


Information system to support the operation of the car park

Nemanja D. Kaplanović

Serbian Armed Forces, Center for Command and Information Systems and IT Support (CKISIP), Belgrade, Republic of Serbia,
e-mail: knemanja00@gmail.com,
ORCID iD:  <https://orcid.org/0009-0004-6778-7685>

DOI: <https://doi.org/10.5937/vojtehg72-48066>

FIELD: IT

ARTICLE TYPE: review paper

Abstract:

Introduction/purpose: Information systems represent a set of people, data, processes and information technologies related to the purpose of data collection, processing, storage and their filtering into useful information needed for supporting organizations or decision making. The information system is developed as a Web application with the help of the Javascript programming language, Node.js, and React.js, while in the background it relies on data storage and communication with the SQL database.

Methods: Modern web application development tools are used and tested on low budget hardware.

Results: Effective software for a car park with low maintenance cost and great reliability has been developed.

Conclusion: The software made work easier, increased access to data and facilitated data review, reducing the possibility of error. The application described in this article and made for research shows how modern commercial technologies can be used for military purposes.

Keywords: software, solution, car park, JavaScript, node, SQL, military, technology.

Introduction

The constant improvement and expansion of technology has led to the fact that there is almost no sphere (social, economic, military, etc.) that has remained immune to its implementation and application. The potential for application is significant and there are no limits to the depth to which technology will reach into people's lives. The key part is the development of software which people come in contact with in their daily activities and which has become their tool for doing business. In this paper, an Information System will be presented for the needs of operating a car park. (Kaplanović, 2023)

Used technologies

Agile method in IS development

The agile development method consists of phases, but the entire project is executed through cycles (sprints) that have some of the required system functionalities as the end product. Cycles are iterative, adaptive, divided in time, fast and flexible to new requirements or changes. The method requires continuous communication with users and developers who develop the solution. After each cycle, there is a consolidation with the already completed part of the solution from the previous cycles. Unfinished solutions are handed over to users for use and testing at a certain point in time. The received feedback from users affects the modification of the product of a certain cycle (part of the solution) which will be performed in one of the following cycles.

Used information technologies for development

Node.js is an open source environment, implemented in the JavaScript programming language. JavaScript as a relatively new programming language has brought convenience in writing codes because it can be used on both the client side and the server side. Node.js is platform independent and it is intended for the development of scalable network applications. Its architecture is event-driven and contributes to the ability of input/output operations to be executed in real time. It allows writing command line tools and scripts as well as running on the server side. The manager package provides access to a huge database of packages. It also supports boot chaining projects. By using the HTTP1 package, a simple web server can be created very quickly.

Some of the typical web tasks are not directly supported by Node. To add specific request handling, URL paths, or dynamically creating server responses, it is necessary to write a code or facilitate the work by using one of the web frameworks. Express is one of the most popular and minimalist ones. In this paper, it was used for processing HTTP requests (GET, POST, UPDATE, DELETE) from different paths.

Sequelize is a library for Node.js that provides work with relational databases. It uses ORM (Object-Relation Mapping) functionality which manipulates the database using objects and the very methodology of object-oriented programming. The benefit is that there is no need to write raw SQL. It supports work with databases such as MySQL, PostgreSQL, SQLite, and Microsoft SQL Server. The developer defines the models that represent the tables in the database and the relationships between them.

Models are used to create, read, update, and delete (CRUD operations) a data base. This approach allows the same code to be used over and over again and to access any SQL database without changing the query. (Pundsack, 2023)

The React JavaScript library was used to create the user interface. React creates components (tables, buttons, forms, menus) that are modular code pieces and whose community constitutes the user interface. The point is to make such components that they can be used multiple times without retyping their code. It is not necessary to reload the page for changes to occur. React is based on the principle of the virtual DOM (Virtual Document Object Model). It allows monitoring the state of certain components and updating only those that have changed. This allows for better performance and more efficient display updates. The data itself is transferred from the top to the bottom of the hierarchically organized components. This is made possible by using unidirectional binding which also makes it easier to follow the flow of data. (Grebe, 2019)

Analysis of the requirements for an ideal solution

The driving force behind the development of this information system is the desire to facilitate and automate the process of vehicles entering/exiting to/from a military facility, to monitor their conditions and to organize drivers. The information system for supporting the car park operation is designed to consist of two groups: guards and administrators. There is information on each vehicle: vehicle brand, vehicle type, vehicle status, registration number, and VIN number. The guard user group is obliged to keep records, enter data on which driver leaves or enters the facility with which vehicle, with which travel order number, and on which date. The group of administrators is responsible for entering, updating, or deleting data on vehicles, drivers and records, as well as entering user accounts and maintaining them. In the security service, in the analysis of the existing organization, the potential for speeding up work and increasing efficiency was noticed, and it was chosen as a priority for the development of the first version of the application. A client-server architecture was chosen as a solution. A centralized server with a database allows users to simultaneously access data with different devices. (Ellk et al, 2020)

The idea of improving the existing working method

When a vehicle enters a military facility or exits from it, a certain procedure must be followed. The method of operation is as follows:

- the vehicle approaches the gate,

- the guard approaches the vehicle, takes the license plate number from the driver,
- enters the security facility and enters data into the record book,
- returns to the driver and returns the documents, and
- opens the gate for the vehicle to leave the facility.

