ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 7, No. 1, 2022

SOFTWARE SYSTEM FOR MONITORING SITUATION IN THE SETTLEMENT

Serhii Kundys[1], Bohdan Havano[1], Mykola Morozov[2]

[1]Lviv Polytechnic National University, 12, Bandera Str., Lviv, 79013, Ukraine.
[2]Technical University of Munich, Boltzmannstr. 3, Garching b. Munich, 85748, Germany
Authors' e-mail*: serhii.kundys.ki.2018@lpnu.ua; bohdan.i.havano@lpnu.ua; mcmikecreations@gmail.com*

https://doi.org/10.23939/acps2022.__.___

Submitted on 23.05.2022

Abstract: The goal of the work is to develop the software system of monitoring of a situation in the settlement. It consists of the user interface which is presented as a mobile application and the server. This paper describes the process of developing a client-server software system in stages using the latest technologies which will be relevant and easy to maintain in the future. The technologies used in the development process, the systems and modules which were integrated into the project, the main approaches to software development, as well as an explanation of why this particular stack of technologies was preferred for the implementation of this software system have been described. To make sure that developed mobile application meets common optimization requirements it has been tested for resources usage.

***Index Terms*: client-server architecture, mobile development, software system, database, JavaScript, Node JS, React Native, Firebase.**

INTRODUCTION

Today, as it has been never before, our society must take care of the organization of its security, the security of its locality, because the health and even the lives of its inhabitants depend on it. It is the most important component of the life of each individual, because without it everything else is impossible.

The urgency of the topic of security of the settlement is to take certain steps, some progress in ensuring safe living conditions in a certain area, within a certain settlement. To inform and protect anyone who is at risk of becoming a participant or victim of an emergency in their locality when any threat to human health or life is identified. And to improve this process, to allow ordinary residents of a settlement to participate directly in the process of protecting security and improving the well-being of their region, it was planned to create a system for monitoring the situation in the settlement. This is the main goal, the primary problem which is solved in the software system of this work.

A software system [1] is a group of specific software integrated tools and components that meet the needs of a specific group of users of a given software system, which are in common using database. Today's software systems due to their flexibility in implementation are able to meet any needs of users.

Mobile application [2] is a software that runs on platforms that use mobile operating systems. At the beginning of its existence, mobile applications had a narrow range of applications, as they were intended for everyday use. Examples of such applications are alarm clock, calculator, stopwatch, timer, calendar and others. Typically, such applications do not even require access to the World Wide Web [3]. Later, more sophisticated applications appeared that allowed the user to check e-mail, search for information on the Internet, use the browser, listen to music, etc. A major breakthrough in this area were Google applications such as Google Chrome, Gmail, Google Calendar, Google Drive, Google Maps and more.

To date, the leader of the most common mobile applications in Ukraine is Google Chrome, it is installed on 98.2 % of mobile devices running the Android operating system. YouTube ranks second with 97.2 %, Viber (96.8 %), Gmail (95.8 %) and Google Maps with 86.0 %.

Mobile operating system [4] is an operating system designed for use on mobile platforms. The most common devices on this platform are smartphones, less often - tablets and some other mobile devices. Such systems combine all the necessary functionality of a mobile phone, some functionality of personal computer operating systems, as well as various additional functions for everyday use such as Bluetooth, the ability to connect to a Wifi network and others.

The most common mobile operating systems today are Android and iOS. As of the first quarter of 2018, 383 million smartphones were sold worldwide. Most of them, namely 86.2 % use the Android operating system. The second place is occupied by iOS with a rate of 12.9 %. An important achievement of Android is that it is the most popular operating system in the world regardless of platforms. As of April 2020, Android occupies 39.77 % of all operating systems that currently exist. The second is Windows with a rate of 32.31 %, and the third is iOS, whose share is 17.66 percent. Looking at these statistics, we can safely say that Mobile Operating Systems play a huge role in the life of each of us.

Client-Server Architecture [5] is a software systems architecture in which, as you might guess, two parties are involved: the client side and the server side. Part of the resources in this architecture is focused on the server part, which usually serves the client part. Today, this approach to software development is one of the most common, because with the development of World Wide Web [3], more and more systems in some form communicate with each other or any other third-party services using the Internet. So, in today's reality, the communication of client and server parts plays a key role in the execution of software.

