International Journal
Information Technology Applications (ITA)

Volume 14, Number 2, December 2025

DESIGN AND IMPLEMENTATION OF A PARKING MANAGEMENT WEB APPLICATION

Filip Žemla, Ján Cigánek

Abstract:

The purpose of this paper is to design and implement a web application for parking lot management. The application will contain three types of users: customer, parking lot owner and administrator. After successful registration, the user will log in to the application and will be shown content according to the role to which he belongs. The customer will be able to create a car, modify an existing car and reserve a parking space with online payment. The parking lot owner will be able to create parking lots and then manage them. The administrator's task will be to approve parking lots that will or will not be displayed in the system after his action. The main goal of this paper is to create a functional web application that solves the problem of parking lot management through an online system. The implementation is carried out using the Angular 15 framework and Firebase technology, using the Firestore database. This paper aims to introduce the online parking system in more detail and provide users with a convenient and effective solution for finding and booking parking spaces. Using this application, it is expected to improve the overall parking process, increase the availability of parking spaces and minimize time delays. The web application will contribute to smoother traffic and better use of available parking spaces.

Keywords:

Angular, Firebase, Firestore, parking management, web application.

Introduction

Urban mobility is undergoing a significant transformation driven by the integration of digital technologies into everyday services. One such area where digitization can bring substantial improvement is parking management. In densely populated cities, finding an available parking space remains a frequent source of stress, inefficiency, and traffic congestion. Traditional parking systems often lack real-time availability, reservation options, or integrated online payments [6].

While international solutions like Parclick or ParkingTag offer advanced features, many local applications in Slovakia still rely solely on SMS payments, lack persistent user accounts, and provide limited interactivity. This creates a need for a more comprehensive, user-oriented solution that supports both drivers and parking lot owners [1, 2].

The aim of this study is to design and implement a functional web-based parking management system. The application supports three types of users: customers (who reserve and pay for parking), owners (who register and manage parking lots), and administrators (who approve new listings). The system allows real-time interaction, integrates an online payment gateway (Stripe), and visualizes parking locations using interactive maps [8].

The solution is developed using the Angular 15 framework for the frontend and Firebase (with Firestore) as the backend and real-time database. This combination ensures high compatibility, fast development, and scalability [5, 14].

The rest of this paper is organized as follows: Section 1 discusses the state of the art in online parking systems. Section 2 outlines the architecture, design and validation of the proposed solution. Section 3 presents the discussion of results and experiences. Final section concludes the paper and discusses possible future improvements.

1 State of the Art

The growing pressure on urban infrastructure, combined with the increasing number of personal vehicles, has intensified the need for intelligent and efficient parking solutions. In the past decade, numerous web and mobile applications have been developed to support parking management by enabling users to search, reserve, and pay for parking spaces. These systems vary widely in functionality, technical complexity, and regional focus [12, 13, 14].

Internationally, applications such as Parclick and ParkingTag provide users with the option to reserve parking spots in advance, thereby reducing stress and improving the convenience of urban mobility. Parclick, available in over 250 cities, offers details on pricing, reviews, and proximity to various destinations. However, it lacks real-time availability tracking. Similarly, ParkingTag, which is popular in Ireland, enables mobile-based reservations and payments but does not include features such as dynamic occupancy monitoring or persistent user profiles [1, 2].

In the Slovak market, the most used systems are Upay and BPS (Bratislava Parking System). These applications rely almost exclusively on SMS-based payment methods and do not offer user accounts, online reservations, or interactive maps. The user experience is limited, and modern authentication or access control mechanisms are absent (Tab.1) [3, 4].

Table 1. Summarizes the core features of the most relevant web applications for parking [1, 2, 3, 4].

Application	Login System	Real-Time Availability	Online Payment	Reservation
Parclick	✓	X	✓	✓
ParkingTag	✓	X	✓	✓
Upay	X	X	SMS only	SMS only
BPS	X	X	SMS only	SMS only

These findings align with previous research which highlights significant disparities between Central and Western European markets. While Western applications often incorporate user personalization, analytics, and real-time data synchronization, Slovak services remain basic in nature and are typically limited to one-time transactions.

From a technological perspective, modern parking systems increasingly utilize services such as Firebase for real-time database synchronization, Stripe for seamless cashless payments, and frontend frameworks like Angular, React, or Vue.js to ensure responsive and scalable interfaces. NoSQL databases, such as Firestore and MongoDB, are favored for their flexibility in storing complex and nested data structures—essential for handling entities like vehicles, reservations, and parking spaces [5, 13].

User authentication is typically handled via solutions like Firebase Authentication or OAuth2, allowing secure access management and role-based permissions. Applications also commonly integrate interactive map libraries such as MapTiler, Leaflet, or Google Maps API to provide spatial context to users. Reactive programming, through libraries like RxJS, enables efficient handling of asynchronous event streams and time-sensitive operations such as reservations or live updates [5, 12].

Despite the availability of these technologies, no existing Slovak solution offers a comprehensive platform that includes unified user identity management, persistent vehicle profiles and booking history, real-time visualization of parking space occupancy, role-based administration and a modern frontend/backend architecture built on open technologies.

This gap highlights the motivation for developing a new solution presented in this paper—a full-featured, modular, and cloud-integrated parking management web application tailored for the Slovak market but compatible with global standards.