The mentioned process can be improved based on the following:

- the vehicle approaches the gate,
- the guard approaches the vehicle,
- the guard enters the data into the application (which vehicle, which driver and the number of the travel order) using a portable device, and
- opens the gate and the vehicle leaves the facility.

The importance of the mentioned procedure is noticeable especially when multiple vehicles leave or enter a facility where there is a shortage of available time for performing such actions. The system can also be applied in conditions where the guard does not have to be at all next to the vehicle. The entire process can be monitored from a remote facility.

The suggested procedure is as follows:

- the vehicle approaches the gate,
 - the security service records the driver and the vehicle with a video from the cameras, and
 - remotely opens the gate and the vehicle leaves the building.
- (Fadeev, 2022)

The way of using the developed IS may differ depending on military facilities, their organization and techniques at their disposal. The system requires any device that can access the network where the application is located and which has the ability to work with the Internet browsers. This system enables all the benefits of digitalization such as: authentication and user authorization, authenticity of entered data, insight into the current state, easier access to information, and storage of copies. These benefits are very difficult to achieve by applying the old approach of manual data entry into record books. The concept of an individual at the gate recording who enters and exits is outdated and prevents the security service from performing their duty of guarding the object. Access to data kept in a record book is limited to the shift guard responsible for keeping it. However, data entered into the application with a centralized database are available to authorized users online. This enables users whose duties are related to the organization to autonomously seek information about currently available manpower or technology. Users who need such information do

not have to call the security service, thus disrupting their work and wasting the time of the duty officer who has to browse the records and search for requested information. The diagrams in the text describe the usage of the application. The elements of the diagram are: actors (users), use cases that describe the tasks that the system performs, links that connect actors and use cases. Figure 1 shows a diagram of the use cases. (Ádám et al, 2021)

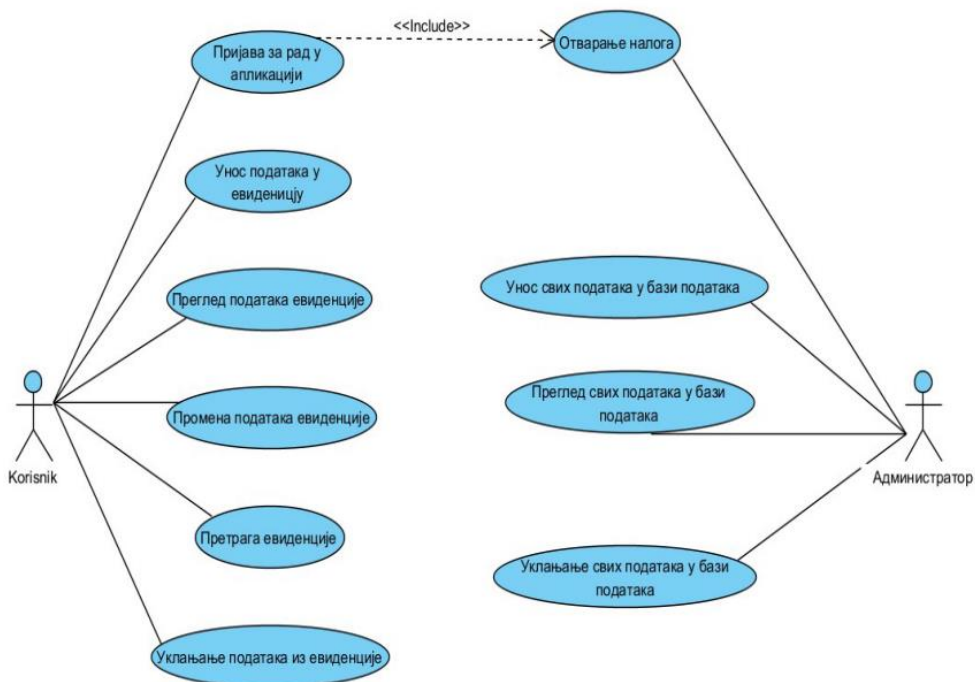


Figure 1 – Use case diagram

An activity diagram consists of different elements:

1. Activities: these are tasks or operations performed in the system. They are represented inside a rectangle and named after the actions they represent.
2. Transitions: show a change in activity and indicate the flow of control or data flow. They are represented by arrows that connect activities and contain conditions related to their execution.
3. Decisions: represent the points in the diagram where the system must make decision on what next step to take. They are marked with

diamonds and contain the conditions that determine which transition will occur.

4. Synchronization: represents events that are executed in parallel, and then connected at one point in the diagram. It is used when necessary to ensure that activities are completed before proceeding to the next step.

5. Initial and final node: the initial node marks the beginning of the activity diagram, while the end point marks the end of the execution of the diagram. Both points are represented by circles.

The activity diagram contributes to a better understanding of complex processes and helps in their implementation, analysis and optimization. The following text describes an activity diagram (Figure 2) of user registration.