From the point of view of software development, this architecture is considered very convenient to implement, that is the reason why more and more software developers are using this approach for development purpose.

Server [6] is one of the components of the client-server architecture, which is usually responsible for processing data received from the client and also serves to host various services which a system interacts with. It often happens that to unload the client part of the system, a part of its functionality is transferred to the server side. This significantly improves the performance of the client part, which has a good effect on the experience of using this system by users, as well as the quality of user interaction with it.

Each server located on the Internet has its own address. Inquiries from the client part are usually made to this address. In response to these requests, the server "provides a service" to the objects that access it.

Client [7] is one of the two main components of the Client-Server Architecture [5], which provides the user interface to interact with a service, as well as uses server resources. The Client side of the system is usually presented as a web, mobile or desktop application. Usually this application is the only way for the end user to communicate with the system. There are also combined Client applications that allow the user to interact with the service on multiple platforms simultaneously. For example, Google services, which typically provide the user with a web interface for using services in the browser, as well as mobile applications for using them on mobile platforms.

An important aspect in the development of the Client part of the Software System [1] is the development of the software interface. It is important that the end user can easily understand the program, navigate the basic functions of the program, as well as efficiently, without difficulty navigate it. Such interface is called User-friendly interface.

JavaScript [8] is a dynamic, weakly typed programming language that is most often used to develop browser applications as well as Client-Server systems. JavaScript was created to work in a browser environment, and interact with HTML markup through scripts, but over time, due to its convenience, its capabilities have greatly increased, and now it is a full-fledged programming language for different code environments. In order for JavaScript code to be executed, the platform on which the code is directly executed must contain a JavaScript engine. The most common of these engines is the V8. It is used in browsers such as Google Chrome, Opera, Edge, as well as the Node.JS platform. In addition, we can highlight "SpiderMonkey", which is used in Firefox, as well as "Chakra" in Internet Explorer.

The JavaScript engine works in the following way: firstly, it parses the code, then converts it into machine code, after finally this machine code is started.

Node.JS [9] is one of the platforms for executing JavaScript [8] code. Usually it is used to implement the server side [6] of the software system [1]. Server code written on this platform is able to meet all the needs of server development, such as: interaction with databases, work with the file system (which is prohibited in the JavaScript browser environment), interception and sending requests, and more. To improve the speed and quality of code execution, Node.JS [9] uses asynchrony which allows to process a large number of requests in parallel without blocking the flow.

React Native [10] is a framework designed for developing mobile applications based on Android and iOS operating systems [4]. This is one of the most popular technologies for developing mobile applications as well as the most popular based on JavaScript [8]. The peculiarity of this technology is that one code base can be used to develop applications on different platforms. This saves a lot of time and resources for development.

The React Native [10] framework is based on the React.JS library, which is used to develop user interfaces in the JavaScript [8] programming language. This means that developers with experience with the React.JS library can start using React Native for a specific purpose in a short period of time. Also, a great advantage of React Native over other analogues is a large developer’s community. This means significantly fewer problems that may arise during software development using this framework.

MongoDB [11] is a non-relational, document-oriented, open source, easily scalable database management system. Its peculiarity is that unlike relational databases, its storage is based on documents namely in JSON (JavaScript Object Notation) objects in the form of key-values. "Under the hood of MongoDB" JSON objects are transformed into BSON objects. Binary JSON (BSON) is a way to represent data structures in binary form.

Mongoose ODM [12] is an Object Document Mapper that provides a user-friendly interface for working with the non-relational MongoDB database. It provides many methods for reading, modifying and creating new data in the context of a specific database.

Firebase is a platform for easy development of both web and mobile applications. Firebase provides developers with various services for the development of a particular functionality, providing the ability to integrate the desired Firebase service (module) into the

project for further use. Examples of such services are described next. Firebase Analytics is used to analyze the use of the application by users, as well as provide various statistics of the application.

Firebase Cloud Messaging is used to implement the Real-time messages and notifications.

