travel route generation system [17] requires a huge amount of specific formatted data that comes from a trustworthy origin. Currently, existing relevant data oftentimes dissatisfies the condition of clear standardization and full value of the information needed. At the same time, there are sources of data which were not used to parse the information from. With an example case of collecting list of people who id related to Saint Petersburg, the entire multilevel process of collecting, parsing and formatting data is analyzed.

The proposed solution includes a set of software instruments to work with Wikipedia that provides such utilities as data retrieval, data filtering, and data transformation to the format which is specified by the subject area. An introduction of scoring system is suggested to rank and rate the relevance of all the information located in Wikipedia categories. It is implied that each Wikipedia category (e.g. Poets, Musicians) is tagged by an expert group. For each tag the scoring systems calculates an integer value that is later used to determine the overall score of each data entity. An expert group also sets all minimal acceptable values for each tag in set which allows filtering low relevant or low contentful entities from the subject area perspective.

Steps of raw data processing:

• Narrow thematic categories exclusion.

• Scoring (calculating integer rating for each person) based on 7 criteria (size of Wikipedia page, languages count that this page is translated to, hyperlinks amount, amount of external Wikipedia services URLs, count of the word “Saint Petersburg” and its forms mentions in the text, amount of categories Wikipedia page belongs to, amount of notes - additional notes, links and footnotes).

• Filtering (removal) of 7-10% (on average) of people from the list tail (frequently these were "empty" pages with a few sentences in its description, empty main body of the page and all 7 criteria had the lowest values).

• Collecting all the information from the people left, which includes each person’s credentials, birth and death dates if applicable, short description.

• Normalization (conversion) of categories titles to the names of professions that these people were engaged in when possible (an example of non-convertible category is "graduates of some university").

Then a term of non-convertible category is introduced separately - this is the category with the title which cannot be converted into an occupation name. This category related work is necessary since it is crucial for the person data model to have the information about their profession as it can be used as a base filter when user requests are being processed. User may wish to walk around the places that are related to famous people of a specific profession or even to an exact person.

3. RESULTS VISUALIZATION

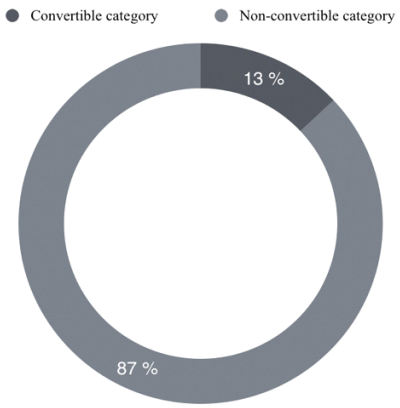

Figure 1. Convertible to non-convertible categories ratio

An average number of people in non-convertible categories as it is shown in Figure 1 exceed an average number of persons in convertible categories by 6.5 times. It is explained by the fact that non-convertible categories set contains the following titles “died in Saint Petersburg”, “awarded a Leningrad defence medal”, “born in the modern Saint Petersburg area”, “Leningrad blockade related people”, “teacher of Saint Petersburg universities”. The category ‘globalization’ is noticeable to relate frequently to geographical position, life cycle step (birth or date) or relation to a certain event.

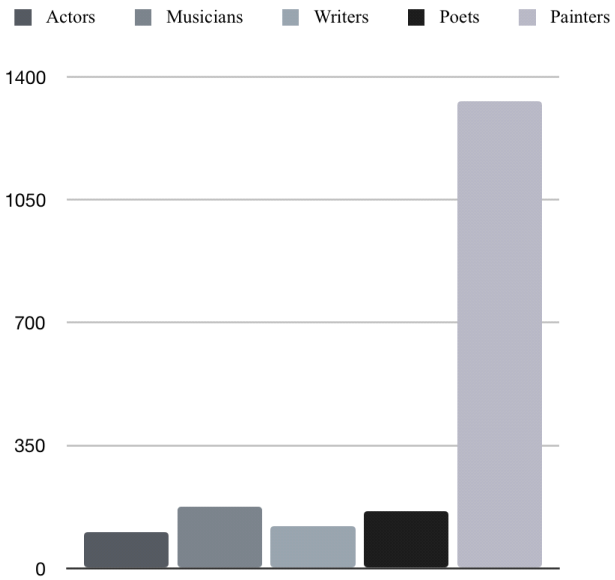

Figure 2. Distribution of creative professions representatives

Painters take the major part (Figure 2) because the students of Saint Petersburg Repin Institute and Art-Industrial Academy are highly recognizable. Another possible explanation of the fact that painters take the major part of the persons fetched is an availability to be visually engaged with the results of their work placed in various museums located in Saint Petersburg in comparison to musicians and poets whose works require a bit more conditional context to be consumed.

Since every data category on Wikipedia can have an endless number of subcategories, it becomes valuable to know which of them has the biggest subcategories set.

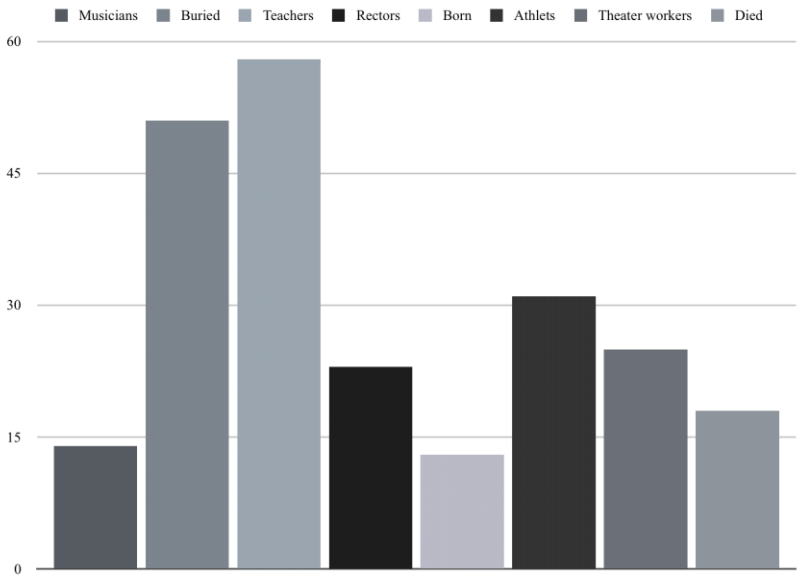

Figure 3. Distribution of persons' creative professions

The highest values are kept by categories "buried in Saint Petersburg" (nested level differentiation by the places of burial) and "teacher of Saint Petersburg universities" (Figure

3). All the rest slice of data stays inadvisable to split to more subcategories in the opposite to unite them in one list. While solving the task of persons' data preparation, 84000 entities have been collected.

After passing through the aforementioned scoring system, only 81k entities have been left. This persons' set was ready for usage in the recommendation system. The same data collection approach was used for "Saint Petersburg attractions" data category retrieval as well. Later the persons' set content was used as tags for the related locations.

A combination of attractions and people related to Saint Petersburg datasets is powerful in terms of providing user with the relevant content. To let them type persons' names and expect routes including locations related to these people to appear in these routes, a work of extracting relations between people and locations has been performed. Since each Wikipedia page has a set of categories it belongs to, it became possible to organize a cloud of tags for each entity collected by parsing these categories, formatting them and attaching to the database object.

Categories names contain words of any part of speech and only nouns were left as tag content to better match the typical user query like "Saint Petersburg of Pushkin". An array of categories names was split into separate words, and then short ones were removed (like particles and articles). After that a dictionary of each word frequency was calculated, one time appeared words were excluded, and the rest of future tags were passed through a pymorphy2 library to filter nouns. If possible, word forms were also converted to neutral representations to avoid the maternal or numeric declension.

