Web-Service for Drive Safely System User Analysis: Architecture and Implementation

Aleksandr Fedotov, Igor Lashkov
ITMO University
Saint Petersburg, Russia
mr.fedotovaleksandr@gmail.com, igor-lashkov@ya.ru

Alexey Kashevnik
SPIIRAS, Saint Petersburg, Russia
ITMO University, Saint Petersburg, Russia
alexey@iias.spb.su

***Abstract*—Drive Safely is a mobile application that is aimed at dangerous situation determination while driving based on information from a smartphone front-facing camera and sensors. The smartphone is mounted in windshield for tracking the driver face. During the operation the Drive Safely application generates statistics that includes recognized dangerous states, time, location, speed, acceleration, driver's face parameters, and other information from smartphone sensors. The objectives of the Web-Service developed is to analyze the data obtained and visualize them for the driver offline. Based on the driver's feedback Drive Safely application can improve the quality of dangerous situations determination.**

I. Introduction

According to the statistics of traffic fatalities for the first half of 2016 17,775 people died in motor vehicle traffic crashes in the U.S. [1]. There is a lot of research and development in the world in the area of Advanced Driver Assistant Systems but due to the cost these systems at the moment are installed only to luxury car segment. Drive Safely application [2], [3] is aimed at dangerous situations determination and recommendation generation for the driver while driving the vehicle to prevent accidents. The application uses the driver smartphone installed in windshield to get the context situation in the vehicle cabin.

Modern smartphones are equipped with front-facing camera and variety of built-in sensors such as an accelerometer, a gyroscope, an ambient light sensor, a proximity sensor, a magnetic field sensor, and GPS that are capable to measure parameters that are needed determine dangerous states. At the moment the Drive Safely application determines two dangerous states: drowsiness and distraction. The smartphone's front-facing camera monitors the head movements, facial expressions, and the prolonged and frequent eye blinks that indicates the micro sleep. Visual cues relevant to the drowsiness state are percentage of closure of eyelid (PERCLOS), eye blink time, eye-blinking rate, eye gaze, pupil movement and eyelid movement. Distraction dangerous state is related to mmaintaining the eye contact with the road. Distraction occurs when drivers divert their attention away from the driving task to focus on another activity instead.

Using the smartphone Drive Safely system generates alerts for the driver using vibration, audible signals, and visual information in according with the proposed recommendation schemes [4]. In addition the Drive Safely application provides statistics about the driver's behavior on his/her route. It tracks such parameters as recognized dangerous states, time, location, speed, acceleration, driver's face parameters, and other information from smartphone sensors. Table I shows statistics for distraction dangerous state and Table II shows the drowsiness dangerous state. The statistics is stored in the cloud systems that provides possibility to use the powerful resources to analyses it.

This paper describes the development of the Drive Safely Web-Service that is able to analyze the presented in the table statistics and visualize it in a driver-friendly interface. Due to the visualization the driver can track a route, become aware of dangerous states determined by Drive Safely, and approve or decline the determined by the application dangerous state. Based on this feedback the quality of dangerous situations determination in Drive Safely application can be improved.

The requirements for the developed Web-Service have been specified as follows. The system should be accessible to a wide range of users and it has to support Multilanguage interface. Web-Service should authenticate users by their Google accounts as Drive Safely application uses a Google account connected to the smartphone. Presented in statistics GPS coordinates have a detection error so they should be approximated to the map roads. The driver route should shown in a map. Driver should have possibility to choose the dangerous state and confirm / decline it.

TABLE I. Distraction dangerous state determination statistics

Date/Time	Latitude	Longitude	Acceleration (m/s^2)	Head angle	Speed (km/h)	City/Country	Light level (Lux)
27/04/17 18:38:08:048	59,94228254	30,262599	11,16	-22,51	12	City	116
13/05/17 16:06:32:875	56,18414201	36,9833062	2,1	20,43	26	City	630
13/05/17 16:08:35:733	56,19890477	36,95948511	2,09	-26,25	64	City	565
13/05/17 16:08:54:948	56,20137014	36,95550604	4,08	35,21	80	Country	465
13/05/17 16:36:43:879	56,38791051	36,66605464	9	32,82	67	Country	731
14/05/17 20:23:06:339	60,73556818	30,09532307	6	24,01	67	Country	59