Firebase Auth - provides a user-friendly interface for authorization, allowing email and password authentication, Google authorization, anonymous authorization, and other types of authorization.

The feature of Firebase is that all listed services can be used in serverless [13] systems. It means that implementation of the server is not required.

Firebase also provides a user-friendly interface for interacting with Server side [6] data. This implementation of the functionality of servers based on the Node.JS [9] platform uses the "firebase-admin" module which is present in npm (Node Package Manager).

Express.JS [14] is a Node.JS [9] open source framework designed for use on the Server [6] side of client-server software system. To some extent, this is an add-on to Node.JS [9] that uses the features of the platform itself. It is not difficult to guess that Express.JS uses JavaScript [8] to function, but its context is different from the browser because it is intended for use outside the browser, namely, as it has been already mentioned, on a server with the Node.JS [9] platform. A good quality of Express.JS is its performance, as it has been optimized to work on the Server side. The main functionality of Express.JS is minimalist, but it is more a plus than a minus because its functionality can be easily expanded thanks to npm modules and plug-ins which can be connected.

Android Studio is IDE (Integrated Development Environment) for development of Android applications. It is Google's technology that was released on December 8 2014. Before Android Studio release there was a necessity of plugins for mobile development. One of such plugins was ADT plugin for the Eclipse platform. It wasn't really convenient so Android Studio became a great decision for easy development of mobile applications.

The last but not least part of application development is its testing. It allows to intercept common bugs and fix them right on the development stage. Usually this kind of work is performed by testers. Testers are people who are responsible for the application testing. They should reproduce all the possible scenarios of user workflow. It allows to go through all the functions that application provides thus testing all of them.

There are 2 (two) types of testing: manual and automation. The advantage of automation tests is that they are run independently from human participation so they can be run even at the night, on weekends, in one word – all the time without exceptions. But the disadvantage is that writing automation tests takes pretty a lot of time and development resources.

Manual testing is another type of testing which means that all the testing process is performed by human. It is more reliable and safer than automation testing as it is more flexible. Manual testing is often performed during the process of development. When a part of functionality is implemented by developers, it can be immediately tested manually.

The performance testing of the developed application should be performed with special instruments. One of such instruments is Android Studio Profiler. It allows to monitor how the Android application use the CPU resources, RAM, network and how it affects battery condition.

PROBLEM STATEMENT

Nowadays there are many solutions that would allow the users to monitor the overall situation in their city, district or region by watching local television, monitoring news or local communities on social networks and more. However, there is still no common solution that would automate this whole process.

The peculiarity of the system described in this paper is that in order to become a participant in the action you only need to have a smartphone connected to the global Internet, as well as a mobile application, the functionality of which will be described in this article. In case of emergencies users receive appropriate notifications. It is very important to receive such notifications in time because they can not only save someone's life, but also allow the user to change plans in advance if they have anything to do with the scene of an emergency and so on.

It should also be noted that the above-described functionality does not end the capabilities of this monitoring system. Of course, health and life are one of the main, if not the most important factors in human well-being, but emergencies are not the only thing that can interest modern society. Society is also interested in entertainment like attending interesting events in an area of interest to the individual. There are also sports and watching various sporting events, as well as direct participation in them. The user has the ability to view all the events available in his (and not only his) region, track them, as well as receive notifications about them. This allows to be aware of all the events that in theory may interest you.

In addition, each user of this system has the opportunity to create their own events. To do this, he needs to go to the event creation section, select the event category (e.g., emergency, sport event, music show, etc.) then select the event address and click the Create Event button. After performing these steps, the event will be created and all participants who are subscribed to this category of events in this location will be notified.

The feature of current software system and the difference from its analogs is a fully customizable system of user events and notifications. It means that user can create its own event category and add custom events to it. Then other users who are using current

system can configure the notification system and logic of showing notifications on their own. This feature makes the system useful for all the segments of the population as unnecessary or not interesting to user events which will not be annoying. Such events will not be shown on the map as well as on events list and will not trigger the user notification.

All these features together give users of the system a global understanding of the situation in their locality allowing them to be aware of all events both emergency and secondary which may to some extent affect the life of a person.