Location-to-person relation was extracted from the location description text with the help of Named Entity Recognition and Natasha frameworks. Rule-based named entity recognition returned either full extracted persons' credentials or only initials with last names which was also enough to check if this person is presented in an existing persons' data set or is needed to be additionally fetched from Wikipedia as well. This extraction could also be contextual in case it provides the type of relationship. As a result, this led to the appearance of verifications concerning relationship extraction correction by comparing the context of the location and person by their categories. Even if categories comparison was impossible due to complete difference between contexts, it was still presumable that the location-to-person relationship was extracted correctly. To solve this kind of cases, an alternative way of checking relationship context has been made. There was a tool developed for that purpose and it had an interactive mode of verifying location context, person context and their relationship format. Thus, a human in a role of the operator could act as a verifier using that tool. This was a one-time need to finalize all works related to database content preparation.

The only data entities extracted from Wikipedia were Person and Attraction (place of interest). The designed usage of the system is intended to let the end user search both with the person they are interested in or specifying the type of the place to visit they wish to know more about. That is why collecting and storing location-to-person (and backwards) relations became such an important subtask of the data collection step.

Decision-Making Math Models

To formalize the math model of decision-making, first of all, it is necessary to calculate the score of people, tags and locations.

Person Score Calculation

Person score is calculated the following way:

Person.score = Page.page_size + Page.languages_count + Page.hyperlinks_count + Page.other_projects_links_count + Page.spb_stems_count + Page.categories_count + Page.notes_count

Where:

Page – source code of Wikipedia page (String)

page_size – length of the page content (Int)

languages_count – number of page translations (Int)

hyperlinks_count – number of hyperlink in page source (Int)

other_project_links_count – number of materials related to the person on Wikimedia foundation resources (Int)

spb_stems_count – number of "Saint Petersburg" and its wordforms occurrences in the page text (Int)

categories_count – number of Wikipedia categories in page source (Int)

notes_count – number of page notes (Int)

Tags Score Calculation

For a set of tags we built a graph, where vertices represent tags, and edges built upon the occurrence of a pair of tags in the same person page. Graph is undirected and weighted, edge weight represents number of tag pair occurrences in the whole set of people.

For each node we calculate betweenness centrality.

A node's betweenness centrality in a weighted network is given by the sum of the weights of its adjacent edges.

$$s_i = \sum_{j=1}^{N} a_{ij} w_{ij}$$

where:

a_{ij} – adjacency matrix between node i and j

w_{ij} – weight matrix between node i and j

The resulting betweenness centrality of a node counts as a tag score.

The example of graph subset visualization (1000 tags) is shown in Figure 4. Node color and size represent tag score scaled from 0 to 1. Edge weight is represented by thickness.

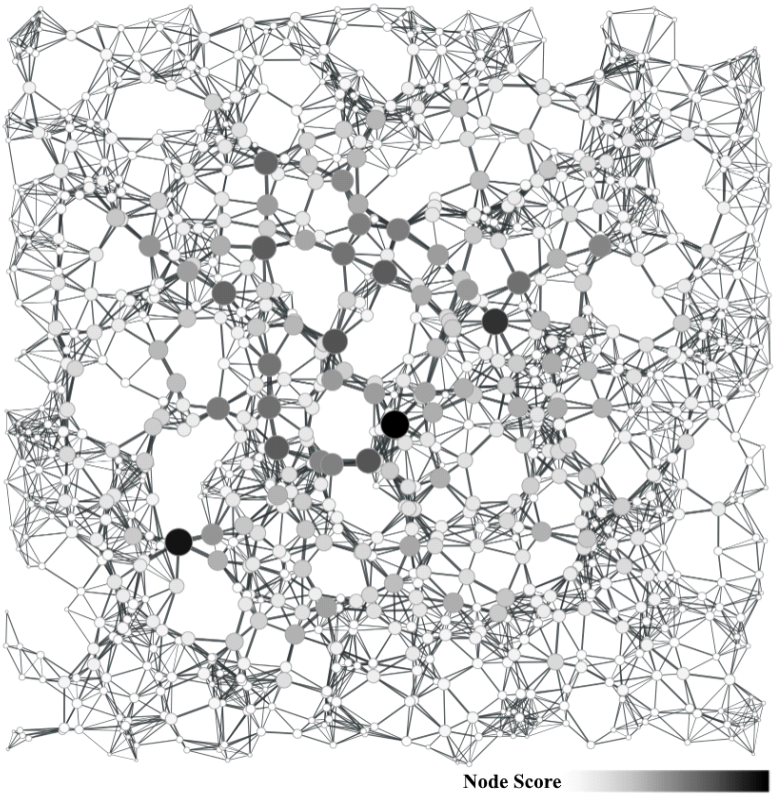

Figure 4. Visualization of tags subset

The part of this graph is demonstrated in Figure 5. Tag “19th-century Russian writers” denoted as node “1”, which has one of the biggest scores (0.92) among tags in this subset. Tag “Translators of Russian Empire” denoted as node “2” has half the value of 1st tag score (0.43). “Russian emigrants to France” (node “3”) has a score of 0.24. These values correlate to the relative popularity of tag appearance among people’s pages.

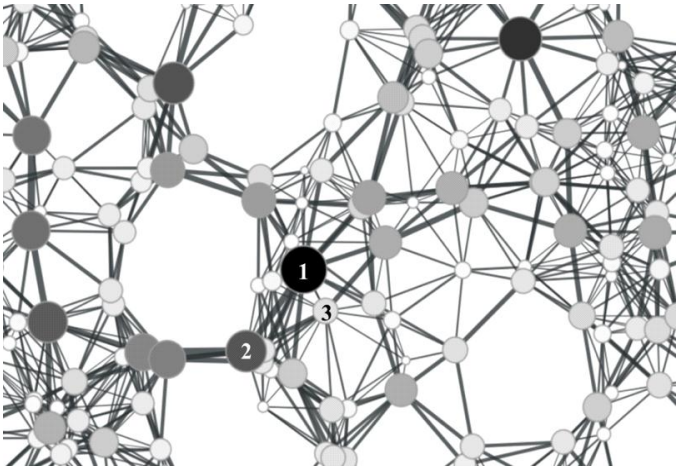

Figure 5. Partition of tags subset visualization

Locations Score calculation

In order to calculate the score of location, we take all tags and people related to this particular location and calculate the sum of their scores.

$$Location.score = \sum tag.score + \sum person.score$$

By using this approach, we would get peak values for cult locations (such as Hermitage). We discard n% of highest location scores (counting them as 1), the remaining scores are normalized in range from 0 to 1.

To construct an optimal tourist route, an optimization model from the discrete programming class is used. Location.Score was chosen as the objective function, calculated as the sum of Tags Score and Person Score according to the formulas given above. The limits are the general fund of time that the user is willing to spend on tourism (T) and the path size (Distance) that he is willing to walk:

$$f(x) = \sum_{i=1}^{m} Location.score_i \cdot x_i \to max$$

$$D = \{ x \in R^m | x_i \in [0,1] \forall\, i = \overline{1,m}$$

$$\sum_{i=1}^{m} distance_i \cdot x_i \leq Distance$$

$$\sum_{i=1}^{m} x_i \geq 2 \; ; \sum_{i=1}^{m} t_i x_i \leq T \}$$

$$x_i = \begin{cases} 1, & if\ the\ sight\ is\ in\ the\ optimal\ route \\ 0, & otherwise \end{cases}$$

To solve the optimization problem, Gomori method (the method of optimal cut-offs) was used, which has proven its effectiveness in solving discrete programming problems.The found solution will correspond to the optimal plan with the minimum number of route points (two), taking into account the formal restrictions on the maximum distance and time available to the tourist.

Routes Recommendation Algorithm

A routes recommendation algorithm is the core of modern travel-related applications. However, creation of such system is a challenging problem [18]. Many of the existing systems are oriented on self-drive tourism [20], while the proximity of locations in Saint Petersburg is better suited for walking tours. To address this issue, mining of user social media data [22,23], historical mobility records [24] and person-to-location relations is performed. Then resulting data is transformed into a set of tags for route personalization.