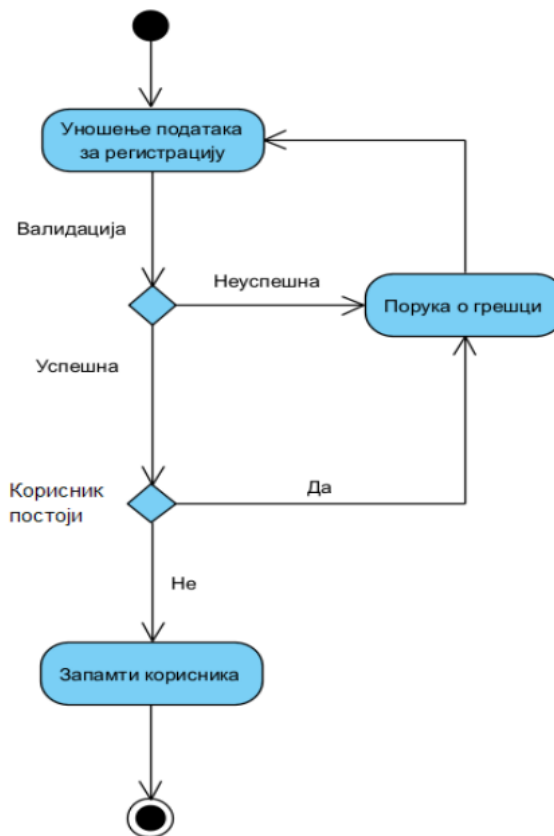


Figure 2 – Activity diagram

The registration activity consists of the following steps:

1. The user enters his username and password into the system.
2. After the username and the password are entered, the system performs a validity check of the entered data. If some of the validation requirements (e.g. password length) are not met, the system displays an error message and prompts the user to re-enter correct credentials. The application performs the check on the client side, thus reducing the task of the server to reject requests that do not meet the condition.
3. If the entered data is valid, the system checks the status of the user in the database data. If the user does not exist in the database, the system updates the database and registers a new user.
4. If a user with the same username already exists in the database, the system displays an error message, informing the user that the account already exists.

The data entry activity is shown in a diagram (Figure 3). The diagram shows the data entry in the record table. The activities are the same for each entry in the application. The following is a description of the diagram from Figure 3:

1. Upon the access to the application, it is checked whether the user is registered for work. If yes, the page for data entry is displayed; if not, the user is redirected to the login page.
2. Enter the data in the input form. Data validation is synchronized with the input. While data entry is ongoing, data validation is ongoing.
3. If the entered data is valid, it is possible to send data to the server.
4. The content of the form is removed. A clean form for a new entry is immediately available thus speeding up user's work.

Figure 4 represents a sequence diagram that shows how the client-side of the application sends a request to the server-side of the application to get the data to display. (Stawowy et al, 2023)

By clicking on the link to display the page, the user sends a request to display it. An example in Figure 4 shows the user accessing the page where the data about drivers is displayed. This is made possible with the help of the `Navigate()` function which assigns the URL for the corresponding page, and calls for the React Router library that routes the users based on URLs. Specific content related to the defined address route is then displayed. The components of a particular page have functions related to the page load event (`onLoad`). The function starts to be executed while loading the bound component, in this case the component in charge of display of driver data (`mainTable`).

The component sends a request to get data using Sequelize. The query is executed in the server part of the application over the database data and returns the requested data as a result. The client side formats and displays the data that the user can see.