TABLE II. DROWSINESS DANGEROUS STATE DETERMINATION STATISTICS

Date/Time	Latitude	Longitude	Acceleration (m/s^2)	PERCLOS	Left eye open probability	Right eye open probability	Speed (km/h)	City/Country	Light level (Lux)
13/05/17 16:19:52:169	56,30814	36,78249	3,47	0,42	0,24	0,20	85	Country	1583
13/05/17 16:23:28:514	56,33012	36,73267	5,58	0,41	0,26	0,12	67	City	1259
13/05/17 16:32:50:919	56,34909	36,70548	3,61	1	0,27	0,20	76	City	902
13/05/17 16:39:08:220	56,41278	36,64071	5,08	0,33	0,26	0,13	104	undefined	992
13/05/17 16:42:38:188	56,45994	36,5926	3,84	0,42	0,22	0,18	104	Country	462
13/05/17 16:43:21:285	56,4688	36,58359	3,83	0,29	0,19	0,25	71	City	404
19/05/17 17:17:06:017	59,93627	30,28634	5,33	0,37	0,25	0,19	51	City	3924

The rest of the paper is structured as follows. Section II describes the related work in the area of driver behavior analysis. Section III describes to develop Web-Service. Section IV concentrates on implementation. Conclusion summaries the paper.

II. RELATED WORK

Currently, there is a large number of systems for the analysis of the driver behavior. The paper [5] provides comparative analysis of the behaviors of non-professional and professional drivers on the basis of the undertaken individual studies. The results showed the correlation between non-professional drivers and ordinary and aggressive violations and errors, while professional drivers were associated with positive behaviors.

Authors of the paper [6] examines older people driving behavior which includes road selection, left/right turn and driving speed. There are 108 participants has been participated in the study. Authors confirmed that older drivers are more reluctant to drive on expressway than other participants for long trips. Considering that driving in non-expressway with long time may cause frequently facing complex and unsafe conditions, more efforts should be carried out to promote and train expressway driving among older drivers.

The paper [7] describes a tracking system that monitors facial feature and create 3D geometric reasoning. The system has three main components: robust facial feature tracking, head pose and gaze estimation, and 3D geometric reasoning to detect Eyes off the Road state.

The system proposed authors in [8] is the one that determines dangerous states by sound. Authors present silence removal approach using short term energy and zero crossing rate prior to extracting the relevant features in order to reduce the computational time in a vehicular environment.

Predictive driver assistance systems use an in-vehicle video network to determine safety critical events [9] using the specially installed two cameras.

The system proposed in [10] is based on machine learning and visualizes dangerous states while driving that helps people to recognize distinctive driving behavior patterns in continuous driving behavior data. Driving behavior can be measured using various types of sensors connected to a control area network.

The model developed in [11] shows how each type of dangerous states affects the distribution of dangerous zones on a city map.

III. WEB-SERVICE DEVELOPMENT

A. Objectives

To develop a system, it is necessary to identify the tasks that it is to perform. From the general description of the application and the users' needs, the following tasks to be solved were identified:

- Authentication of web application users through Google Account.
- Distribution of roles and rights.
- GPS data processing, their approximation to the roadway.
- Visualization of user routes on the data collected.
- Display on the route areas where dangerous driver conditions are identified.
- Verification of the collected data by the user, for retraining the system for monitoring dangerous states.
- Language localization.

To implement the tasks it is necessary to select the technologies and software tools that allow implementing functionality. Next section describes the system and software architecture of the Web-Service for the analysis and visualization dangerous states.

B. System Architecture

All systems are based on technical tools and consist of several objects. The object of the system is a stand-alone solution or a tool for solving specific tasks. The environment is required to install and ensure objects operation. The ideal solution, in this case, is the virtualization environment, which allows deploying the system elements. To develop the application, the «Docker» [12] virtualization environment has been chosen. It allows to solve the task of ensuring the health of the application in various environments, as well as the problems of defining dependencies, updating components,

scaling the application. The Web-service is broken into components based on their functions, which have individual dependencies packaging. They can be deployed on an architecture different from the standard. «Docker» is easy to manage and convenient to configure. Analogues like «OpenVZ» [13], «LXC» [14] have a large overhead, take longer setup, have poor documentation, experience difficulties with seeking for a finished image for the required functional component. Each object of the system developed is located in a certain container, what allows to isolate the machine resources, simplifying the deployment process on the destination server.

Fig. 1 represents the architecture of Web-Service system developed. It consists of six main components.

«Clickhouse» [15] is a column database for OLAP structures. This database management system is designed specifically to collect unchanging time events and store large amounts of information. It has built-in clustering, which allows adapting to the increasing amount of data. As a part of the Drive Safely, «Clickhouse» keeps route data after processing GPS points obtained during the trip. They are stored in the form of events with a time reference.

«Postgres» database management system [16] keeps data of a smaller size that need to be modified. The database contains information about users, days and routes of trips, t confirmed dangerous states, synchronized data and other application data. «Postgres» is integrated with «Clickhouse», which allows to write aggregation requests using both database management systems.

