Development of an Automated Passenger Transport Management System Using Microservices Architecture

Mykyta Dermenzhi[1], Svitlana Kuznichenko[2], Tetiana Tereshchenko[3], Iryna Buchynska[4], Viktoriia Klepatska[5]

[1, 2, 3, 4, 5]*Odessa State Environmental University, 15 Lvivska Str, Odesa, 65016, Ukraine*

Abstract

The paper presents applied aspects of the design and development of an automated passenger transportation management system for any transport company. A flexible architecture of the transport system (TS) is proposed, which is built according to the microservice methodology and simplifies the synchronization processes between drivers and operators. Vehicle information is updated via GPS. The proposed design approaches allow to customize the TS to the individual needs and initiatives of customers and quickly expand functionality and add new features and services. In addition, the system can be used as an intermediate node for embedding in an existing system to collect information and provide a graphical user interface for operators.

Keywords

Transport system, automated system, GPS, microservices architecture.

Introduction

Transport is one of the most important branches of social production and it is designed to meet the needs of the population in transportation. In Ukraine, most bus services are provided by small private enterprises. Their fleet of vehicles provides urban, long-distance and international transportation in Europe and the CIS. A feature of the work of private operators is the difficulty of control and the lack of a single system for monitoring vehicles on routes, which is often an obstacle to making operational decisions to optimize the operation of transport. In this regard, there is a need to create an automated passenger traffic management system with the following capabilities: 2

- Storage and management of information about vehicles and their technical characteristics;
- Driver information management: current position during the trip, contact details, the vehicle to which they are assigned;
- Ability to expand locations on the map and update them: location is a material point, which is added by the operator to establish a point of repair, parking, stopping, etc.;
- Keeping records of reference data on emergency repair and synchronization of data from external resources;
- Creating routes using external APIs, using software to store information in internal repositories and caching this data for the fastest response from the server;
- Systematize available information for planning future transportation;

III International Scientific And Practical Conference "Information Security And Information Technologies", September 13–19, 2021, Odesa, Ukraine
EMAIL: nikita.dermenzhi@gmail.com (A. 1); skuznichenko@gmail.com (A. 2); tereshchenko.odessa@gmail.com (A. 3); buchiskayira@gmail.com (A. 4); victoria.klepatska@gmail.com (A. 5)
ORCID: 0000-0003-3564-9806 (A.1); 0000-0001-7982-1298 (A. 2); 0000-0001-7691-6996 (A. 3); 0000-0002-0393-2781 (A. 4); 0000-0001-5613-6546 (A. 5);

- Automatic processing of the vehicle's arrival to the final stop and the ability to adjust the time in case of trip delays.

In recent years, researchers have shown great interest in the public transport system. This is largely due to the growth of cities and transport development of territories [1]. The principles of creation and features of transport systems (TS) architecture are widely discussed, which can provide important programs and services to improve the safety and mobility of passenger traffic, as well as to optimize transport resources and time [2-4]. The article [5] discusses public transport management systems for future smart cities built using Internet of Things (IoT) systems. TS is actively used in the architecture based on the joint use of IoT and geographic information systems (GIS), which have great potential to support decision-making in various fields of human activity [6-8], including public transport. Thus, in [9] a project of emergency management system for public transport networks was presented, which uses IoT technologies for traffic monitoring, as well as GIS to facilitate situational awareness and emergency operations. This work [10] explores how to develop an extremely flexible and comprehensive architecture for TS that can use the latest technologies, such as cloud computing and the subscribe-publish communication model.

However, despite the large number of publications related to the development of transport management systems, it is important to consider the applied aspects of the development of their infrastructure and implementation stages of design solutions.

In addition, one of the main problems associated with TS are external and internal integration processes, such as: the implementation of data dependencies and tracking their changes, tracking cars using intermediate services, and so on.

Thus, the main goal of this project is to develop an automated passenger transportation management system of the transport company with the ability to: systematize data obtained during trips, receive instructions for creating and installing new services, infrastructure preparation (ease and speed of deployment, integration of customer data, etc.), the ability to use the system on any platform and support for gadgets from phone to computer.