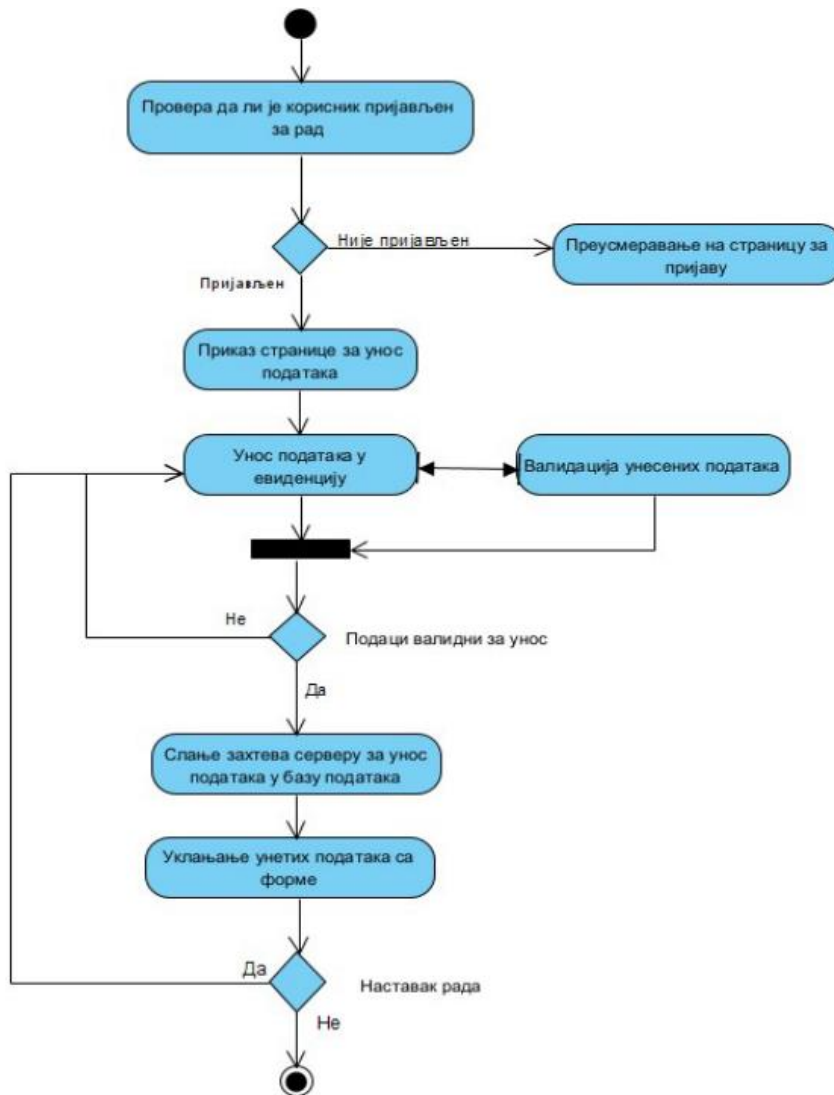


Figure 3 – Diagram of the data entry activities

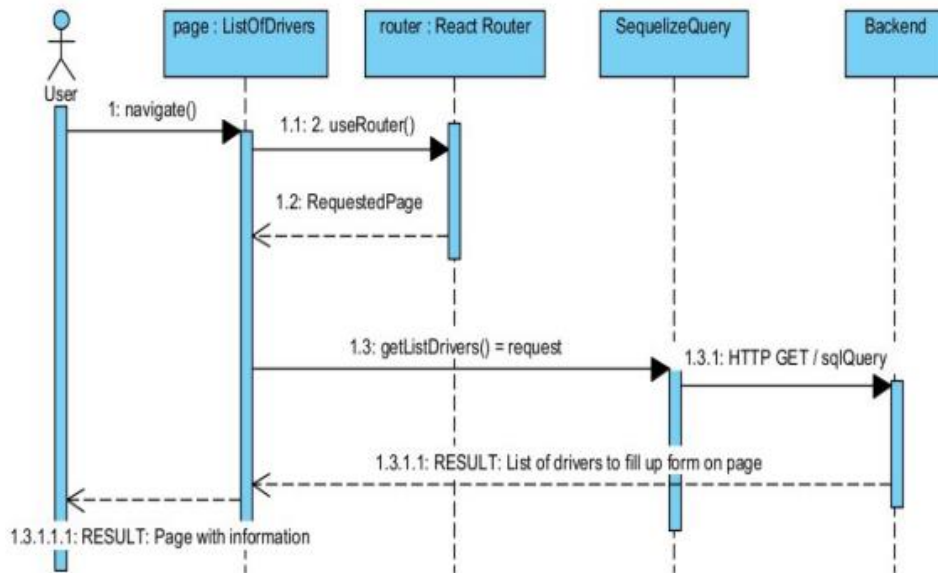


Figure 4 – Diagram of the data filling sequences on pages

The database is created in MySQL, a free open-source software. The chosen system has proved to be highly manageable in data manipulation, fast and efficient. The big advantage is that it can work on multiple platforms (Windows, Linux, macOS, etc.) where the independence from the platform for the operation of the application and its configuration has been acquired.

In Figure 5, a logical model of the database is shown. A logical model consists of entities, attributes, and relationships between them. This model is an approximate representation of a system or process that serves to understand the system and its changes and managing methods. In the relational database, the main entities presented are Vehicles, Records, and Drivers.

The other entities refer to vehicle types, vehicle statuses, vehicle brands and ranks, representing codes that are not commonly changed.