«Application» is a container that includes the «PHP-FPM» process manager [17], which runs scripts in separate processes for each connection. The connections are initiated by «Nginx» proxy server [18], which acts as the primary firewall and caching of static resources.

Processing information about dangerous states of users, it is necessary to avoid re-synchronization for one user. «Redis» solves this problem. «Redis» - networked journaling data store [19], which allows implementing a blocking pattern.

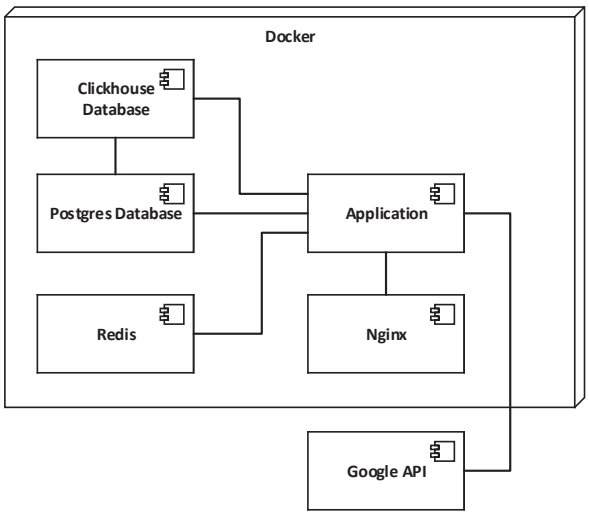

Fig. 1. System architecture of the developed Web-Service

Scaling the number of instances of the application the synchronization runs on one instance, it does not start on another one. The external Google API service is used to authenticate users in the system and makes it easier to integrate with Drive Safely mobile application that uses Android system. To connect to Google API, the protocol «Oauth 2.0» is used [20]. The mobile application saves events to Firebase Cloud Storage in the form of CSV files. These files contain information collected in the vehicle cabin with a frequency of 2-3 events per second. Google Roads API [21] approximates GPS data received from the smartphone to a real roadmap to avoid possible errors. This service interpolates coordinates, i.e. inserts additional points in the order of coordinates for maximum correspondence to the curvature of the road.

C. Program Architecture

To implement Drive Safely the «PHP» programming language has been chosen, as it allows to take the advantage of strict and dynamic typing, has a large community and fast development speed. In order not to search for solutions to typical tasks, the sets of classes previously used and re-usable components are included– in a framework. Framework «Symfony3» has been chosen. This framework is built on the connection of independent components, allowing adhering to the service-oriented approach. Among its -, «Symfony» overcomes its analogs such as «Yii» and «Laravel» in terms of speed and number of integrable libraries.

The architecture of the application completely corresponds to the Model-View-Controller model. MVC is a scheme for applying several design patterns, it supports solution model, user interface, and user interaction. It consists of three groups in such a way that the modification of one of the components has a minimal effect on the others. This design pattern is often used to build an architecture when moving from theory to implementation in a particular subject area.

The scenario for processing the request from the user is shown in Fig. 2. In object-oriented programming, objects are the smallest units of the system. Each object is aimed at performing certain functions and other objects are encapsulated within it. As a result the parent object manages the state of inner objects. The Dependency Injection template is designed to resolve this problem by providing dependency management to the external code. The parent object always works with the finished instance of another object, has no data about its creation and dependencies. The parent object simply provides a mechanism for substituting a dependent object, usually through the constructor or setter method.

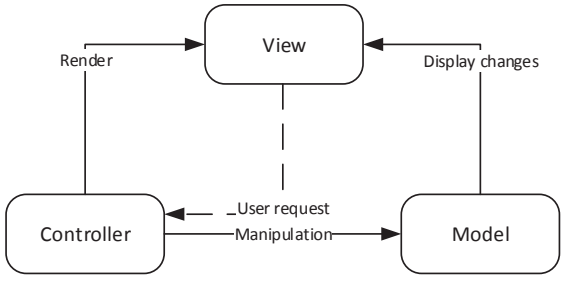

Fig. 2. The request processing scenario

This transfer of control is called Inversion of Control and means that the object itself no longer controls the state of its descendant objects.

The Dependency Injection component in «Symfony 3» relies on the container, manages all registered services and tracks the relationships between them, creates service instances. The components of the application can be divided into 3 groups as shown in Fig. 3: «Symfony» components, vendor components and the system components themselves. Supplier components are the third-party libraries and classes that implement a specific functionality that can be used in different projects.

«Symfony» components are the following. The Security component is the application firewall that provides a user authentication process. «Routing» component specifies the correspondence between the request address and its handler - the controller. «Console» is needed to run the console component of the project. It is necessary for writing synchronization and data processing commands. The «Locale» component is responsible for managing the localization of the application. «File System» is used to work with the file system.