Presentation of the main research material

The main non-functional requirements (NFR) for the project are:

- completeness of information and technical specifications;
- availability of the service regardless of time;
- the ability to expand the content;
- dynamic cartography, independence from the map provider;
- confidentiality of customer data;
- security of obtaining data on synchronization of movements of system objects;
- service of operators and drivers in real time with minimal delays;
- design flexibility and ease of use by end customers;
- the ability to scale the system and the number of services during support;
- the possibility of the system deployment process on different platforms;
- support for data integration using resources and sources implemented by the customer;
- possibility of constant logging and data recovery;
- simplification of visualization on mobile devices and tablets, to maintain the full consistency of data, as well as the use of the platform by operators in cases of inability to use a computer;
- fast conversion of content with translation into other languages;
- dynamic specifications (vertical expansion): CPU speed, memory, disk space, network performance.

In addition, to determine the progress of the project, functional sections were introduced, which in turn build the system as a whole: Models, Vehicles, Drivers, Contacts, Locations, Cities / Regions / Countries, Routes, Schedules, Trips, System Settings.

Prerequisites for the implementation of the system

With each passing day, the implementation or sequential integration of artificial intelligence (AI) according to [11] research is becoming an increasingly important process in information systems. Such services can allow you to build trips more accurately, help resolve conflicts, and more. This project is integrated with TomTom systems, which provide AI API for building and calculating trips. Such routes are close to real, and it becomes easier for drivers and operators to coordinate actions.

Most modern systems use the Global Positioning System (GPS), which allows you to get the coordinates of vehicles and place marks on a virtual map [12].

It should be noted that the developed system also implements this approach and deepens it. This is an optimization model, when information is provided in small portions, and the user while navigating the virtual map loads updated or changed data. This reduces the load on the server and facilitates display on the client application.

Additionally, the system provides the ability to automatically calculate and build routes, simplify the workflow for operators and synchronize statuses for system users. In addition, the system can be used as an intermediate node to integrate into an existing system and used to gather information and provide a graphical interface for operators. This approach saves time on reporting and allows you to use the data recorded in the system as automatically generated reports. This increases the level of security and also reduces the risk of error on the part of the operator. All current data: vehicle location, rerouting, trip status update, adding or deleting locations and system users, are stored at the information store level.

Note that most of the existing logistics systems are tied only to the construction of vehicle tracking without the possibility of integration into this process of other users of the system, such as operators, administrators, persons who provide parking and repair facilities, etc. This project solves the main problem of transport companies: combining all operators, drivers and intermediate users of the system in a single application and saving current data for sequential analysis.

Project consumers can choose their own configuration and use integration processes to enter existing data (drivers, locations, automotive objects, routes, etc.) using an external integration service. This allows you to not disrupt the technological processes of the business and make it easy to combine two or more projects in one ecosystem.

It is assumed that the consumer is a self-employed person or a commercial group who is interested in saving time and money on late shipments, and instead wants to use systematization, storing important data on travel and drivers for further analysis. Data analysis of this type can provide an understanding of how employees perform their work and will allow them to track problems without direct contact with operators.

Implementation of constant vehicle tracking

There are several basic approaches to obtaining information from external resources [13]:

- Long Polling;
- WebSockets;
- Server-Sent Events.

All of them allow you to get up-to-date information using external systems.

WebSocket is a computer communication protocol that provides full-duplex communication channels over a single TCP connection. It is synchronous, which gives a high level of correctness, but connecting a large number of connections can lead to poor performance of the module.

Server-sent events (SSEs) are a one-way communication channel where events are transmitted only from the server to the client. Events sent by the server allow browser clients to receive the stream of events from the server over an HTTP connection without constant polling.

It was decided to use a combined approach to connect the system to the constant updating of vehicles via the GPS system: to create an intermediate module that will use web sockets, and to receive synchronous information.