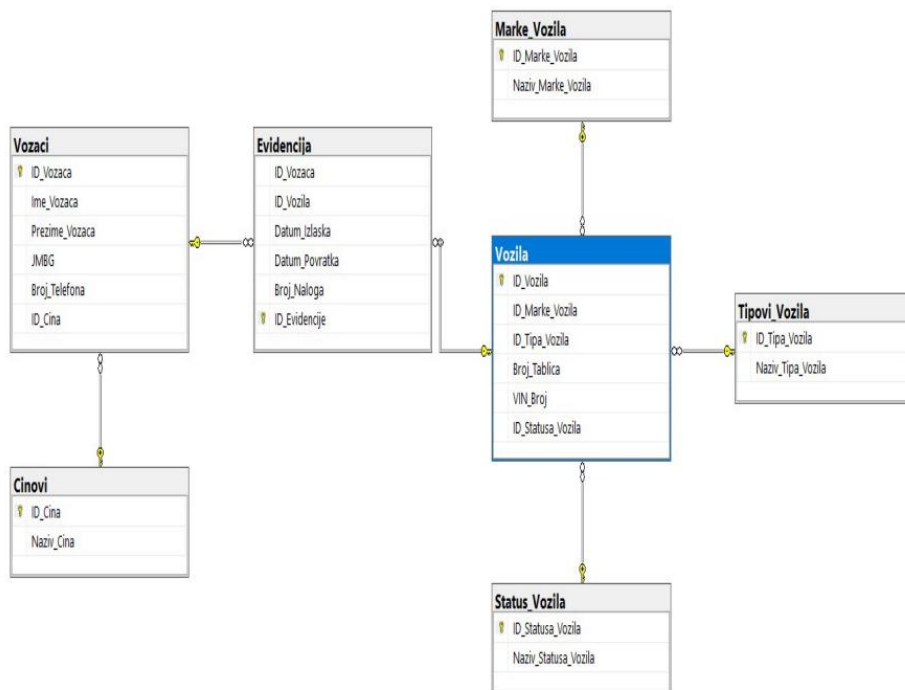


Figure 5 – Database model

Implementation of solutions

Application solution implementation is the process of translating the design and application requirements into a real code and functionalities that work within the web environment. This includes client and server side application programming, database usage, environment configuration and testing. The development of the user interface and functionality on the user's browser are part of the client-side application. The user interface is an important aspect of every application because most users rate the application mostly based on of their subjective feeling gained by using it. Users get in touch with the application only through the interface. It is important that the user's work itself be intuitive. Whether or not an app gets adopted depends more on its design than on its functionality. In practice, applications with poorly designed user interfaces rate much lower even though they have more functionality or a higher speed of operation than those applications that are appealing to the eye. To create the user interface of the test application, React.js was used.

Another part of the solution is programming the server side of the application. Node.js was used for the development of communications with the database, processing of client requests, and data processing. Security is one of the priorities of developing server-side applications. The data is the most valuable resource that attracts potential attackers. The degree of security and the level of investment in application security depends on the data value. Of course, it is important to implement protection on the client side as well, but only after implementing adequate protection on the server side. There are many examples of bypassing client-side protection where attackers manage to send written requests and talk to the server without authorization. This leads to attackers coming into contact with the data and the database itself without being authorized. There is a data leak, sabotage of the application, and it can also result in hardware damage.

Figure 6 shows an example of the model implementation for Sequelize. To create a model, it is necessary to define a table with the attributes. Each attribute is defined by the properties such as data type, whether it can be "null", whether it is a primary key or a foreign key for the table, etc. When working with relational databases, it is also necessary to define relations over certain attributes in the model. The relations that Sequelize supports are basic and refer to one-to-one, one-to-many and many-to-many. For defining these relations, it is necessary to combine the four associations that Sequelize offers: HasOne, BelongsTo, HasMany, BelongsToMany. In the example in Figure 6, in the first part of the code, under the method `sequelize.define()`, the primary key `IDVozaca` is defined in the `Vozaci` model. Once the model definition is complete, its relations with other models are established. `Vozaci.associate()` is a function that defines the relationship between models over certain attributes.

An example of the code for processing requests on the server side is given in Figure 7. The import of the `MarkeVozila` model provided the methods that come with Sequelize and make it easier to work with the database without writing queries but by calling the methods on models that change queries. The methods shown in Figure 7 are: `findAll()`, `create()`, and `findByPk()`.

In Figure 7, the input requests (POST request) in the database and data withdrawal are processed (GET request) from the database. Asynchronous functions are used, which as a result return data to the JSON format.

```

1  module.exports = (sequelize, DataTypes) => {
2    const Vozaci = sequelize.define("Vozaci", {
3      IDVozaca: {
4        type: DataTypes.INTEGER,
5        allowNull: false,
6        autoIncrement: true,
7        primaryKey: true,
8      },
9      ImeVozaca: {
10     type: DataTypes.STRING(20),
11     allowNull: false,
12   },
13   PrezimeVozaca: {
14     type: DataTypes.STRING(20),
15     allowNull: false,
16   },
17   JMBG: {
18     type: DataTypes.STRING(13),
19     allowNull: false,
20   },
21   BrojTel: {
22     type: DataTypes.STRING(12),
23     allowNull: false,
24   },
25   });
26
27   Vozaci.associate = function (models) {
28     Vozaci.belongsTo(models.Cinovi, {
29       foreignKey: {
30         name: "IDCina",
31         type: DataTypes.INTEGER,
32         allowNull: false,
33       },
34       onDelete: "CASCADE",
35       onUpdate: "CASCADE",
36     });
37     Vozaci.hasMany(models.Evidencija, {
38       foreignKey: {
39         name: "IDVozaca",
40         type: DataTypes.INTEGER,
41         allowNull: false,
42       },
43       onDelete: "CASCADE",
44       onUpdate: "CASCADE",
45     });
46   };
47
48   return Vozaci;
49 };
50

```

Figure 6 – Sequelize example of the Vozaci (Drivers) model

```

const express=require('express');
const {MarkeVozila}=require('../models');
const router=express.Router();

router.get("/", async(req,res)=>{
  //res.send('Hi');
  const listOfMarke=await MarkeVozila.findAll({
    order:[['NazivMarke','ASC']]
  });
  res.json(listOfMarke);
});

router.post("/", async(req,res)=>{
  const data=req.body;
  await MarkeVozila.create(data);
  res.json(data); //potvrda
});

router.get("/edit/:id",async(req,res)=>{
  const id=req.params.id;
  const mark=await MarkeVozila.findByPk(id); //findOne({where:{idMarke:id}})
  res.json(mark);
})

```

Figure 7 - Sample request processing code

Application operation

At the start of the application, the home page is displayed. The home page contains a message and a login button. Its role is to inform the user who accesses it that the system is operational and in function. When being logged out of the account, the user will be automatically redirected to the home page. (Jakovljević et al, 2022)



Figure 8 – Welcome screen

By clicking the "Log in" button, the user is redirected to the login page on which there are the fields for entering the username and the password. If valid data is entered, the user is redirected to the home page for work. Attempts to log in with incorrect information will result in displaying a warning to the user (Figure 9) and the user remains on the login page.