The proposed recommendation Algorithm 1 uses previously collected data from the database and user request as an input. Request contains the user coordinates, tags and duration in minutes. Combined with the information from the database (such as relations between locations and historical figures, list of possible activities in location etc.), it is possible to create virtually endless number of routes.

First, algorithm collects near expert-created routes based on their center of mass and tags specified. Then it uses the same user geolocation and tags to retrieve a list of nearest locations. The list of locations is further transformed into a list of optimal possible routes between each location; optimization is done by calling Google Maps API. Each expert route is transformed into subset of routes by excluding different locations to match duration expected. The next step is filtering both the expert and generated routes.

This operation involves the following procedures:

• Rating calculation for each route in the subsets of routes based on tags relevance and locations value.

• Searching for the most rated route in each subset.

• Exclusion of all routes in each subset excepting the most rated ones.

At the end of the pipeline, the algorithm returns optimal route for a given set of criteria.

Algorithm 1. suggest_route

Input: latitude, longitude, tags, duration, database

Output: optimal route

0: expert_routes = request list of routes made by experts from database that are near to latitude and longitude

1: nearest_tagged_locations = request list of locations from database containing specified tags that are near to latitude and longitude

2: distance_matrix = call Google Maps API to get distance matrix for nearest_tagged_locations

3: generated_routes = use distance_matrix and duration to generate routes for nearest_tagged_locations

4: expert_routes_subsets = generate subsets of routes by excluding locations to match duration

5: all_routes = expert_routes_subsets + generated_routes

6: for each route in all_routes

7: for each location in route

8: coefficient = compute intersection of user tags and location tags

9: add rating for route based on location value and coefficient

10: endfor

11: endfor

12: sort all_routes by decrease of rating

13: return first element of all_routes

All technical tools were developed using Python 3 programming language during these processes and were covered with unit tests that allow to be sure that the system works correctly at any moment - both in general and in particular (i.e. each submodule). That also allows to be adaptive to any changes on the Wikipedia side for the future similar data collection operations. Such libraries as requests and re were helpful as well since regular expressions are oftentimes a good fit for solving text-related tasks and network-related functionality was required only once. The idea of downloading all the source data to the working machine and then use it as long and variably as needed was introduced, discussed and implemented. Total amount of 15 GB of data was downloaded.

Thus, it should be mentioned that similar solutions are aimed at user mining based on data about the attractions they choose, their travel habits, data from social networks, which is then converted into a set of tags to personalize the route, to understand the behavior of tourists. In contrast to similar solutions, the developed methods and algorithm allow calculating locations score as the sum of person score and tags score and obtaining a tourist route in an optimization setting with restrictions on the time and distance funds - to generate the best solution from the range of admissible values.

Technical Realization

There are multiple ways to organize how the components of this system will be communicate with each other. In general, what should be understood under the term

'architecture' is principal system organization represented by its elements, their collaboration both among themselves and between them and the environment, and all the ideas which point out its design and evolution.

Potential candidates on the architecture role when it comes to organizing the system which has a server component in it are:

- Client-server
- Heavy client
- Three-tier architecture

Client-server is an architecture where a server operates as a massive logic item which handles general business requirements, hosts data of the entire system and does everything for client applications to just easily consume. In that case the client is just a software application that accepts user input and sends it to the server, basically it is just an interface for the user to communicate with the server. Client and server communicate via a network layer which is commonly represented by TCP/IP protocol. This architecture is scalable and can be improved in performance perspective, flexible and combinable with other architecture types on the server side. On the other hand, it brings the 'only one control center' problem which can be solved using decentralized parallel computations.

Heavy client is an architecture where client application does not depend on the server too much. On the contrary it contains a massive set of functionalities which only uses servers as a data storage entity. Client processes and represents all the information. Since every copy of a software with such type of architecture contains all the business logic it usually takes a lot of disk space to store and run such applications, but it is easier for the end user to sync their application state between multiple devices especially if such feature is supported on the system level.

Three-tier architecture is a sort of client-server architecture where a database has its individual separate place in the architecture components list. Client layer usually provides only graphic interface functionality to the user and does not contact with the data storage directly in any way. It does not execute business logic or store its own state because of reliability requirement. The middle layer of this architecture is the applications server layer. These applications are small pieces of executable logic which are also called terminals which can be delivered and interpreted by the client. The third layer is a data layer which stores and organizes the content of the system. Third layer can be accessed from the applications layer only, not from the client layer. In the very primitive cases all these layers can be placed on a one computational device.

In order to decide which architecture to give a preference to, there must be an understanding of which components does the routes system consist. Since expected use cases are based on mobile platforms such as iOS and Android, only those targets can be treated as clients of this system. In addition, there is no intent to store all the business logic in client applications and adapt their proper rendering to every platform where the system is currently or in the future will be used. This brings the team to the client-server architecture as on the first stage of project lifecycle there will no be a lot of user generated data to maintain, also a variety of the data operated will not be a quite extensive, there will be only one entity to apply business requirements on the go and mobile applications will be only in a consumer role in their relations with the server. Client-server architecture satisfies all these expectations.

It is represented by a web resource accessible via HTTP protocol that handles business logic and provides all the formatted data. The server satisfies a list of business requirements declared by the project needs and functionality set initially invented by project idea holders. A

server must be able to collect, store and modify routes, locations, and user data; process data provided by user and suggest either expert or generated routes if requested.

- The server should:
  - Store locations, routes and user data.
  - Serve routes and locations data to the client applications.
  - Generate on-demand personal routes based on user data.
  - Modify data in database based on user input, i.e. update location value.
- The following criteria must be complied. It has to:
  - Work robustly and consistently.
  - Generate optimal routes respecting relevance of each location and duration of travel time.
  - Have clear and well-documented API for consumer applications.
  - Provide database flexibility.

Backend stores the following data needed for route generation algorithm:

- Tags are content labels that represent relation to person or cultural category.
- Location Instances represent points of interest, containing coordinates, descriptive information, set of tags, and value.
- Routes are set of locations with description that stores locations consequently in the way they are offered for the user to follow.

Data model is demonstrated in Figure 6.

The Graphene-Django Framework is used for the backend core. It provides a set of utilities to quickly organize working environment and data storage, configure database and implement API endpoints that could be used by client applications.

While a database engine was being selected with the key priorities as the speed of deployment, readable and easy-to-use on a mobile side data formatting, scalability and a lack of strictly designed schema. MongoDB meets all these requirements since it stores data in JSON documents and uses inner memory, which ensures that all operations will be executed fast enough. Package Django-nonrel was used to add non-relational database support, and server engine was set to django_mongodb_engine.

The key endpoint that server has is the one that takes user parameters and returns best-fit routes back. All parameters are provided to the server in URL as GET parameters. There are data type, data value and data limits checks for the input parameters since requests can be received from not only a trustworthy origin and this is mostly about security whilst working with any data that comes from the user.

Client applications use GraphQL language to specify queries and data fields needed to be received.

The deployment is done via Docker container. The list of dependencies used for the codebase has a specified version for each dependency with an auto update disabled in order to work with only tested and secure versions of these frameworks. Every operation performed places a set of log lines in the logging journal that can be later analyzed in case of any bugs revealing, though no personal information is logged. For the server responses we use HTTP codes that indicate appropriate state of the request but no special format of the response text. Every new version of the back end functionality is fully covered with unit tests, and release checklist on deployment, environmental and product state is respectively performed. Since code version control is used during development, no other versions caching systems are required for the opportunity to roll any functionality back if needed.