As soon as the socket sends a new message from the vehicle tracker, this message is transmitted to the asynchronous queue. Thus, the queue will use the SSE approach, which sends messages to all system subscribers (authenticated users) through other communication channels (Fig. 1).

Architectural approaches during implementation

The project is based on the principles of the structure of micro-services, as shown in Fig. 2. The first service is a graphical user interface (GUI) layer what is a web application. The next mechanism in the system that provides most of the data changes and external resource providers is called the micro-service pool. A microservice is a small piece of system that can reside on different servers or even be embedded in a partition of an application, and works with external APIs, data warehouses, or prepares scheduling events.

The basic access service (BDS) is a common component for connecting to the data access level (DAL) and some necessary plug-ins: mapping, service data, etc. An internal database is an abstract layer that can be switched between providers and has a single interface. The only limitation is the need to have entities that are used in the main application and stored procedures. This type of abstraction makes it possible to choose the most convenient provider for the database and absolutely provides a connection to the structure, as separate as possible from dependencies. This principle is the principle of GRASP, Low Coupling & High Cohesion [14]. To understand below (Fig. 3), the main set of micro-services in system architecture is presented.

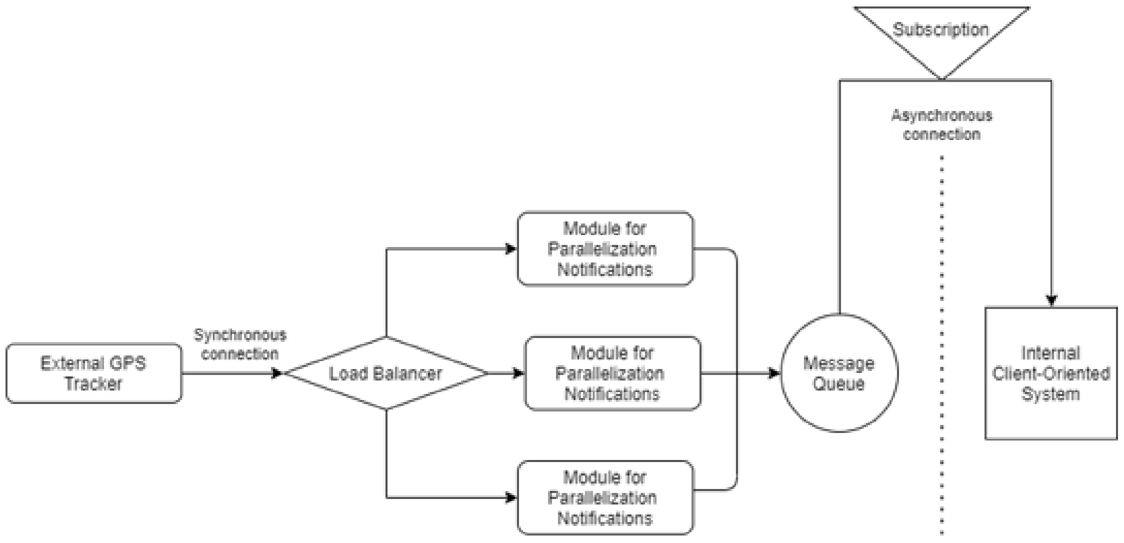
Subscription
Asynchronous connection
Module for Parallelization Notifications
Module for Parallelization Notifications
Module for Parallelization Notifications
External GPS Tracker
Synchronous connection
Load Balancer
Message Queue
Internal Client-Oriented System