«Http Kernel» is the core of the framework, the handler of incoming requests, initiates the remaining components. «Form» is a component for convenient work with forms of different types and levels of nesting. «Asset» manages resource files for a web client.

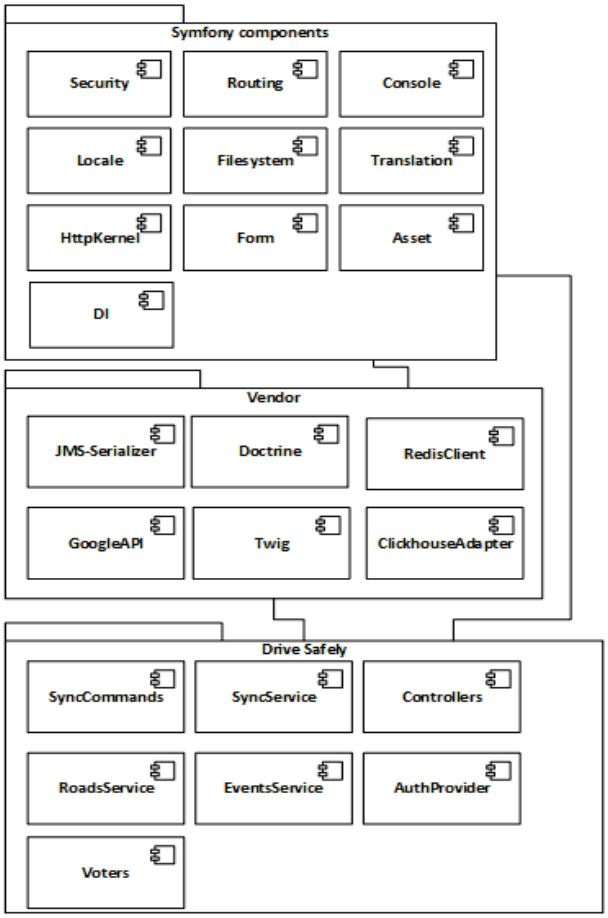

Fig. 3. Web-Service Components Chart

«DI» is the implementation of Dependency Injection and Inversion of Control pattern. Its structured code allows adhering to the best practices.

Supplier Components:

- «JMS-Serializer» is a serializer and deserializer of the objects. Used for API and for mapping the objects for «Clickhouse» database.
- «Doctrine» is ORM for working with «Postgres» that also provides DBAL for «Clickhouse».
- «Redis Client» is a client for working with «Redis» necessary to implement non-duplicating synchronization.
- «Google API» is a library for working with the Google API.
- «Twig» is a view template for displaying pages and organizing their hierarchy. «Clickhouse» Adapter is an adapter for working with «Clickhouse».

The components of the application are to be considered in detail. «Sync Commands» is a set of commands for synchronizing events from the cloud storage of Firebase. It includes parsing csv files and calling the «RoadsService» functions. Events are collected data from the driver's cab at a specific point in time. «Sync Service» a set of control classes for calling synchronization commands for users. It uses «Redis» to block the synchronization of an individual user so that at a given time the commands are not called for this user on other instances. «Controllers» are the controllers of the application transfer data from models to a template or serializer. «Roads Service» is a set of classes for identifying road routes in a sequence of events approximating GPS data to a roadway. «Events Service» is a set of classes for working with events defined as dangerous. The event data structure has differs from the main one. In addition, this service is responsible for confirming a dangerous condition. «AuthProvider» is an integration component for Google authentication. «Voters» are classes for providing access control to application objects. It checks whether the current application has rights for actions with a particular system object.

Web-Service class diagram is presented in Fig. 4. Controllers are the classes responsible for the relationship between the operation logic and the template by which the data is displayed. «PanelController» is responsible for displaying the pages of the user panel.

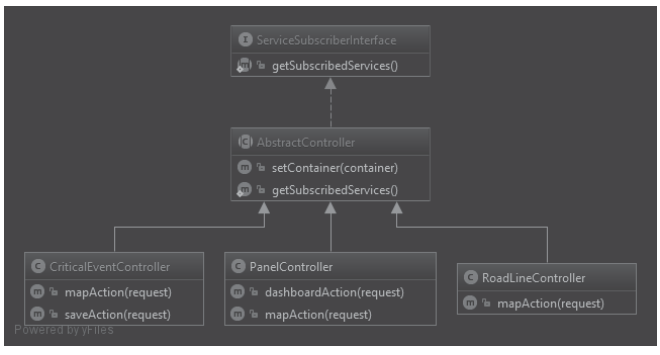

Fig. 4. Web-Service class diagram

«CriticalEventController» gives and stores data about dangerous states. «RoadLineController» provides a set of coordinates for the selected trip.