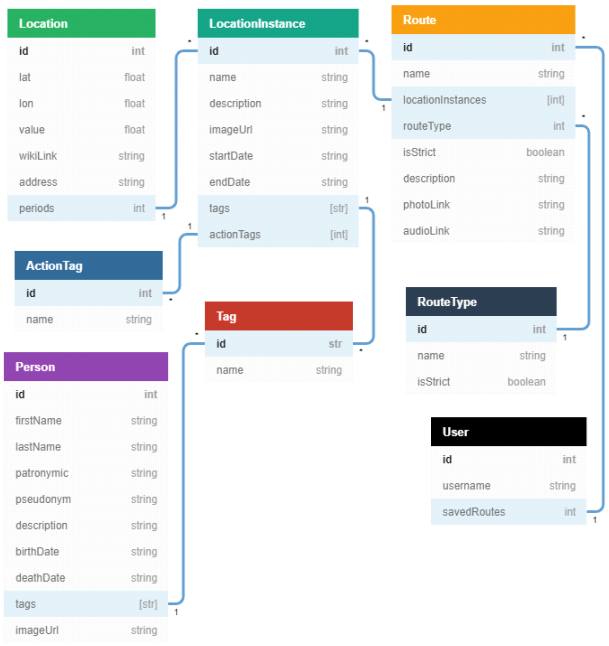

Figure 6. Data structure used by route generation algorithm

The codebase is manually typed whenever it is written with Python 3 which supports dynamic types and strictly linted before the code is pushed to the production version of the server. It allows several developers to write a readable and understandable code for everyone.

Mobile Application

A mobile application is required to provide recommendation and navigation mechanisms to the end users. Development process of the mobile application can be started once there is data collected and formatted, and a server with live endpoints that can be accessed by the application to execute the business logic.

In the app, an authorized user should have the possibility to:

- View the list of nearby routes.
- Get recommended and personalized routes.
- View the route details and all the points with their descriptions.
- View their personal location and route chosen on the map.
- Follow the route by going to each point.
- Finish the route and go back to the list of routes.

From the technical perspective, the app should use a single codebase for Android and iOS and be efficient enough to provide a normal user experience.

Moreover, the application should have a user friendly and intuitively understandable user interface (UI).

The following features were extracted from the initial business requirements:

- The app should:
  - Authorize (login/register) the user.
  - Display the list of nearby routes.

- Display the list of recommended routes based on a keyword and duration entered by the user.
- Display the chosen route description.
- Show the user location and connected route points on the map.
- Navigate the user on the route chosen.

• The app should comply with the following criteria:

- Work on both iOS and Android with a single codebase.
- Have a performance close to native (under the condition that the Internet connection on a device is stable).
- Follow the standard industrial UI design guides.

The mockups of all application screens (Figure 7-9) were developed according to the user journey through the app and using Google guidelines to Material Design. Material design was selected for mockups creation as it is easy to adapt to any other mobile system design requirements when it comes to serving each set of users with their preferred native interface experience they expect. Specifically, iOS Human Interface Guidelines have another point of view on how interfaces should look like because of another system representation in general and all the components that system interfaces are built with.

Android users subconsciously attribute a level of trust and security to a Material Design app because they associate the app with Google. Moreover, this concept is widely used in mobile and web apps today, and its elements have become habitual to the vast majority of users. Furthermore, there is a wide range of standard components such as icons, fonts, cards that can be used when developing the user interface for an app. Thus, the concept of Material Design was chosen to develop the UI side of the mobile app.