Figure 1: Combined approach to sending messages

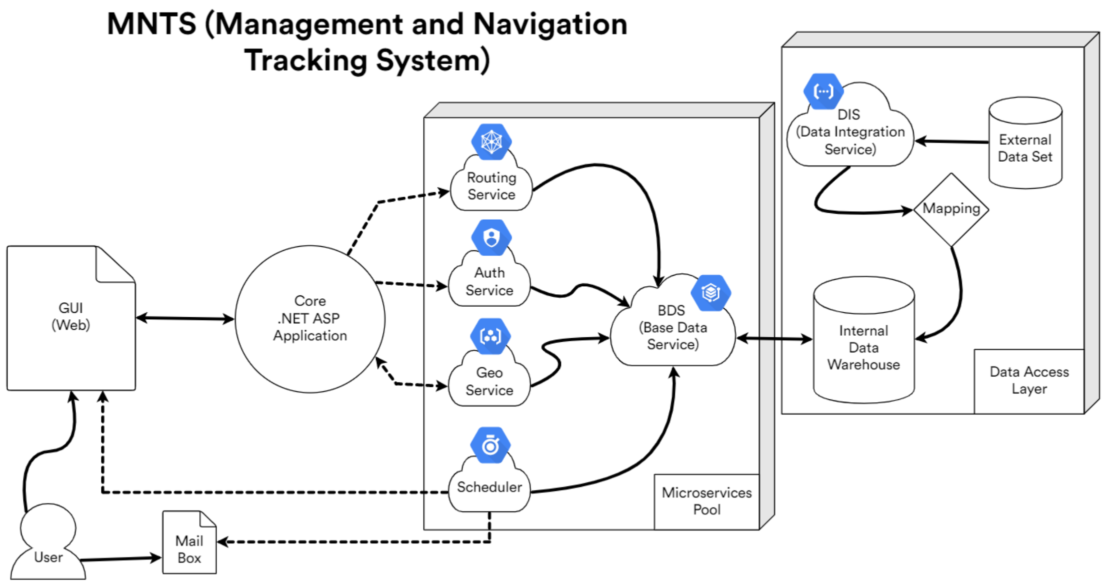

Figure 2: Diagram of micro-service architecture

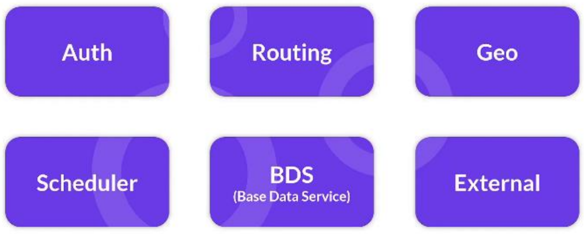

Figure 3: Micro-services in the system

External datasets will be processed by the Data Integration Service (DIS), which will then be passed to a mapping process, where the data is converted to an internal model. Authentication service (AS) is a modern generation of token systems called JWT (JSON Web Token). This service allows you to authenticate and then identify moderators using only a hash with metadata sealed in it. The Geolocation Service (GS) is the "core" of the project and its main part. This means that the GS directly affects the data, in addition, it must build linked lists and include some additional data about places and objects.

It should be noted another pattern that was used during the implementation of the system: the Routing Service (RS) is based and configured using the TomTom API. To reduce requests, a Proxy template [15, 16] has been implemented, which envelops this functionality and saves data to reduce the number of requests, thus reducing the cost of services (Fig. 4).

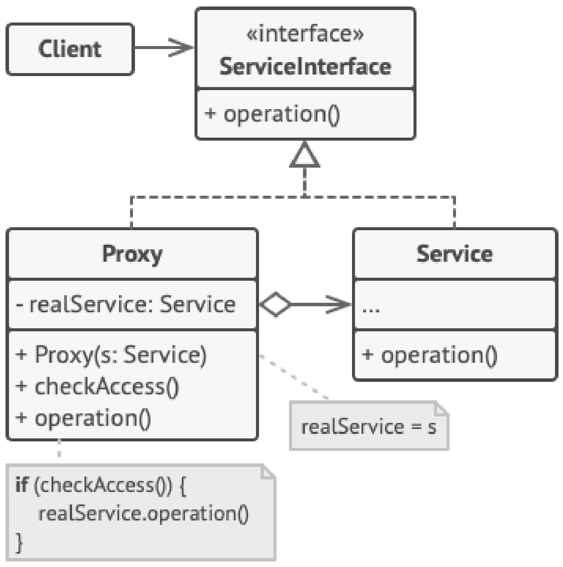

Figure 4: Proxy Pattern

For example, if a route has the same location, the system will use the saved data instead of making a new request.