Fig. 5 presents console commands that migrate data from the event log files stored in Firebase to the application database. «PanelStatEventsCommand» is the command that processes and stores data from a file with coordinates and states of the system at certain points in time. «ParseCriticalEventCommand» is the command that processes and stores data from a file with dangerous states. «LoadFireBaseUserFileCommand» searches and downloads unprocessed files from Firebase Storage. Fig. 6 presents classes that provide access to the application entities. They encapsulate the logic of queries to the database. A separate repository is created for each entity as in terms of the functional point of view they provide access to different objects. «RoadLineRepository», «UserRepository», «CriticalInfoEventRepository», «CriticalEventRepository», «StatEventRepository» are repositories for corresponding models containing queries to databases.

Fig. 5. Console commands that migrate data from the event log files stored in Firebase to the application database

Classes that control the creation and updating of entities are presented in Fig. 7. «CriticalEventManager», «StatEventManager», «RoadLineManager», «CriticalInfoEventManager» are managers responsible for a particular entity perform additional conversions if necessary. Separated classes ensure the purity and statefulness of the code.

The diagram of the services is presented in Fig. 8. «CriticalEventParser» is a class for processing dangerous state files. «StatEventParser» class processes files with event data in the driver's cab. «StatEventFactory» and «CriticalEventFactory» are the entity creation factory of «StatEvent» and «CriticalEvent» made from the lines of the file. They perform the conversion of string data to the types stored in the database. «BatchGenerator», «ObjectGenerator», «FilterGenerator», «RoadLineGenerator» are generators for processing files that are lined up for streaming processing.