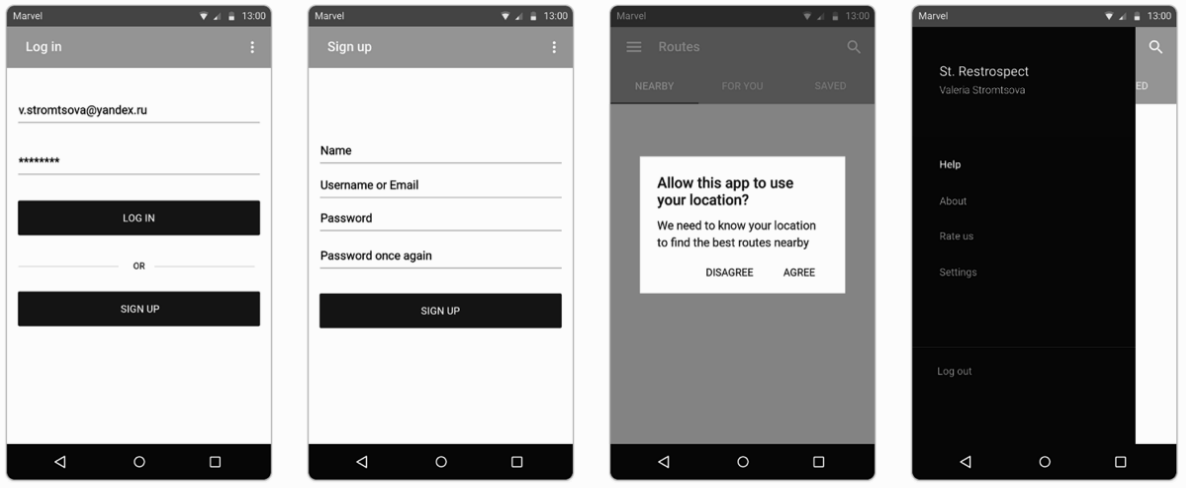

Figure 7. General mockups

International Journal of Aquatic Science
ISSN: 2008-8019
Vol 12, Issue 02, 2021

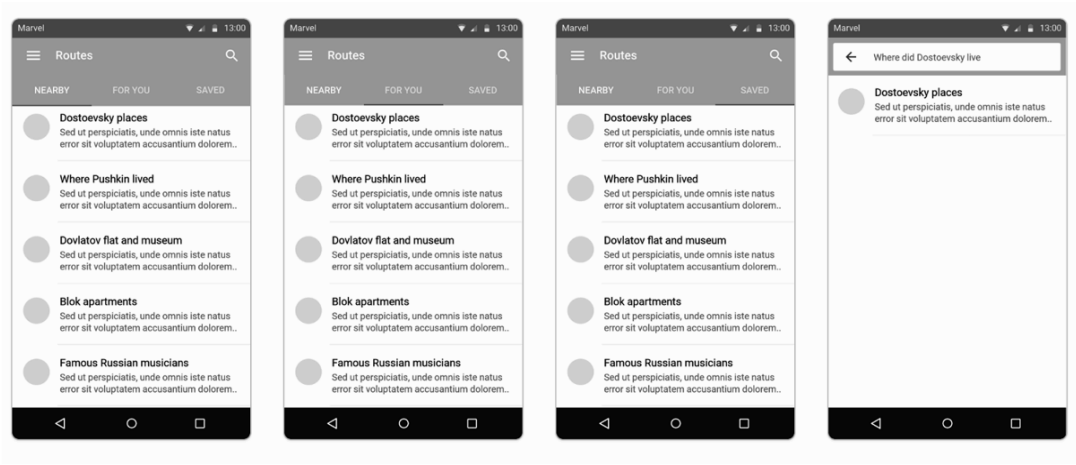

Figure 8. Routes lists

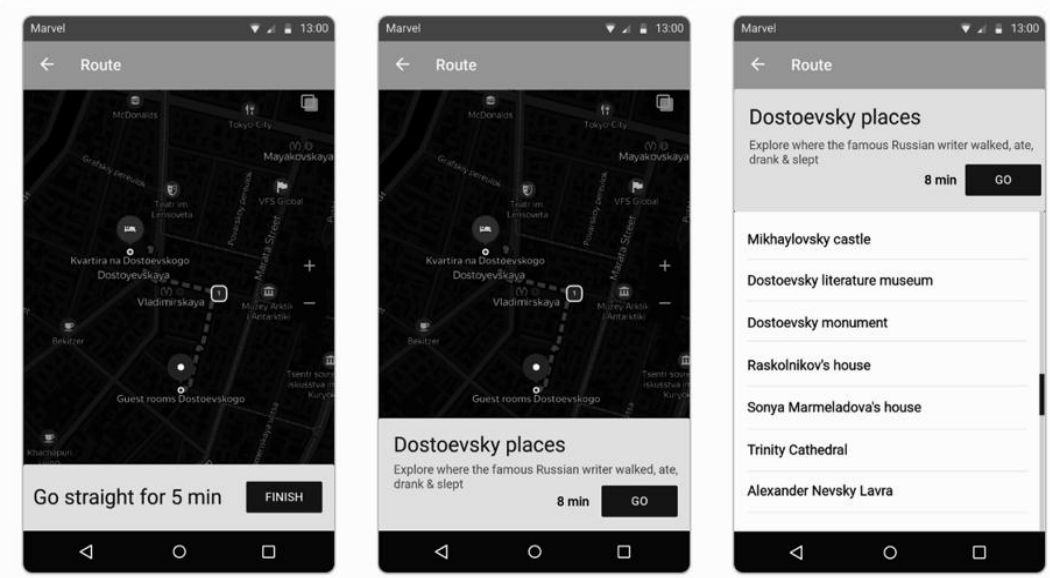

Figure 9. Route views

The architecture of the mobile app complies with the component approach, used as a standard in React Native applications. In addition, three principles of Redux are used to build the application. Three redux principles are defined by the technology itself as they form its foundation. First is that there must be only a single source of truth. Applications store their global states in an object tree which can be passed from the server as well and configure the initial application environment. Second principle is about an access mode of this global state: it is in read only mode. It cannot be changed directly. Different application modules can only express their intents to modify the state and since all tasks in the application are executed on a serial queue one by one this does not produce any memory issues. The third principle is that those intents to modify the state are called pure functions or reducers. They accept the previous app state as an input and return a new one after some action is completed. These reducers can be reused and separated by several application modules.

On a high level of abstraction, the application code is split into several parts:

• App, a single code base for both iOS and Android including:

• Components, UI elements that are used to display the information to the user and react to the user actions.

- Locales which provide localization for the whole application.
- Redux (with subcomponents actions, reducers, store) which is used to handle the business logic.
- Routing which provides the navigation between screens.
- Screens, that are meta-components combining different components to display on a single screen.
- Services to interact with the backend via API.
- Theme provides consistent styling (colors, fonts, paddings, etc.) for the whole app.
- Android, Android-specific settings.
- iOS, iOS-specific settings.
- Tests covering the App code.

There are dozens of instruments that can be used to make an information system. They are usually split into two categories: low level CASE tools and upper level CASE tools. Each of these categories has its advantages and weaknesses such as upper level CASE tools are very strict and not specific project loyal which means they do not take into consideration which pieces the project consists of rather than they dictate the evolution path for the entire project. On the other hand, low level CASE tools worsen teams' communication problems. Luckily there is an instrument that combines both the lower and upper tools methods which is Unified Modelling Language. That is very important in a client-server system to pay attention to the project lifecycle in general and to each part of it in particular during the development process. This modelling language usage provides such opportunities as conceptual model creation using business objects models, usage diagrams examples and sequence diagrams, logic system model designing with a class diagrams and state diagrams; and physical model production with deployment and components structural diagrams. All these aspects led to a copious UML usage in this route's system development process.

The component diagram for the mobile application can be seen in Figure 10. Mobile application consists of a set of screens user interacts with, architecture modules like redux and routings that are suitable for navigation, theme module to support easy customization of core colors of the app user interface, a bunch of services which are split into core services (database or networking related) and upper layer services (actions like adding route to favorites or leaving a review), and locales module that lets the app interface to be translated into several languages.

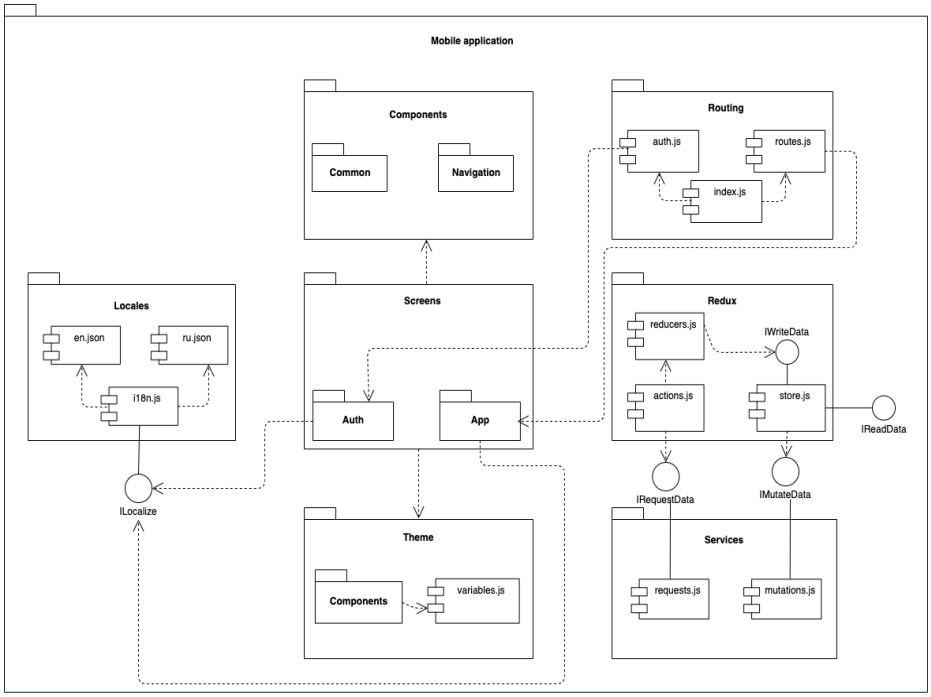

Figure 10. Mobile application component diagram

The activity diagram for the mobile application is demonstrated in Figure 11. User is prompted to share their timing limitations on how long the route should last. The cloud of their interest is also specified by a set of tags that describe routes, locations and people they are related to. After that, a route is generated on the server and returned back to the application. User is now able to start their walk through this route; their location will be tracked via smartphone GPS module, and information about locations included in the route will be changed on the screen accordingly.

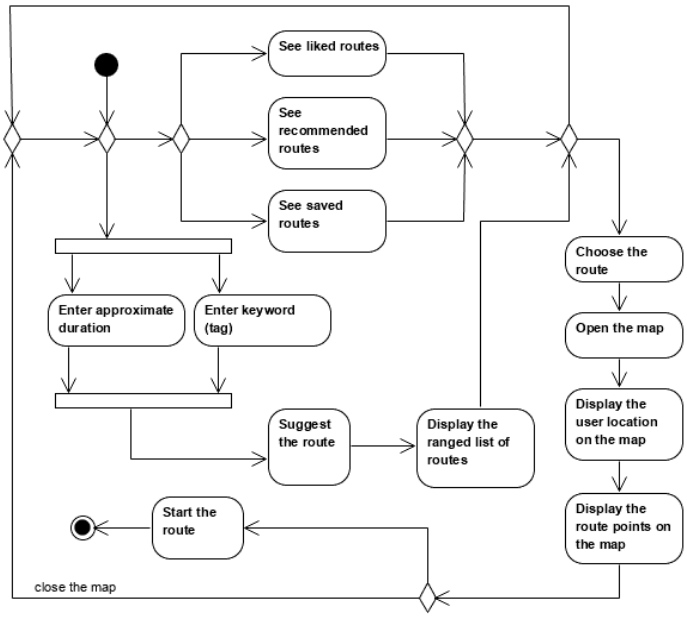

Figure 11. Mobile application activity diagram

There are also features like adding route to the list of favorites by liking them, checking out routes recommended and constructed by the group of experts and saving the route locally for the later use (walking them through) at any time in future.

The classes diagram for the mobile application is shown in Figure 12.