PURPOSE OF THE WORK

The purpose of this work is to create a software system for monitoring the situation in the settlement using the latest technologies to develop this type of system, namely the Client side in the form of a mobile application for Android and the Server side in the form of Node.JS server using Express.JS framework and connected MongoDB database. Apart from that – adding integration of Firebase services for authorization and implementation of real-time notifications. It also should implement the idea of fully customizable system for user events and notifications. It means that end users should be able to configure the events and notifications system on their own. This is really important aspect of system functionality.

Another important requirement for the system is application optimization. As the mobile application uses the map rendering, there are often problems with RAM usage. Current application should meet the requirements for standard Android applications. The RAM usage for standard Android applications is between 130 MB and 400 MB.

The resulting program should be a working system for monitoring the situation in the settlement that meets all the requirements described above and allows users to monitor the situation in their settlement and directly participate in the security of their locality.

ALGORITHM FOR CREATING THE SOFTWARE SYSTEM

One of the main requirements and ideas for the implementation of this software system was the use of modern technologies which today are preferred for the implementation of such types of systems. The programming language which is used for the entire system development is JavaScript, the development environment – WebStorm Code Editor.

As a result of the comparison of advantages and disadvantages of technologies (described below), it was decided to use the following technology stack for Client-side implementation: React Native, Firebase, Mapbox maps, Mapbox API.

The main technology for developing a mobile application that will be supported by the Android operating system is React Native as it is one of the most popular, most convenient and most developed frameworks for developing mobile applications using JavaScript. The peculiarity of the React Native is that the application written for the Android operating system can be easily implemented for iOS in the future. The cross-platform nature of the React Native allows to use in the project one code base that can be used for different operating systems.

The Firebase Auth service is used to authorize users using a Google Account. It has no obvious analogues due to the speed of development, ease of use and the huge opportunities provided by the service.

The npm module "@rnmapbox/maps" is used to display maps with events marked in the appendix. For location search implementation Mapbox API service is used. The main advantage over Google Maps and other similar technologies is the pricing policy, as these services are almost free for small and medium software systems.

As a result of the advantages and disadvantages of server-side technologies (described below), it was decided to use the following technology stack: Node.JS, MongoDB, Mongoose ODM, Firebase Services.

First of all, server uses the Node.JS platform as well as the Express.JS framework as they both are the most convenient technologies to work with the REST API. With the help of Mongoose ODM the MongoDB database is connected where the basic data of the system is stored. Their main advantage is speed and ease of development and they are better optimized to work with large amounts of data than relational databases.

The npm "firebase-admin" module, namely its Messaging service, is used to send real-time notifications to users. It can easily communicate with applications using Firebase on the Client side. For using appropriate Firebase services, it is needed to create Firebase account and then create appropriate project which will be used for Firebase services setup.

There is a model of Client-Server Architecture (see Fig. 1) where the application data transfers and its components are described.

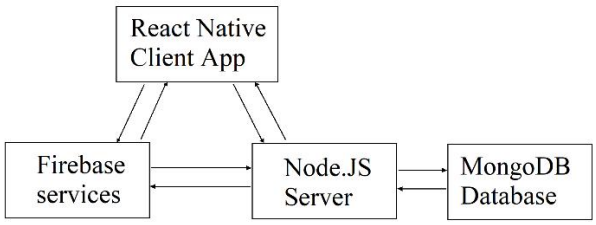


Fig. 1. Client-Server Architecture

The last but not least part of application development is its testing. Moreover, testing part is one of the most important in the development process as it allows to intercept common bugs and fix them right on the development stage. The performance test is used.

DEVELOPMENT OF THE CLIENT-SIDE MOBILE APPLICATION

Work on the client part of the system begins with the creation of the project. To start working with React Native [10] you must first have installed Node.js [9] and Android Studio in the operating system as well as configure the operating system to work with React Native. Then in the console (or in the code editor terminal) enter the command "npx react-native init project-name" in order to create a React Native project. Once the project is created, we can run it. To do this, run the "npx react-native start" command to run the JavaScript Bundle, and "npx react-native run-android" to run the Android Emulator. Once the Android Emulator is run, successfully the smartphone with opened application will appear on the desktop. After completing these steps, the client part of the software system can be directly started to be implemented.