The scheduler is the only module that can be either a separate service or a built-in part of the developed system. For the data warehouse, SQL Server was chosen, which provides the best connection experience and many features, such as triggers to schedule or even save data analysis if necessary.

Internal processes of the system

To explain the implementation, consider the Use-Case diagram in the Fig. 5, which presents the two main users of the system: the administrator and the driver.

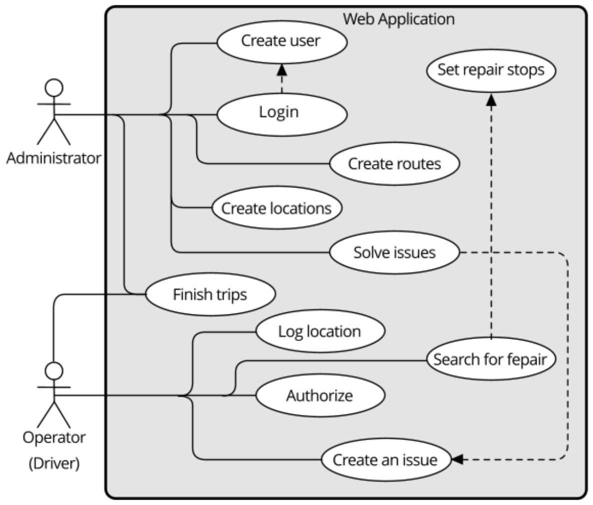

Figure 5: Use-Case diagram

As can be seen from the diagram, the driver has the following functionalities: go through the process of authorization and authentication, has automatic logging of its location and display in the application for administrators, can create requests to solve problems during trips, search for parking or repair; register intermediate statuses and terminate trips automatically or manually.

In turn, the administrator monitors the capabilities and responsibilities of drivers, confirming actions, as well as coordinates and solves problems as needed: creates new administrators and moderators, drivers, logs in the system, creates locations, fills in new information for system users, sets stop for repairs and parking, solves problems and problems of drivers during trips, creates new routes and manages trips.

The next step of implementation is the functions of creating new entities of the system, as shown below in the activity diagrams: the processes of creating a new driver, creating and making changes to routes and locations, creating a trip and accompanying processes during its execution.

As shown in the Fig. 6 - the process of creating a driver begins with filling out a questionnaire and then it branches into two parallel processes, which can be asynchronous, which means the possibility of their delayed completion. If necessary, a separate entry is created for the driver with a new bus or an existing one is selected. After filling out the questionnaire, you can additionally fill in the contact details. Finally, all processes are synchronized and sent to the database.