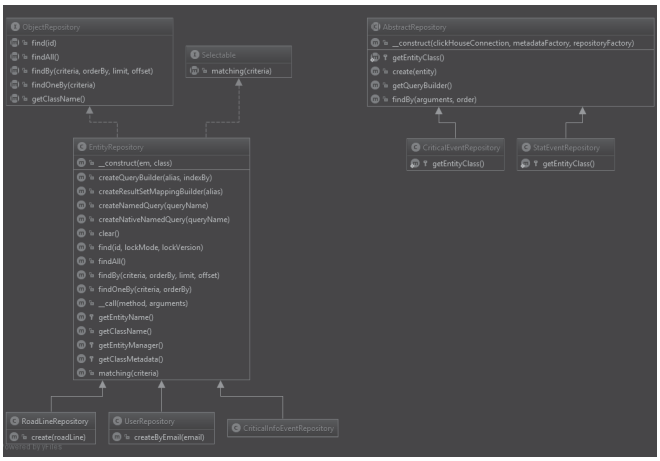

Fig. 6. Classes that provide access to application entities

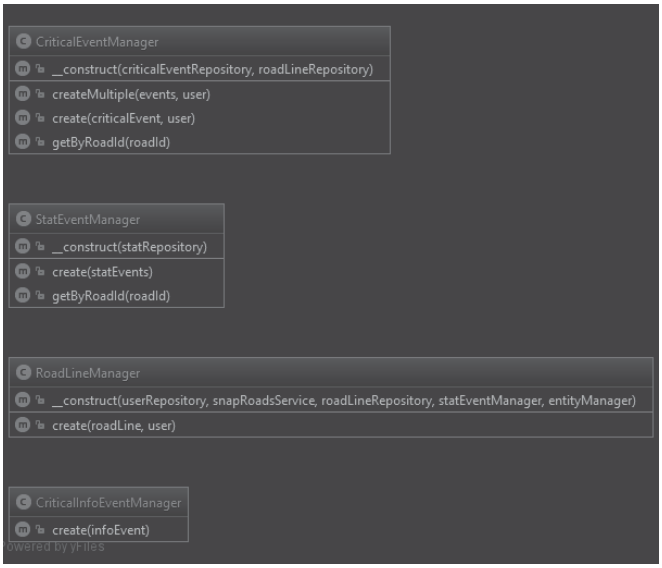

Fig. 7. Classes that control the creation and updating of entities

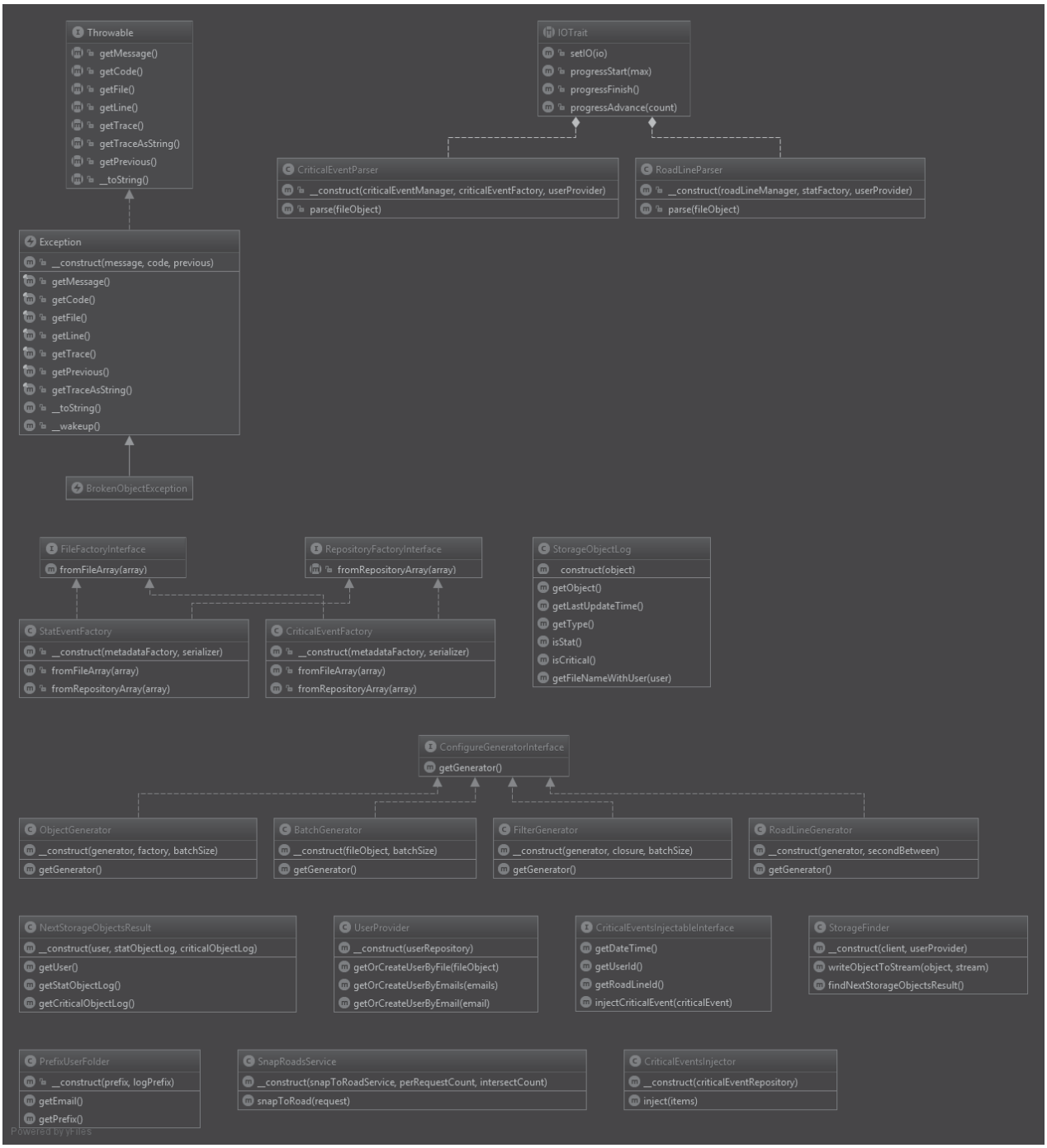

Fig. 8. Services diagram

«BatchGenerator» splits the sequence into blocks of data for block processing. «ObjectGenerator» performs the generation of objects using factories that implement the «FileFactoryInterface» interface. «FilterGenerator» filters data on the conditions specified in Closure. For example, if the nearest objects have the same stats or no valid data, they are skipped. «RoadLineGenerator» aggregates events in a continuous sequence of one route. «SnapRoadService» performs the interpolation and approximation of GPS points using Google Roads API.

«UserProvider» provides the user's entity by necessary criteria. «StorageFinder» is a service that follows Firebase files that are not processed. It uses supporting classes «StorageObjectLog», «PrefixUserFolder» «NextStorageObjectsResult», which encapsulate storage objects. «CriticalEventInjector» aggregates objects of dangerous states and information from users.

D. Data Architecture

Fig. 9 shows the key entities of the Web-Service developed. They illustrate how data is logically grouped into the proper

objects. Within the project structure five main models for data storage are used. «CriticalEvent» represents events identified by the system as dangerous. The entity contains a data structure responsible for a critical state. It contains an extensive aspect of the parameters that display the values collected in the driver's cab. «StatEvent» represents events occurring for values within a short time period. It contains data on the movement of the vehicle, as well as smartphone performance.