The first stage of development of this software system is the implementation of user authorization. There are two types of authorization available: Google and anonymous authorization. The first opens up much more opportunities to use the application and it is believed that the user will use this type of authorization when working with the system. A user authorized by Google will be able to filter and track events, receive relevant notifications and create their own events. An anonymous user can only view and filter events by category and location. However, when users run the application and they have not previously been logged in with Google, they are automatically logged in as anonymous users.

In order to start working on the software implementation of authorization, you must first create a Firebase project and activate two types of authorization: Google and anonymous.

Next, install the modules "@react-native-firebase/auth" and "@ react-native-google-signin/google-signin" using npm, which provide authorization functionality using the Firebase services. You also need to add the ability to sign out for a user who is logged in through Google Authorization. To do this, use the method auth().SignOut().

The next step is to implement the display of maps on the application screen. There are various services for working with maps such as Google Maps, Mapbox, OpenStreetMap. However, after analyzing their capabilities and pricing policy it was decided to use Mapbox.

Setting up Mapbox service is pretty simple and can be performed in few minutes. In order to start using Mapbox first of all install the "@ rnmapbox/maps". Next, you need to register the Mapbox Account and create a token to access the use of this service in your application. Then in the project code using the method MapboxGL.setAccessToken("your-token") to initialize the Mapbox instance. See Fig. 2 where running Mapbox maps service is presented on Android Emulator's screen.

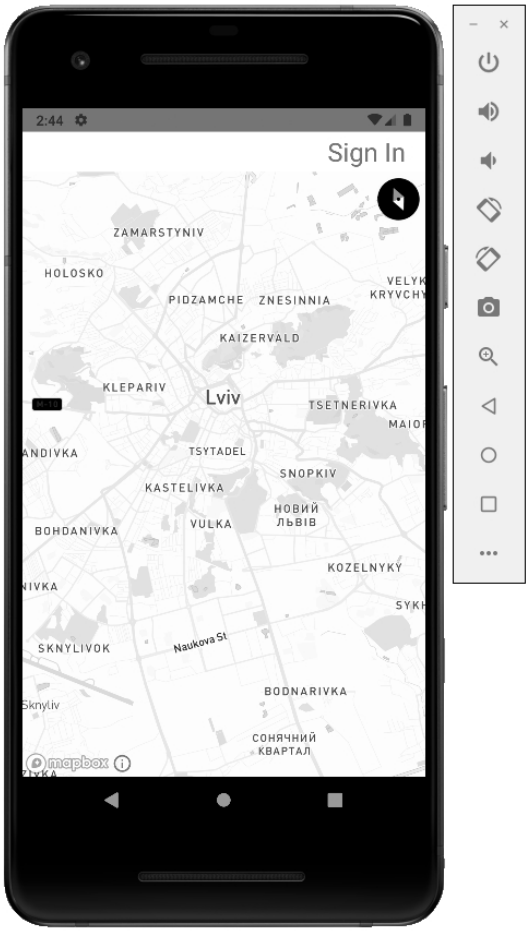


Fig. 2. Mapbox maps rendered on the Android Emulator screen

First of all, the Mapbox services are almost free for usage. It allows to make up to 50000 loads of maps for the application per month that is quite enough for maintaining such kind of applications with pretty small number of users at the beginning. After the application becomes more popular and achieves a large auditory it is important to use better plan which can be accessed by buying it and setting new configuration token to the project

And last but not least is the implementation of real-time notifications. The "@react-native-firebase/messaging" module is used to implement real-time notifications to user's device. This module allows to monitor notifications coming from the server in real time as well as display them even if the application has not been launched or when the smartphone screen is locked. Notifications are sent to users on special tokens that are automatically provided to each user of the system when he logs in with Google authorization. Then, using Firebase Messaging hooks user can be subscribed to incoming notifications. After the notification's setup process given module will perform Foreground, Background and Quit notification.