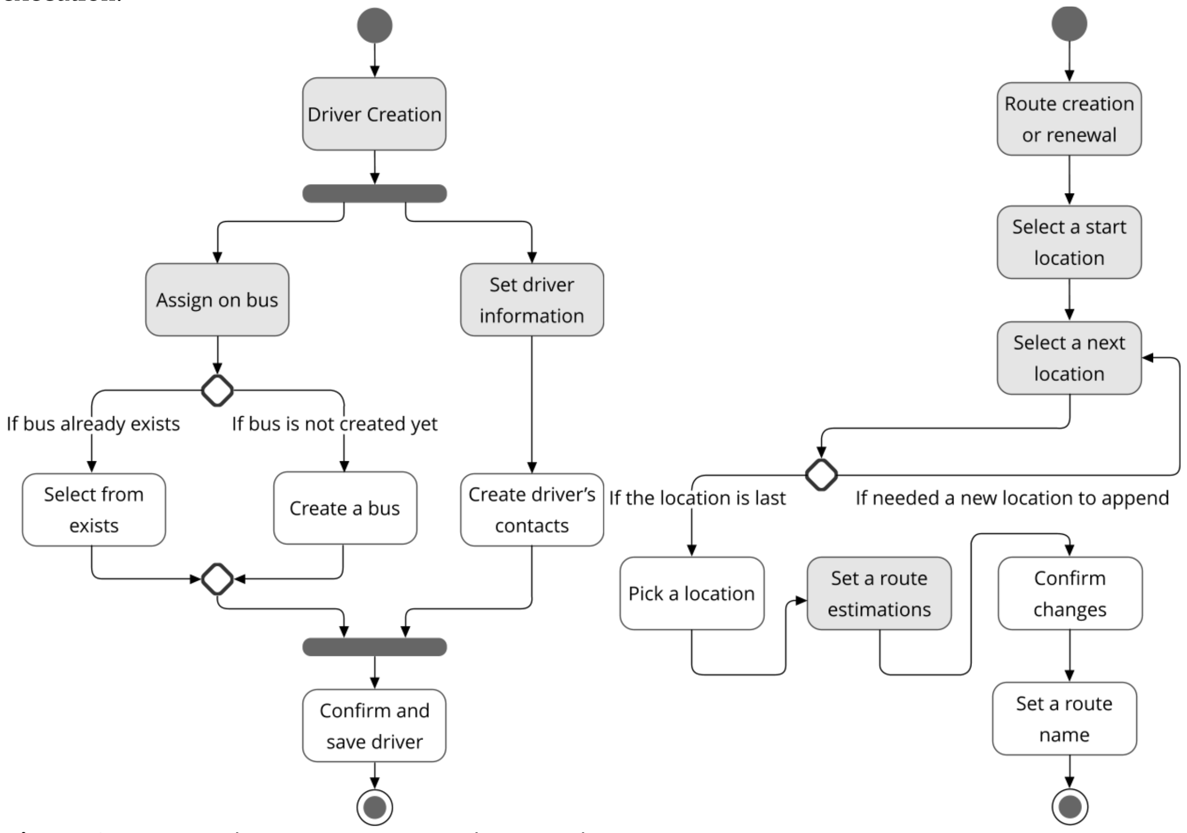

Figure 6: Activity diagram. Creating a driver and route

Creating and updating routes begins with a modal window that offers to specify or redefine the starting point, if this has been done or the route already has a starting position, the user can select new intermediate stops or change existing ones. The final step is the process of specifying the last location, which cannot be changed after. The system automatically offers estimates of time and route length based on data obtained during route calculation.

In order to start a trip, you need to create a new schedule or choose an existing one, after which the algorithm branches into two parallel processes: creating a system of stops with time intervals, setting metadata for a specific trip, as in Fig. 7. The system also has an automatic time calculation, which determines how much real time is required to travel the distance based on the geolocation of data. The process of creating metadata takes responsibility for specifying which bus, driver will perform the trip, as well as adjusting such parameters as date and time, etc.

Finally, we note the life cycle of the entire system through a state diagram (Fig. 8), which indicates the process, starting from the creation of the user, ending with the trip process and logging its final destination.

Conclusions

The developed system and the conducted researches introduce the service methodology for creation or integration into the existing systems for simplification of processes of synchronization between drivers and operators. The proposed approaches allow you to customize the system to individual customer needs and allow you to quickly expand the functionality regardless of existing ones.

As mentioned in the sections, each function or micro-service corresponds to an independent micro-service, which helps to increase the stability of the system. Despite the fact that the system already has a lot of services to address the basic needs of customers, it has and will expand in commercial projects. This project solves the problem of bureaucracy when generating reports, analyzing trips, creating plans, etc. But the most important aspect it covers is the need to synchronize information between operators and drivers.

As every professional employer wants to provide good working conditions, and in the transport industry there is a growing demand for personnel management, and the staff is expanding every year, there is a need to quickly form trip schedules to avoid conflicts. Conflict situations mean overlapping schedules of routes, as well as emergencies. .