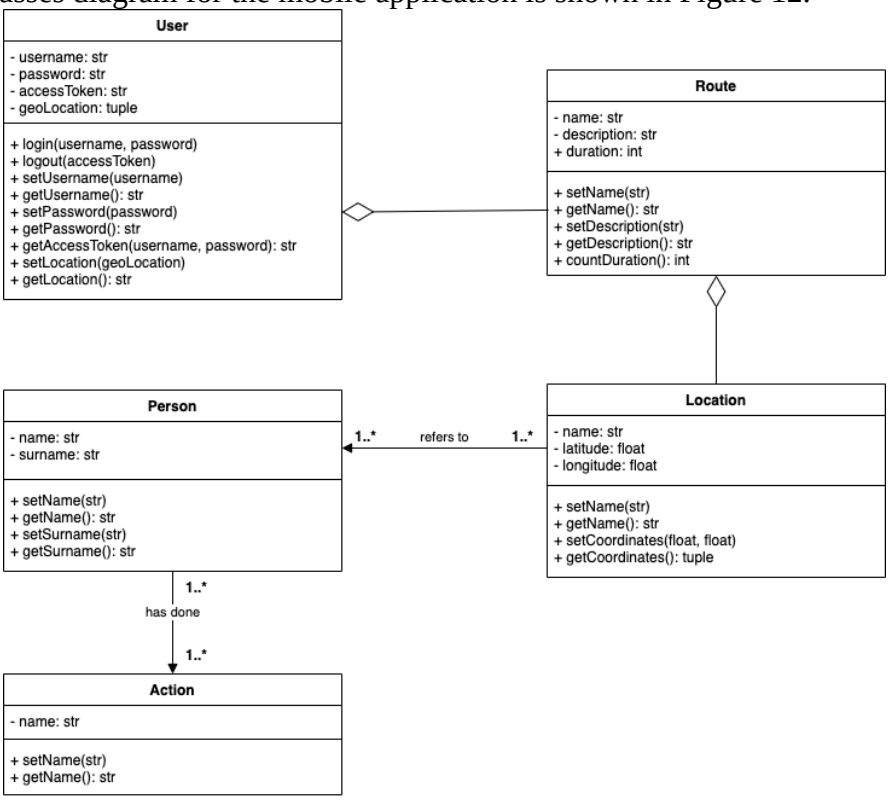

Figure 12. Mobile application classes diagram

Core system classes are User, Route, Location, Person and Action.

User class describes the basic set of properties that those who consume this system features have. They have credentials fields, accessToken and geoLocation properties which are constantly refreshed during mobile application usage. All the methods supported by User class are getting and setting each field. User class has no business logic methods. Location class describes some place in the city with its name and geolocation coordinates. Latitude and longitude are stored as separate float values but can also be accessed as a tuple. Route class describes a sequence of Location class instances. It stores information about its name and description as well as total duration in minutes that will take an average pedestrian to walk this route through. Person class is about famous human beings who lived in the past or are currently alive and it only keeps their credentials which are name and surname. Action class represents a description of any activity that can be performed in the specific location. This is needed to tell the user what they can do on some route that has this location in it.

When designing system models, their properties and methods, it is crucial to keep everything compliant with software engineering principles like single responsibility (which defines that every class must solve only one business logic problem), open-closed (forbidding classes to be modified but allowing them to be extended in its functionality), Liskov substitution (working with classes that inherit from other classes the same way as if they were their root classes), interface segregation (visual separation of features accessible to user by their

business needs) and dependency inversion (designing everything in the way classes depend on easily replaceable abstractions rather than concrete implementations) principles.

The data flow diagram for the mobile application is demonstrated in Figure 13.

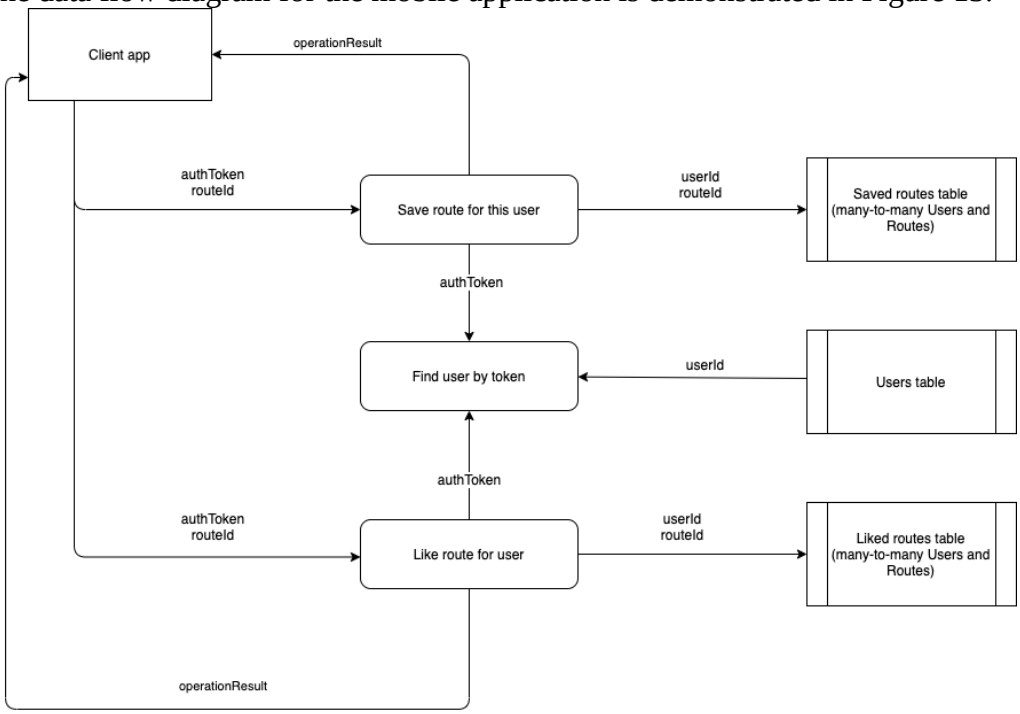

Figure 13. Mobile application data flow diagram

Mobile application uses token-based authentication system. Once user has been authenticated, a token string is generated for their session. Next, the user passes this token in the Authorization HTTP header for each request they make. Since every business logic method on server requires authentication, the first action the server does when receiving a request with token included is finding user by this token in the database and then proceeding to the actual execution of the request.

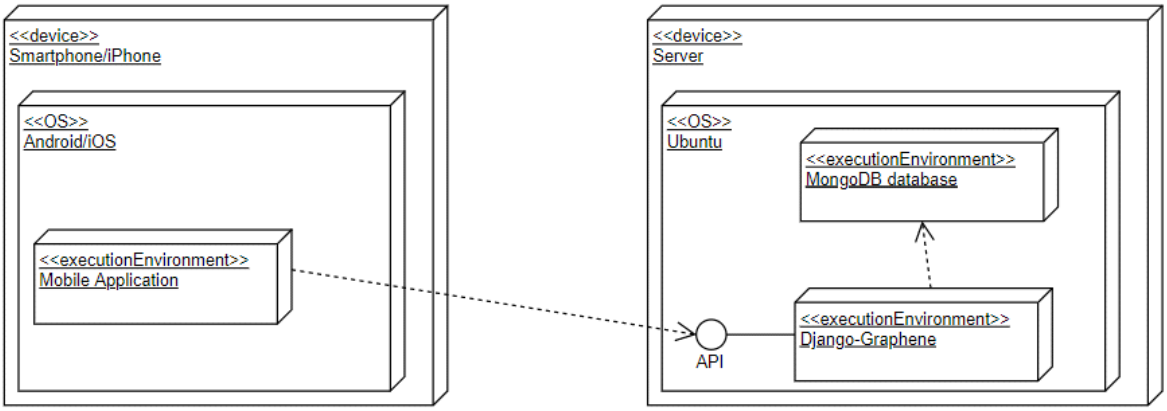

Figure 14. Mobile application deployment diagram

The application is deployed to smartphones running on Android or iOS operation systems (Figure 14). Each of them provides its own limits and rules on running the same tasks related even to the user location-tracking task.