Foreground notification is a type of notification that is executed in case the user is currently using the

application namely the current application is active. Foreground notification is not directly displayed on the smartphone screen like other types of notifications. It receives a signal that notification has come and gives developer an opportunity to manage notification behavior by itself. It is really useful as in common cases if the application is currently active then notifications should not be shown.

Background notification is a type of notification that is executed in case the application is working in background namely running but not active.

Quit notification is a type of notification that is executed in case the application is not event running.

The received notification is presented in Fig. 3.

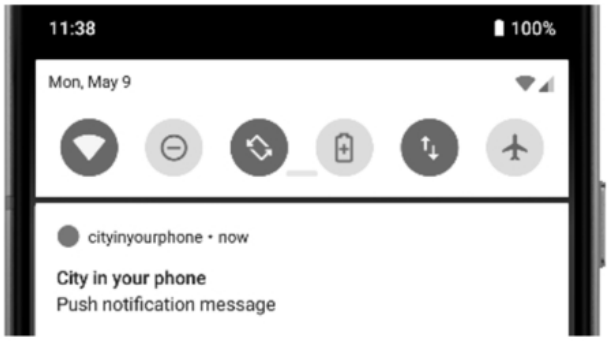


Fig. 3. Real-time notification

DEVELOPMENT OF THE SERVER-SIDE APPLICATION

As in the client part, to start working on the Server side the operating system program Node.JS is needed to be installed. For npm project initialization, enter the "npm init" command in the console or terminal and follow the NPM (Node Package Manager) instructions to setup the project. Then, for a more convenient development process the Express.JS framework is preferred to be installed. It can also be installed using NPM. The next package that should be installed is "nodemon". This module is the best decision for tracking local changes on the server. In case any changes were permitted, nodemon will automatically restart the server application making Server API stay up to date.

To implement one of the main functions of this server, namely real-time notifications, install the npm module "firebase-admin". It allows to initialize the Firebase service. Given module provides developer with a dozen of Firebase methods and classes designed for working with Firebase services on the Server side. In that case, Firebase messaging class with its functionality provide us with methods for working with real-time notifications service like sending notifications to the Client side of program system. However, before that you need to configure the Firebase service by creating a Firebase Service Account. After performing that step developer gets a special application credentials in JSON format which can be downloaded and then located into the server project directory. Use their credentials to initialize server Firebase application calling the appropriate admin.initializeApp({credential: admin.credential.cert(serviceAccount)}) method.

This server, as it has been already mentioned, serves the client part. So, to process all the client requests it must have a set of the following endpoints:

POST /event/create – used for creating new event.

POST /event/update – used for event modification.

POST /event/delete/{event_id} – used for deleting event.

POST /location/subscribe – used for subscription for the location.

POST /event/subscribe – used for subscription for the event.

POST /event-type/create – used for creating new event type (event category).

POST /event-type/subscribe – used for subscription for the event type (event category).

POST /user/create – used for creating new user.

POST /user/update – used for user modification.

The logic of notifications is the following: after user registration, the user receives a special token that identifies the user's device. This will allow to send notifications to this device in the future. Later, when an event is created, a sample is selected from all users who are subscribed to this category of events, as well as those who are subscribed to the location where the event takes place. Notifications are sent to these users using the admin.messaging().send(message) method.

DATABASE DEVELOPMENT

This software system uses the non-relational MongoDB database as well as the Mongoose ODM (Object Modeling Tool). Our system data will be stored in the cloud using the MongoDB Cloud service. The MongoDB Atlas service is used to work with MongoDB Cloud. This is a technology that allows to manage MongoDB databases located on the cloud, and greatly simplifies the process of deploying databases. So, the first step is to create a MongoDB Atlas account and create a new project for our application.

The next step is to model the database. This is one of the most important steps in creating a database. Database modeling is performed taking into account all the requirements for the developed system. The simulated database allows to clearly see all the connections between the collections and the data types which are used.

See Fig. 4 where the database model of current software system is shown.

The database contains of 6 (six) collections:

User - saves user data.

UserConfig - saves user application settings (such as color theme, localization, etc.).

Location - saves the location unit.

EventType - saves all available event types (event categories).

Event - saves events.

Subscription - saves the user's subscriptions to certain events.