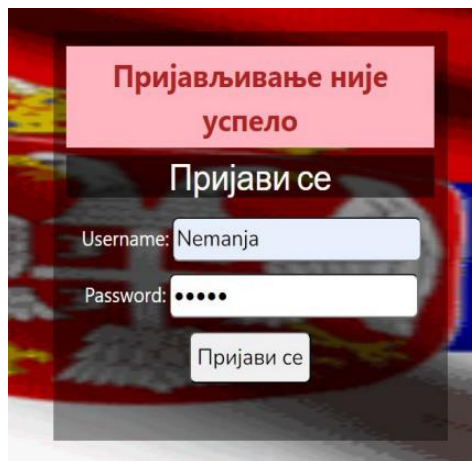


Figure 9 – Log In tab

The appearance of the home page after login differs from the appearance when the user logs into the application. This is achieved by following the application based on the type account and its privileges show the options available to user in the navigation menu through the application. In the continuation of the text, the images of the menus for privilege users are given as "Administrator" (Figure 10) and as "Duty Officer" (Figure 11). If the user tries to access the page, by manually entering the URL, for which there is no privilege, the application will redirect him to the home page.

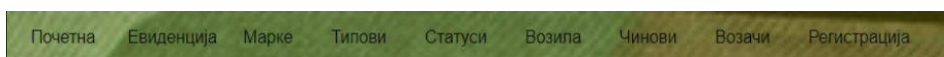


Figure 10 – Administrator menu

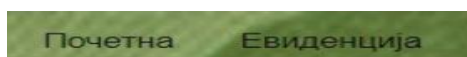


Figure 11 – Duty officer menu

By clicking on the menu option, the user goes to the page for working with the selected data. The key functionality of the application is recording the entry or exit of the vehicle, and it is accessed through the "Evidencija" option in the menu (Figure 12).

The records page displays a sub-menu containing functionalities related to working with data related to the selected page - on the page for working with records, it is the "Unesi novu evidenciju" option.

The data in the records are displayed in the form of a table with columns and rows. The data that require attention are marked in red. Each row, in addition to the columns that display the information about the data, also contains a column with a button for changing and deleting data. When the driver finishes task and returns, its return time is entered by pressing the "Izmeni" button. After the return time is entered, the data will no longer be displayed in red. Pressing on the "Brisi" button opens a window for deleting the data. Data search is enabled by the "Pretraga" field in which the user enters a term for search. It is not necessary to select a search category because the search is performed through all categories.

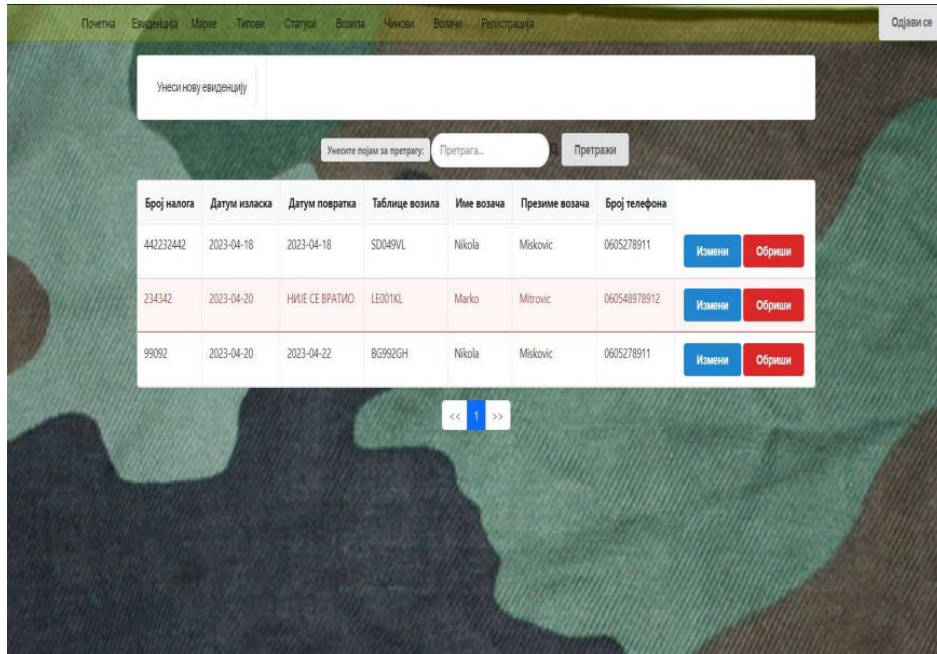


Figure 12 – „Evidencija“ page

The layout of the pages for displaying data, entering new data, changing and deleting data about drivers, vehicles, vehicle types, status of vehicles, brands of vehicles and their ranks follows the same template as the pages for working with records.