«User» is the model of the user that contains the credentials of the «Oauth» identifiers and the data of the last synchronization.

«RoadLine» is a model representing a dedicated route within a certain time range. It groups events into a continuous sequence that can be displayed on the map. «CriticalInfoEvent» contains the data that the user provides after examining the statistics. It is necessary for training the system.

IV. Implementation

The development process is divided into several stages. The result of each stage is the input parameter for the next one. Thus, the set of steps creates a continuous chain that shows the development sequence of the Web-Service as seen in Fig. 10. In order to understand what tasks the system should solve first, it is necessary to analyze the subject area, then choose the tools that can solve the tasks. After developing the architecture, it is implemented with the tools selected adding operation logic that meets the requirements and templates responsible for displaying it.

Fig. 9. Key entities of the developed Web-Service

The next stage is to add dynamics to the user interface and conduct integration testing.

Fig. 11 presents the Web-Service main screen that contains brief information about the product, has a navigation menu and a login button. The homepage of the personal profile contains some graphs on the user's routs and dangerous states as shown in Fig. 12.

Fig. 13 shows the map of dangerous states. Here it is possible to choose the day and the perfect rout on a given day. The map indicates the points at which dangerous states are found.

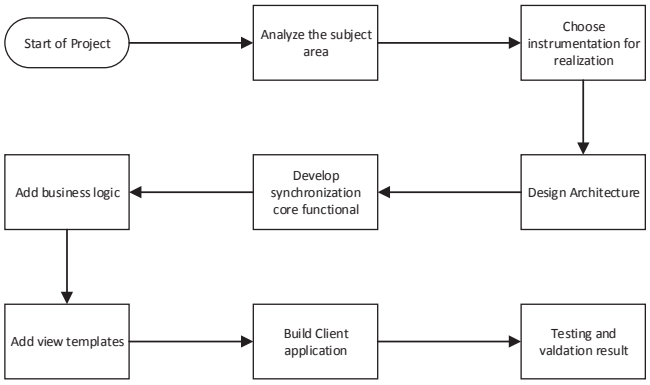

Fig. 10. Development sequence of the Web-Service

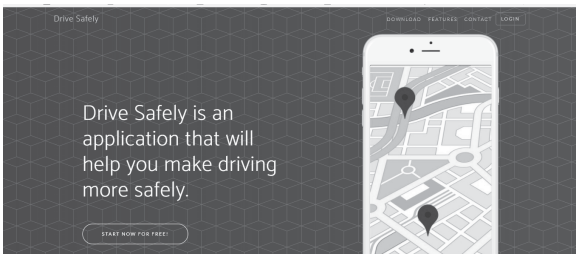

Fig. 11. User interface: Web-Service main screen

Fig. 12. User interface: homepage of the personal cabinet

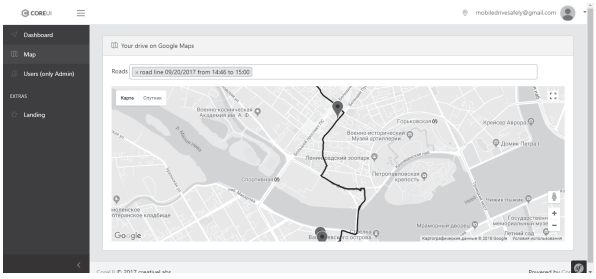

Fig. 13. User interface: map, route, and determined dangerous states

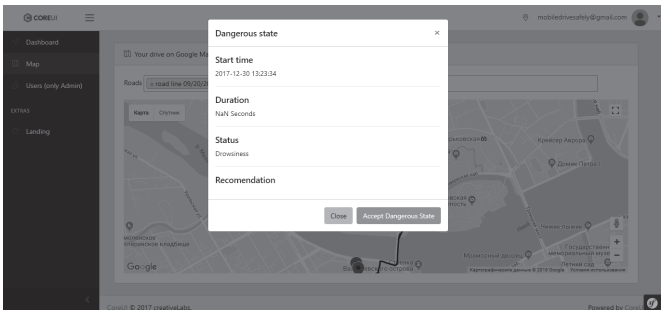

Fig. 14. Dangerous state

When the user clicks at the point, a pop-up window shows specific information about the dangerous state (Fig. 14). In addition, the user is able to note whether there is a critical condition in fact or not. A confirmation is stored in the database and used for the automatic improvement of the system.

V. Conclusion

The paper presents Drive Safely Web-Service for the driver behavior analysis which is used by Drive Safely system. The Drive Safely system generates driving statistics describing the driver's behavior on his/her route. The statistics is uploaded to the cloud and becomes accessible for the Web-Service. Web-Service developed allows tracking the driver's path and confirming or declining the determined dangerous states. This provides Drive Safely system with the possibilities to evaluate and enhance dangerous states determination.