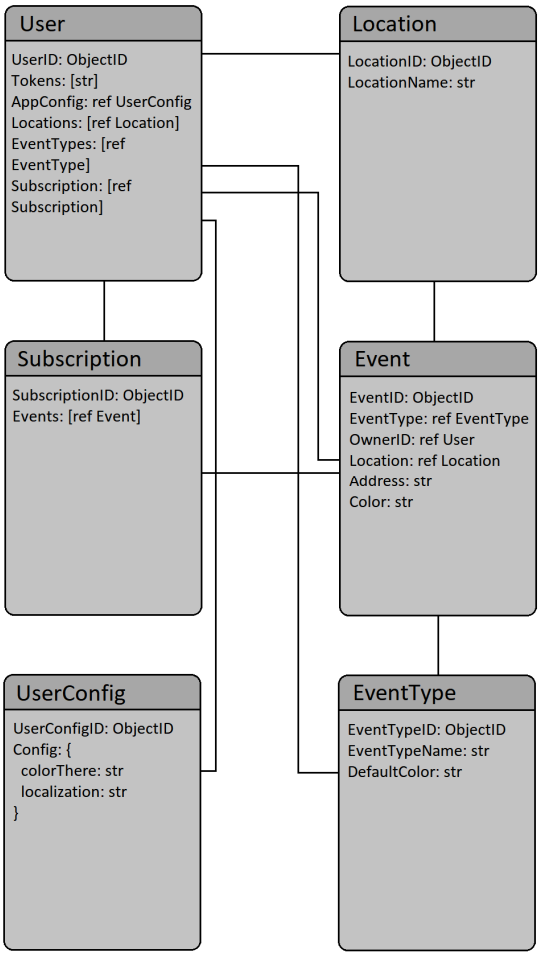


Fig. 4. Database model

APPLICATION TESTING

To test current system, it was decided to use manual testing as it is more flexible and reliable. And, as result of testing, all the found bugs and problems were fixed.

In the process of performance testing it was decided to use Android Studio Profiled instrument as it is built into the IDE, so there is no need to use any another external tools.

There are requirements for mobile applications memory usage which applications should meet. The “Standard” applications should use up to 400 MB of Random-access memory (RAM), the “Media-intensive” – from 400 MB to 700 MB and the “Huge” – from 800 MB and up to 1200 MB.

To test the application performance, it is important to make it work at full strength. This type of testing is called stress testing. Stress testing is used to get the stability of the tested system or application. During the test there are performed operations that bring the use of resources by the application to the maximum. In our case, the most resource-intensive operations are working with maps like intensive map viewing, zoom in and out, focus on locations, etc. So, during the testing in stress mode there was a maximal value of CPU load of 57 % and the maximal RAM usage of 299.3 MB. The battery expenditure is heavy but it is okay for this kind of test (see Fig. 5).

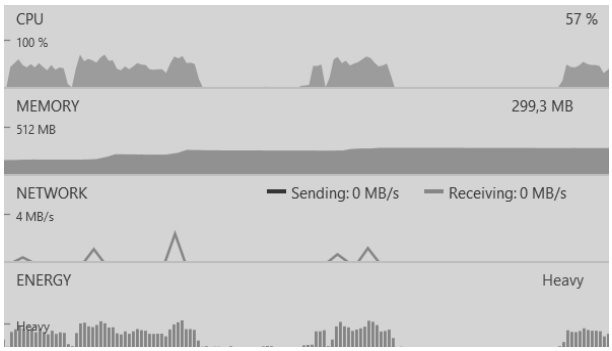


Fig. 5. Performance test in stress mode

CONCLUSION

As a result of this work, a software system for monitoring the situation in the settlement using the latest technologies (which will be relevant and easy to maintain in the future) to develop this type of system, namely the Client side in the form of a mobile application for Android and the Server side in the form of Node.JS server using Express.JS framework and connected MongoDB database, was created. Apart from that, there were integrated Firebase services for authorization and implemented real-time notifications. The idea of fully customizable system for user events and notifications was also implemented. It means that end users are able to configure the events and notifications system on their own. This is really important aspect of system functionality. Another one important requirement for the system was application optimization. As the mobile application uses the map rendering, there are often problems with RAM usage. Current application meets the requirements for standard Android applications. The RAM usage for standard Android applications is between 130 MB and 400 MB namely up to 299.3 MB in the stress mode. The resulting program is a working system that allows users to monitor the situation in their settlement and directly participate in the security of their locality.