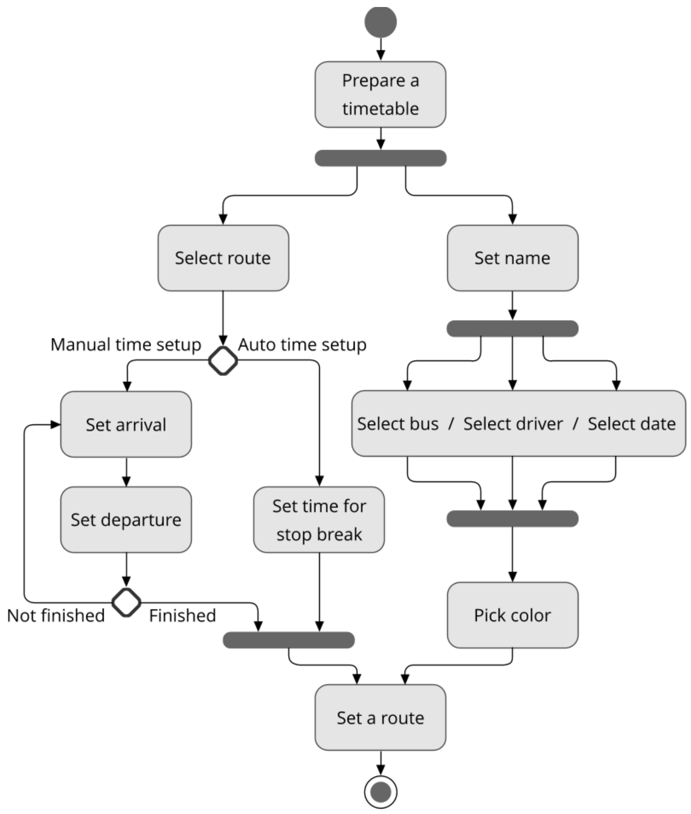

Figure 7: Activity diagram for creating a new schedule

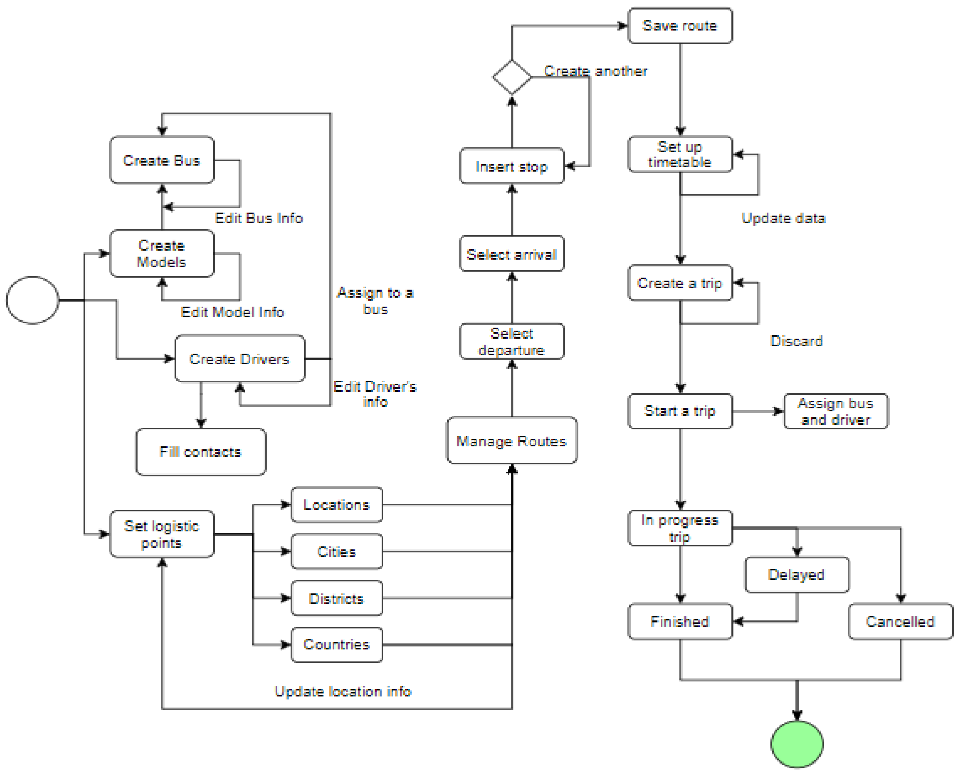

Figure 8: Life cycle. State diagram

It should also be noted that iterative operations were performed to increase optimization and correctness of the system, which led to an increase in possible connections to the tracking. All locations are stored in chunks, and when the client moves, it receives a new image with new data and cached previous ones, which saves memory and makes the application faster. Each route created by the user will be saved in the repository, and when the user tries to build a new route with multiple repetitive routes, this will result in retrieving the saved data instead of calling the API. This saves the client money and affects the performance of the application.