Technology Stack

While choosing the technology stack, the following considerations were taken into account:

• A single codebase should be used for both iOS and Android operating systems (in other words, a cross-platform app should be developed).

• Efficiency, speed and reliability are crucial when it comes to navigating the user through the city following a specific route.

• The technologies should be relevant and supported by a large community. This is important for future maintenance and scalability of the app.

A comparison of frameworks that could be used for a mobile application development is shown in Table 1.

Table 1. Framework comparison

Framework / Criterion	**Apache Cordova**	**Xamarin**	**React Native**
Usage of platform-native programming languages	-	C, C++	Objective-C, Java
Supported platforms	Android, iOS, Windows, Blackberry, web	Android, iOS, Windows, Blackberry	Android, iOS
User interface development technologies	HTML5, CSS3, JavaScript, AngularJS, Ionic	HTML5, CSS3, JavaScript	HTML5, CSS3, JavaScript, React Native Components
Similarity to platform-native apps	Medium	High – compiled to native app	High – built as native app
Data model	WebView	WebView	Own data model

React Native was chosen as the tool to build the app. It allows using its own data model to avoid WebView [25] problems with performance and resource usage. React Native possesses a large community of developers and maintainers, a detailed documentation, a great number of tutorials, and good performance, which makes it an optimal tool to create the app.

Moreover, to implement fully the functionality of the app, the following libraries were used:

• Apollo GraphQL provides integration with backend API using the GraphQL query and mutation language.

• Redux is a widely used tool to manage an app global state with such concepts as actions, reducers and store.

• React-native-localize is used to translate the text on user interfaces depending on the locale that is set on the operating system level.

• Native base provides standard UI components and allows changing easily the application theme colors.

• React-navigation is used to provide the navigation between the application screens (views).

React-native-maps, react-native-geolocation-service, react-native-maps-directions provide an API to display the map on screen, set points (locations) on the map, display the current device location and calculate and display routes between given points.

Consequently, the choice of technology stack was made based on the various criteria, such as popularity and documentation, community size and support, and technical requirements (e.g. the application should be cross-platform).

The application was developed according to the aforementioned business requirements and features needed, as well as the architecture described above. It is always a huge must to analyze what has been done.

There are several groups of possible testing methods that can be performed. Basic classification consists of inner system understanding, an object of testing, a subject of testing, a testing time limit, a level of testing components isolation, a level of testing automation. User interface, performance, functional and component testing as well as end-to-end testing were selected for this case based on the system representation, its architecture and parts it is built on. Mobile applications must provide an especially acceptable level of positive user experience which means user interface validating took the majority of the entire testing timeline. There was a focus group of potential users organized and placed in a simulation of real application usage scenarios with every step performed being collected for the analytics purposes. The group was large and diverse according to the project scale.

After installing the application, the user is prompted to sign up or log in the app by either creating a username and password or using the existing ones. After signing up or logging in the app, the user is redirected to the app main screen. At this point, the application finds the device location and passes it to the backend in order to receive the nearest routes. Then, the routes are displayed on the screen in a list. Having chosen a route from the list, the user is redirected to the route detail screen. Here, the route title, description and points list can be seen. Furthermore, the current location and the route as a whole are displayed on the map (Figure 15-16).

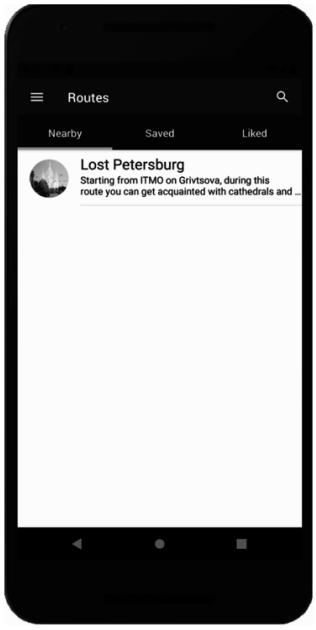

Figure 15. The main application screen

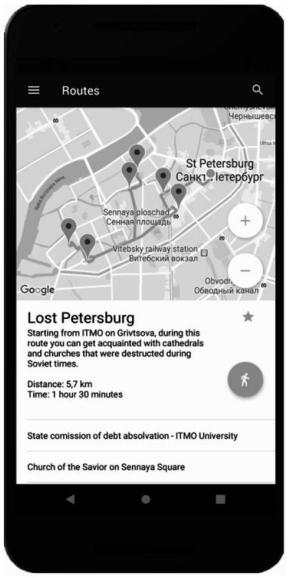

Figure 16. The main application screen

After tapping on an action button, the user is redirected to the route navigation screen where each location details can also be viewed. The device location on the map is updated in real time, and the next location shown to the user is updated when the user walks by the locations (Figure 17).

Figure 17. Route navigation screen and location details screen

The user has a possibility to get a list of recommended and personalized routes by entering a text query and/or setting the desired duration of the route [17].

4. RESULTS

Figure 18 is the approbation diagram that clearly shows that the majority of the users enjoyed how flexible can routes be built based on their individual preferences. The fact that a little more than a half of users marked building individual routes feature as an attractive one confirms that all the prepared routes were sufficient enough in terms of user's expectations. Tour guides provided were also marked as highly informative, captivating and enjoyable.

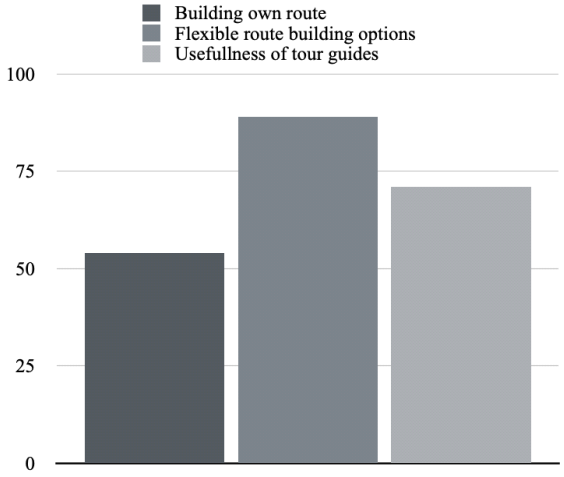

Figure 18. Approbation diagram

Figure 19 is the approbation diagram that shows the assessment results of the reduction in the intensity of tourist routes' development.

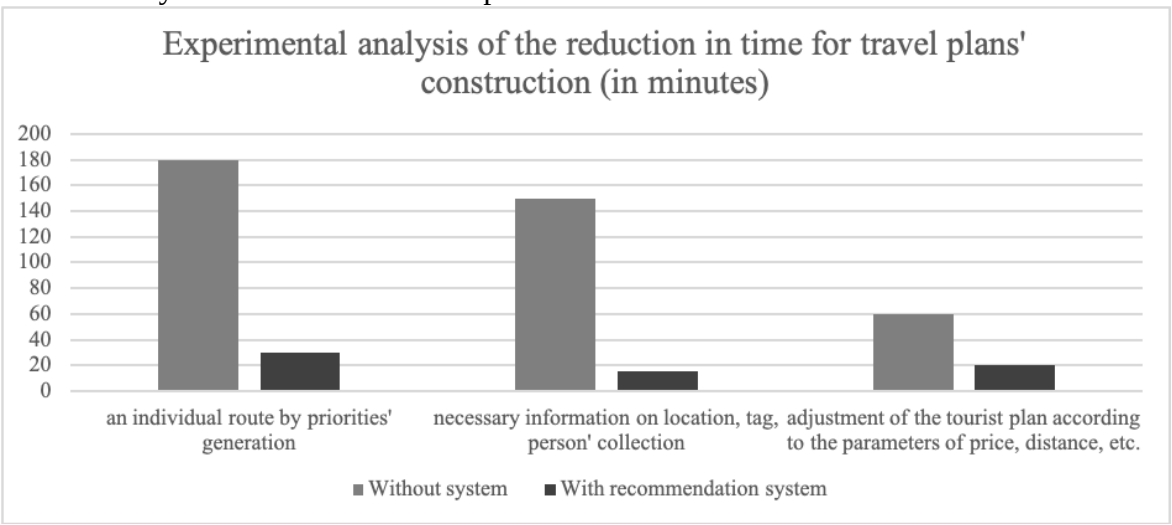

Figure 19. Testing diagram

It can be noticed that there is a significant reduction in time resources for the following important stages of tourist routes' construction: an individual route by priorities' generation; necessary information on location, tag, person' collection; adjustment of the tourist plan according to the parameters of price, distance, etc.