REFERENCES

[1] Czibula, G. *et al.* (2018). “An aggregated coupling measure for the analysis of object-oriented software systems”, *Journal of Systems and Software*, 148, pp. 1–20. doi: 10.1016/j.jss.2018.10.052.

[2] Jabangwe, R. *et al.* (2019). “Software engineering process models for mobile app development: A systematic literature review”, *Journal of Systems and Software*, 145, pp. 98–111. doi: 10.1016/j.jss.2018.08.028.

[3] Craig A. Knoblock (1997). “Searching the World Wide Web”, *Computer Conference*, 12, pp. 8–14. doi: 10.1109/MIS.1997.10004.

[4] Cotroneo D. *et al*. (2016). "Software Aging Analysis of the Android Mobile OS", *International Symposium on Software Reliability Engineering*, pp. 478–489. doi: 10.1109/ISSRE.2016.25.

[5] Bharat S. Rawal *et al*. (2012). "Split protocol client/server architecture", *IEEE Symposium on Computers and Communications*, pp. 348–353. doi: 10.1109/ISCC.2012.6249320.

[6] Holliday M. A. and Scott A. S. (2016). "A software development course based on server-side Javascript", *IEEE Frontiers in Education Conference*, pp. 1–5. doi: 10.1109/FIE.2016.7757650.

[7] Alvarez M. *et al*. (2004). "Client-Side Deep Web Data Extraction", *E-Commerce Technology for Dynamic E-Business, IEEE International Conference on*, pp. 158–161. doi: 10.1109/CEC-EAST.2004.30.

[8] The Modern JavaScript Tutorial (2022). [Electronic resource]. – Access mode: https://javascript.info/. (Accessed: May 10 2022).

[9] Liang L. *et al*. (2017). "Express supervision system based on NodeJS and MongoDB", *International Conference on Computer and Information Science*, pp. 158–161. doi: 10.1109/CEC-EAST.2004.30.

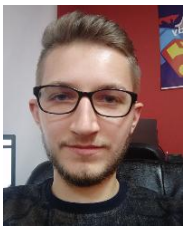
[10] React Native Tutorial (2022). [Electronic resource]. – Access mode: https://reactnative.dev/docs/tutorial. (Accessed: May 10 2022).

[11] Taylor D. (2022). "What is MongoDB? Introduction, Architecture, Features & Example". [Electronic resource]. – Access mode: https://www.guru99.com/what-is-mongodb.html. (Accessed: May 10 2022).

[12] Mundo J. (2017). "An Introduction to Mongoose for MongoDB and Node.js". [Electronic resource]. – Access mode: https://code.tutsplus.com/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527. (Accessed: May 10 2022).

[13] Lakhai, V. and Bachynskyy, R. (2021). "Investigation of Serverless Architecture", *Advances in Cyber-Physical Systems*, 6(2), pp. 134–139. doi: 10.23939/acps2021.02.134.

[14] What Is Express JS In Node JS (2022). [Electronic resource]. – Access mode: https://www.besanttechnologies.com/what-is-expressjs. (Accessed: May 10 2022).



Serhii Kundys is a student who is currently receiving B.S. degree in computer engineering at Lviv Polytechnic National University. His research interests include web front-end development using Vue.js and React.js technologies, back-end development using Node.js and mobile development using React Native framework.

Bohdan Havano received the B.S. degree in computer engineering at Lviv Polytechnic National University in 2015 and M.S degree in system programming at Lviv Polytechnic National University in 2016. He has been doing scientific and research work since 2017. His research interests include architecture and data protection in cyber-physical systems.

Mykola Morozov received the B.S. degree in software department at Lviv Polytechnic National University in 2021. Currently, he is a Postgraduate student of Informatics Department at Technical University of Munich. His research interests include pathfinding algorithms.