References

[1] M. Karpinski, S. Kuznichenko, N. Kazakova, O. Fraze-Frazenko, D. Jancarczyk.. Geospatial Assessment of the Territorial Road Network by Fractal Method. Future Internet. 12. 201 (2020). DOI:10.3390/fi12110201.

[2] Z. Cortes, J. Andres; A. Serna, M. Dario and A. Gomez.(2013). Information systems applied to transport improvement. Dyna rev.fac.nac.minas. vol.80, n.180. ISSN 0012-7353

[3] Alam, Muhammad & Ferreira, Joaquim & Fonseca, José. (2016). Introduction to Intelligent Transportation Systems. 10.1007/978-3-319-28183-4_1.

[4] A. Nuzzolo and A. Comi, "Advanced public transport and intelligent transport systems: New modelling challenges," Transp. A, Transp. Sci., vol. 12, pp. 674–699, Sep. 2016.

[5] Dinh Dung, Nguyen & Rohács, József & Rohacs, Daniel & Boros, Anita. (2020). Intelligent Total Transportation Management System for Future Smart Cities. Applied Sciences. 10. 8933. 10.3390/app10248933.

[6] Kuznichenko, S.,Buchynska, I.,Kovalenko, L.,Tereshchenko, T. Integrated information system for regional flood monitoring using internet of things. CEUR Workshop Proceedings, 2019, 2683, стр. 1–5

[7] Kuznichenko, S., Kovalenko, L., Buchynska, I., Gunchenko, Y. , Development of a multi-criteria model for making decisions on the location of solid waste landfills. Eastern-European Journal of Enterprise Technologies, 2018. Vol.2, No. 3(92). P. 21–31. DOI: 10.15587/1729-4061.2018.129287

[8] S. Kuznichenko, I. Buchynska, L. Kovalenko, Y. Gunchenko. Suitable site selection using two-stage GIS-based fuzzy multi-criteria decision analysis. Advances in

Intelligent Systems and Computing, 2020, 1080 AISC, стр. 214–230

[9] P. Du, J. Chen, Z. Sun, and Y. Li, "Design of an IoT-GIS emergency management system for public road transport networks," in Proc. 1st ACM SIGSPATIAL Int. Workshop GIS Emergency Manage., Bellevue, WA, USA, Nov. 2015, p. 12.

[10] Robayet Nasim. Architectural Evolution of Intelligent Transport Systems (ITS) using Cloud Computing. Licentiate thesis. Karlstad University Studies, 2015:2.

[11] Ricardo Salazar-Cabrera, Álvaro Pachón de la Cruz, Juan Manuel Madrid Molina, Sustainable transit vehicle tracking service, using intelligent transportation system services and emerging communication technologies: A review, Journal of Traffic and Transportation Engineering, Vol.7, Issue 6, 2020, P. 729-747, ISSN 2095-7564

[12] Kamel, Mohammed B.. (2015). Real-Time GPS/GPRS Based Vehicle Tracking System. International Journal Of Engineering And Computer Science. 10.18535/ijecs/v4i8.05.

[13] Long Polling vs WebSockets vs Server-Sent Events, 2019. URL: https://medium.com/system-design-blog/long-polling-vs-websockets-vs-server-sent-events-c43ba96df7c1

[14] Design model: Use-Case realization with GRASP patterns, 2001. URL: https://www.pearsonhighered.com/assets/samplechapter/0/1/3/0/0130925691.pdf

[15] Design Patterns: Elements of Reusable Object-Oriented Software 1st Edition, Kindle Edition, by Gamma Erich, Helm Richard, Johnson Ralph, Vlissides John