Pressing the "Unesi novu evidenciju" option opens a page to enter data in the records. Filling in the fields is done by choosing one of the offered options or manually. The validation of the entered data is performed when the "Enter" button is pressed. A validation example in Figure 13 shows an appropriate message with a description of the error that needs to be corrected to enter the data.

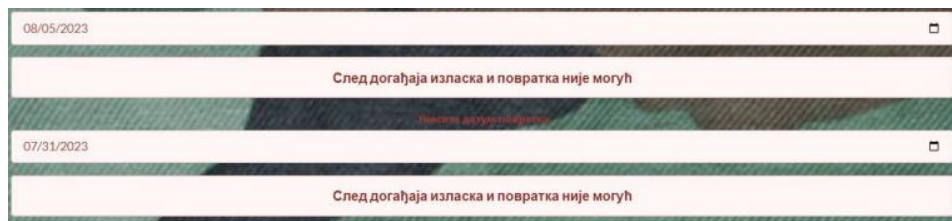


Figure 13 – Fields to enter new data and errors

Possibility of further expansion of the system

The conceptual solution was given and realized by the author; therefore, there is room for refinement of current functionalities and addition of new ones. To begin with, it is necessary to form a group of different specialists in order to carry out the assessment of the application and its testing in real conditions. After that comes a creation of a newer version of the application that would represent a revision of the existing one. It is desirable that the same expert group continue improving the application and at the same time conduct training for the application maintenance. Further extensions may refer to the introduction of new account types such as "Mechanic". The role of the mechanic would be to enter data about functioning, defective and available vehicles. Thus, the solution would include other structures. The ultimate goal would be a system that links together the entire work of a military facility. It would enable the entry of employees who come to work on a particular day. There would be an exact list of people and vehicles currently available, thus the person responsible for assigning tasks can easily connect available drivers with ready and free vehicles and assign them a task. And guards themselves would find it easier to record the entry and exit of vehicles. Monitoring the operation of one facility would make work easier and more organized. The idea that would arise from this way of solving the problem would influence the improvement of work automation. What complete automation means is that all human actors are replaced by sensors. The solution is presented below. The driver comes to work with his own identification card (ID) which contains a chip, and he scans the ID thus becoming visible for task assignment. The mechanic ensures that the vehicles condition data are accurate and entered in time. The commander connects the driver and the vehicle through the application and assigns a task. The driver goes to perform the task, takes over the assigned vehicle and drives to the exit gate. A camera with a sensor reads the license plates of the vehicle and compares them with the available vehicles, the driver approaches the point and reads his ID. The system checks the camera data and the ID data, verifying whether there is a task assigned to that particular driver with that particular vehicle. If it finds a match, the gate opens and the vehicle leaves. The information about that exit appears in the records. When the vehicle returns, it repeats the same procedure as when exiting. The camera reads the license plates, the driver is authenticated via the reader of smart cards. If there is a match in the records that that driver with that vehicle left the facility, the gate opens, the vehicle enters and the records become completed with his return time. (Karakebelioglu et al, 2021)

Conclusion

The application developed aims to improve the current method of executing the actions related to the process of vehicle entry and exit from a military facility. Developing modern web applications that help with everyday activities is at the very top of people's interests. Adaptive applications such as those created in web technologies are intended for any device type or screen size. The advantage is that they do not need to be already installed: all that is necessary is a web browser and access to the network where the application is located. The goal of this paper is to bring people closer to the idea of using smartphones or other smart devices for something useful at their work.

References

- Ádám, A.B., Kocsány, L. & Szádeczky-Kardoss, E.G. 2021. Using coverage path planning methods for car park exploration. *Acta IMEKO*, 10(3), pp.15-17. Available at: https://doi.org/10.21014/acta_imeko.v10i3.1021.
- Elk, D.S., Mohsin, A.H. & Hasan, S.A. 2020. Feasibility Study for the Establishment of a Multi-Story Car Park, A Case Study. *The Open Civil Engineering Journal*, 14, pp.179-187. Available at: <https://doi.org/10.2174/1874149502014010179>.
- Fadeev, I.V. 2022. Car park as a definitive factor development of car service system. *Vestnik RGATU*, 14(1), pp.159-167. Available at: <https://doi.org/10.36508/RSATU.2022.89.86.018>.
- Grebe, S. 2019. *Hands-On Full-Stack Web Development with GraphQL and React: Build scalable full-stack applications while learning to solve complex problems with GraphQL*. Birmingham, UK: Packt Publishing. ISBN: 978-1789134520.
- Jakovljević, I., Spremić, M. & Marković, Z. 2022. Methods for Life Extension of Multi-Storey Car Park Buildings. *Structural Engineering International*, 33(2), pp.314-324. Available at: <https://doi.org/10.1080/10168664.2022.2073318>.
- Kaplanović, N. 2023. *Informacioni sistem za podršku rada auto parka*. BS thesis. Belgrade, Serbia: University of Defence (in Serbian).
- Karakebelioglu, A.F., Eren, O., Koten, H., Alp, H. 2021. Designing and Analyzing Park Sensor System for Efficient and Sustainable Car Park Area Management. *Sustainable Engineering and Innovation*, 3(1), pp.44-48. Available at: <https://doi.org/10.37868/sei.v3i1.id123>.
- Pundsack, M. 2023. Viewpoint: Learning lessons on fire-safe design of car parks. *The Structural Engineer*, 101(11), November 13. Available at: <https://doi.org/10.56330/YIDI5544>.
- Stawowy, M., Rosiński, A., Perlicki, K., Wilczewski, G. & Czarnecki, T. 2023. Estimating the Measurement Uncertainty of the Number of Vehicles in a Car Park Using an Indirect Method. *Applied Sciences*, 13(10), art.number:5938. Available at: <https://doi.org/10.3390/app13105938>.