Acknowledgment

The work has been partially financially supported by grant # 17-29-03284 of the Russian Foundation for Basic Research, by the Russian State Research # 0073-2018-0002, and by Government of Russian Federation (Grant # 08-08).

VI. References

[1] Early Estimate of Motor Vehicle Traffic Fatalities for the First Half (Jan–Jun) of 2016, National Center for Statistics and Analysis, Report No. DOT HS 812 332, Washington, DC: National Highway Traffic Safety Administration.

[2] A. Kashevnik, I. Lashkov, V. Parfenov, N. Mustafin, O. Baraniuc, "Context-Based Driver Support System Development: Methodology and Case Study", *Proceedings of the 21st Conference of Open Innovations Association FRUCT*, Helsinki, Finland, 6-10 November 2017, ITMO University, St. Petersburg. 2017. pp. 162–171.

[3] A. Smirnov, A. Kashevnik, I. Lashkov, V. Parfenov, "Smartphone-Based Identification Of Dangerous Driving Situations: Algorithms and Implementation", *Proceedings of the 18th Conference of Open Innovations Association FRUCT*, St. Petersburg, Russia, 18-22 April 2016, ITMO University, pp. 306–313.

[4] A. Smirnov, A. Kashevnik, I. Lashkov, "Human-Smartphone Interaction for Dangerous Situation Detection & Recommendation Generation while Driving", *Speech and Computer: 18th International Conference on Speech And Computer (SPECOM 2016), Proceedings, Budapest, Hungary, 23-27 August 2016*, pp. 346–353.

[5] M. Maslac, B. Antic, K. Lipovac, D. Pešic, N. Milutinovic, "Behaviours of drivers in Serbia: Non-professional versus professional drivers", *Transportation Research Part F*, vol. 52, 2018, pp. 101–111.

[6] Y. Zhao, T. Yamamoto, T. Morikawa, "An analysis on older driver's driving behavior by GPS tracking data: Road selection, left/right turn, and driving speed", *Journal of Traffic and Transportation Engineering*, In Press.

[7] F. Vicente, Z. Huang, X. Xiong, "Driver Gaze Tracking and Eyes Off the Road Detection System", *IEEE Intelligent Transportation Systems Society*, vol. 16, Issue: 4, 2015, pp. 2014–2027.

[8] N. Kamarud, A. Wahab A. Rahman, K. Ikhwan Mohamad Halim, "Driver behavior state recognition based on silence removal speech", *International Conference on Informatics and Computing (ICIC)*, 2016.

[9] E. Ohn-Bar, A. Tawari, S. Martin, M. M. Trivedi, "On surveillance for safety critical events: In-vehicle video networks for predictive driver assistance systems", *Computer Vision and Image Understanding*, vol. 134, May 2015, pp. 130-140.

[10] H. Liu, T. Taniguchi, Y. Tanaka, "Visualization of Driving Behavior Based on Hidden Feature Extraction by Using Deep Learning", *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, Issue 9, 2017, pp. 2477–2489.

[11] S. Sharda, "Mapping Risky Driver Behavior and Identifying their Contributory Factors: A Spatial Statistical Approach", *Master of Science in Civil Engineering Thesis*, Montana State University, Bozeman, Montana, 2016.

[12] Docker official website, Docker - Build, Ship, and Run Any App, Anywhere, Web: https://www.docker.com/

[13] OpenVZ official website, OpenVZ Virtuozzo Containers Wiki, Web: https://openvz.org/

[14] LXC official website, Linux Containers - LXC – Introduction, Web: https://linuxcontainers.org/lxc/

[15] Clickhouse official website, ClickHouse — open source distributed column-oriented DBMS, Web: https://clickhouse.yandex/

[16] Postgres official website, PostgreSQL: The world's most advanced open source database, Web: https://www.postgresql.org/

[17] PHP-FPM official website, Home - PHP-FPM, Web: https://php-fpm.org/

[18] Nginx official website, nginx news, Web: http://nginx.org/

[19] Redis official website, Redis, Web: https://redis.io/

[20] OAuth 2.0 spec official website, OAuth 2.0 – OAuth, Web: https://oauth.net/2/

[21] Google Maps API official website, Google Maps APIs Google Maps APIs, Web: https://developers.google.com/maps/documentation

[22] PHP official website, PHP: Hypertext Preprocessor, Web: http://php.net

[23] IEEE official website, Visualization of Driving Behavior Based on Hidden Feature Extraction by Using Deep Learning - IEEE Journals Magazine, Web: http://ieeexplore.ieee.org/abstract/document/7839988