5. DISCUSSIONS

The new intellectual system for walking routes generation and recommendation has been proposed and implemented. Since the development process has covered everything starting from data collection to the user experience on mobile application designing, it was manageable and continuous. The system developed can provide an extensive set of possible routes for Saint Petersburg as it contains information about all attractions and famous places to visit and is able to generate routes based on user preferences. New models of decision support based on an optimization model from the class of discrete programming and graph theory are constructed.

This system takes durational, positional and culturological aspects into consideration to offer a route to the end-user and covers the majority of sets of preferences to avoid conflicts of interests while traveling in groups. The system aims to provide the optimal route possible by either creating completely new routes from the very beginning or modifying existing ones to fit the user requirements. Significant culturological patterns were revealed in the information categories associated with St. Petersburg, including the clear dominance of the categories "artists", "buried in St. Petersburg", "teacher of St. Petersburg universities" in comparison with other cultural categories.

6. CONCLUSION

The results of practical testing provided information that the use of a mobile application has advantages in such aspects as building individual route, flexible route building options, tour guides usefulness; the IT use allows to significantly reduce costs at the stages of a complex of routes' constructing according to individual preferences, collecting all the necessary information of the route's subject and the ability to automatically take into account and correct the limits of time and financial resources.

The next stage in the system's development is tagging system's improving, a mathematical model's developing by introducing new metrics on the tag graph and introducing multi-criteria optimization.

7. REFERENCES

[1] Khlopotov, I. Kotciuba, V. Stromtcova, A. Kudriashov, M. Galperin, "Personalized Travel Routes Generation for Mobile Application", In. *Proceedings of the 26th Conference of Open Innovations Association FRUCT*, 2020, pp. 530-538

[2] Cenamor, T. de la Rosa, S. Núñez, D. Borrajo, "Planning for tourism routes using social networks", *Expert Systems with Applications*, vol. 69, March 2017, pp. 1-9.

[3] Zheng, H. Ji, C. Lin, W. Wang, B. Yu, "Using a heuristic approach to design personalized urban tourism itineraries with hotel selection", *Tourism Management*, vol. 76, Feb. 2020, pp. 103956.

[4] Bassano, S. Barile, P. Piciocchi, J.C. Spohrer, F. Iandolo, R. Fisk, "Storytelling about places: Tourism marketing in the digital age", *Cities*, vol. 87, Apr. 2019, pp. 10-20.

[5] Zheng, Z. Liao, "Using a heuristic approach to design personalized tour routes for heterogeneous tourist groups", *Tourism Management*, vol. 72, June 2019, pp. 313-325.

[6] Uwaisy, Z.K.A. Baizal, M.Y. Reditya, "Recommendation of Scheduling Tourism Routes using Tabu Search Method (Case Study Bandung)", *Procedia Computer Science*, vol. 157, 2019, pp. 150-159.

[7] Ma, A.P. Kirilenko, S. Stepchenkova, "Special interest tourism is not so special after all: Big data evidence from the 2017 Great American Solar Eclipse", *Tourism Management*, vol. 77, Apr. 2020, pp. 104021.
[8] W. Yoo, J. Goo, C.D. Huang, K. Nam, M. Woo, "Improving travel decision support satisfaction with smart tourism technologies: A framework of tourist elaboration likelihood and self-efficacy", *Technological Forecasting and Social Change*, vol. 123, 2017, pp. 330-341.
[9] Xiang, "From digitization to the age of acceleration: On information technology and tourism", *Tourism Management Perspectives*, vol. 25, 2018, pp. 147-150.
[10] X.F. Fan, D. Buhalis, B. Lin, "A tourist typology of online and face-to-face social contact: Destination immersion and tourism encapsulation/decapsulation", *Annals of Tourism Research*, vol. 78, 2019, pp. 102757.
[11] Li, L. Xu, L. Tang, S. Wang, L. Li, "Big data in tourism research: A literature review", *Tourism Management*, vol. 68, 2018, pp. 301-323.
[12] Bu, J. Parkinson, P. Thaichon, "Digital content marketing as a catalyst for e-WOM in food tourism", *AMJ*, 2020.
[13] Reichstein, R.-C. Harting, "Potentials of changing customer needs in a digital world – a conceptual model and recommendations for action in tourism", *Procedia Computer Science*, vol. 126, 2018, pp. 1484-1494.
[14] Bec, B. Moyle, K. Timms, V. Schaffer, L. Skavronskaya, C. Little, "Management of immersive heritage tourism experiences: A conceptual model", *Tourism Management*, vol. 72, 2019, pp. 117-120.
[15] Sumardi, R. Wongso, F.A. Luwinda, ""TripBuddy" Travel Planner with Recommendation based on User's Browsing Behaviour", *Procedia Computer Science*, vol. 116, 2017, pp. 326-333.
[16] Boulaalam, B. Aghoutane, D. El Ouadghiri, A. Moumen, M. Malinine, "Proposal of a Big data System Based on the Recommendation and Profiling Techniques for an Intelligent Management of Moroccan Tourism", *Procedia Computer Science*, vol. 134, 2018, pp. 346-351.
[17] Bin, T. Gu, Y. Sun, L. Chang, L. Sun, "A Travel Route Recommendation System Based on Smart Phones and IoT Environment", *Wireless Communications and Mobile Computing*, vol. 2019, doi: https://doi.org/10.1155/2019/7038259.
[18] Xu, T. Hu, Y. Li, "A travel route recommendation algorithm with personal preference", In: *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2016, doi**:** 10.1109/FSKD.2016.7603205.
[19] Cui, J. Luo, X. Wang, "Personalized travel route recommendation using collaborative filtering based on GPS trajectories", *International Journal of Digital Earth*, vol. 11, 2018, doi: https://doi.org/10.1080/17538947.2017.1326535.
[20] Liu, J. Xu, S. Shaoyi Liao, H. Chen, "A real-time personalized route recommendation system for self-drive tourists based on vehicle to vehicle communication", *Expert Systems and Applications*, vol. 41 (7), 2014, pp. 3409-3417.
[21] Ashokkumar, N. Arunkumar, S. Don, "Intelligent optimal route recommendation among heterogeneous objects with keywords", *Computers&Electrical Engineering*, vol. 68, 2018, pp. 526-535.
[22] B. Chandanshiv, A. Nawathe, "Travel Route Recommendation by Mining Travelogues and Community Contributed

Photos using Cosine Similarity", *International Journal for Scientific Research&Development*, vol. 5, 2017, pp. 1500-1504.

[23] **Kotciuba, I., Shikov, A., Vlasov, R. The concept of organizing educational literary tourism by means of information technologies as a region development tool. *E3S Web of Conferences*, 2020, 208, 05003**

[24] Ting Wen, J. Yeo, W. Chih Peng, S. Hwang, "Efficient Keyword-Aware Representative Travel Route Recommendation", *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, 2017, pp. 1639-1652.

[25] Gavalas, M. Kenteris, C. Konstantopoulos, G. Pantziou, "Web application for recommending personalised mobile tourist routes", *IET Software*, vol. 6, 2012, pp. 313-322.