Система de información de apoyo al funcionamiento del estacionamiento de coches

Nemanja D. Kaplanović

Fuerzas Armadas de Serbia, Centro de Mando e Información Sistemas y soporte de TI (CKISIP), Belgrado, República de Serbia

CAMPO: IT

TIPO DE ARTÍCULO: artículo de revisión

Resumen:

Introducción/objetivo: Los sistemas de información representan un conjunto de personas, datos, procesos y tecnologías de la información relacionados con el propósito de la recopilación, el procesamiento, el almacenamiento de datos y su filtrado en información útil necesaria para apoyar a las organizaciones o a la toma de decisiones. El sistema de información se desarrolla como una aplicación Web con ayuda del lenguaje de programación Javascript, Node.js y React.js, mientras que en segundo plano se basa en el almacenamiento de datos y la comunicación con la base de datos SQL.

Métodos: Se utilizan y prueban herramientas modernas de desarrollo de aplicaciones web en hardware de bajo presupuesto.

Resultados: Se ha desarrollado un software eficaz para un estacionamiento de coches con bajo coste de mantenimiento y gran fiabilidad.

Conclusión: El software facilitó el trabajo, aumentó el acceso a los datos y facilitó la revisión de los mismos, reduciendo la posibilidad de error. La aplicación descrita en este artículo y realizada con fines de investigación muestra cómo las tecnologías comerciales modernas se pueden utilizar con fines militares.

Palabras claves: software, solución, estacionamiento, JavaScript, nodo, SQL, militar, tecnología.

Информационная система управления автопарком

Неманя Д. Капланович

Вооруженные силы Республики Сербия, Командно-информационный центр и ИТ-поддержка (ЦКИСИП), г. Белград, Республика Сербия

РУБРИКА ГРНТИ: 20.23.25 Информационные системы с базами знаний

ВИД СТАТЬИ: обзорная статья

Резюме:

Введение/цель: Информационные системы представляют собой: совокупность персонала, данных, процессов и информационных технологий, связанных с целью сбора, обработки, хранения данных

и их филтрации в полезную информацию, необходимую для поддержки организаций или принятия решений. Информационная система разработана в качестве веб-приложения с помощью языка программирования Javascript, Node.js, и React.js, в то время как в фоновом режиме она полагается на хранение данных и связь с базой данных SQL.

Методы: В данной статье на бюджетном оборудовании применены и протестированы современные инструменты разработки веб-приложений.

Результаты: Разработано эффективное программное обеспечение для автостоянки с низкими затратами на обслуживание и высокой надежностью.

Выводы: Разработанное программное обеспечение облегчило работу, расширило доступ к данным и упростило их просмотр, снизив вероятность ошибки. Приложение, описанное в данной статье, было специально разработано для исследований. Оно доказывает, что современные коммерческие технологии могут применяться в военных целях.

Ключевые слова: программное обеспечение, решение, автостоянка, JavaScript, узел, SQL, вооруженные силы, технология.

Информациони систем за подршку рада ауто-парка

Немања Д. Капановић

Војска Србије, Центар за командно-информационе системе и информатичку подршку (ЦКИСИП), Београд, Република Србија

ОБЛАСТ: информационе технологије

КАТЕГОРИЈА (ТИП) ЧЛАНКА: прегледни рад

Сажетак:

Увод/циљ: Информациони системи представљају скуп људи, података, процеса и информационих технологија чији је циљ прикупљање, обрада, складиштење података и филтрирање у корисне информације потребне за подршку организацији или доношење одлука. Информациони систем је развијен као веб апликација уз помоћ програмског језика Javascript, Node.js, React.js, а у позадини се ослања на складиштење података и комуникацију са СКЛ базом података.

Методе: Користе се и тестирају савремени алати за развој веб апликација на нискобуџетном хардверу.

Резултати: Развијен је ефикасан софтвер за паркинг са ниским трошковима одржавања и великом поузданошћу.

Закључак: Софтвер је олакшао рад, повећао приступ подацима, олакшао њихов преглед и смањио могућност грешке. Описана апликација намењена је истраживању и показује како се модерне комерцијалне технологије могу користити у војне сврхе.

Кључне речи: софтвер, решење, ауто-парк, JavaScript, node, SQL, војска, технологија.

Paper received on: 04.10.2023.

Manuscript corrections submitted on: 04.03.2024.

Paper accepted for publishing on: 05.03.2024.

© 2024 The Author. Published by Vojnotehnički glasnik / Military Technical Courier (www.vtg.mod.gov.rs, втг.мо.унр.срб). This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/3.0/rs/>).