2 Architecture and Design

The primary aim of this research was to design, develop, and validate a modular web-based application for parking lot management, focused on usability, real-time interaction, and scalability. The application was built as a prototype with the potential for real-world deployment in urban environments, offering digital services to customers, parking lot owners, and system administrators.

a) Research Object and Objectives

The research focused on building a system that allows:

- **Customers** to register vehicles, browse parking lots on an interactive map and reserve available spots with integrated online payment.
- **Owners** to define and manage their parking areas, including visual boundary mapping and availability tracking.
- **Administrators** to approve new parking entries and monitor all system activity.

The key objective was to implement all core features using modern, open, cloud-native technologies, with minimal backend overhead and maximum user responsiveness. The emphasis was placed on:

- Separation of user roles and permissions.
- Real-time data synchronization.
- Map-based interaction and reservation interface.
- Stateless, serverless backend powered by Firestore and Firebase.

b) Backend and Data Management

The backend was realized entirely using **Google Firebase**, with **Cloud Firestore** as the central data repository. Firestore enables flexible document-based data modeling, real-time listeners for dynamic updates, and cloud-hosted rule-based access control. The data model consists of five main Firestore collections:

- **Users** – manages account identity, role flags (isCustomer, isAdmin) and user ID references.
- **Cars** – each document represents one registered vehicle, linked to its owner via userId, and stores details such as spz, brand, contact data, and status.
- **ParkPlaces** – contains all parking lots created by owners, including fields for name, coordinates (lat, lng), pricing, approval status (parkApprove), and a list of slots with occupancy status and reservation metadata.
- **MapData** – stores arrays of geolocation points (lat, lng) forming polygon boundaries for each parking lot, used for rendering.
- **Reservation** – tracks booking entries with links to vehicle, user, selected lot, slot number, and timestamps.

A full Firestore schema sample is shown in (Fig.1, 2), demonstrating how documents and subcollections reflect the real-world structure of parking operations.

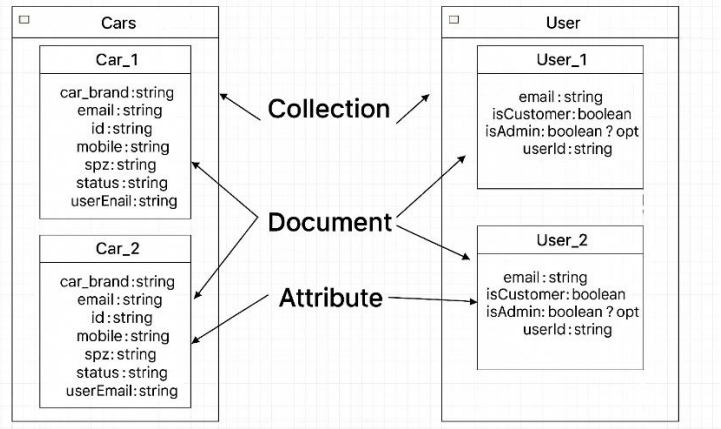

Fig.1. Cars and User collection preview.

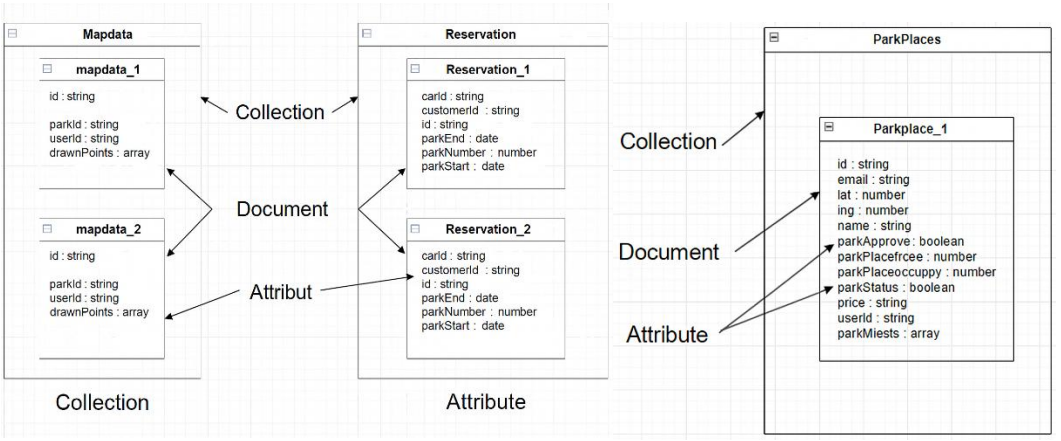

Fig.2. Sample Mapdata and Reservation and ParkPlaces collection.

c) Authentication and Role Management

User authentication is provided by **Firebase Authentication**, offering email/password registration and Google login. After successful sign-up, users confirm their identity via email and are then assigned a role in Firestore (isCustomer, isAdmin) as stored in the Users collection. The frontend application includes a registration form and login interface (Fig.3). If a user does not confirm their email, access to the system is denied. Security rules in Firestore strictly define who can access or modify which data:

- Customers may read and write their own cars and reservations.
- Owners may only access their ParkPlaces and associated data.
- Administrators may read and approve all park entries, and toggle their visibility or status.

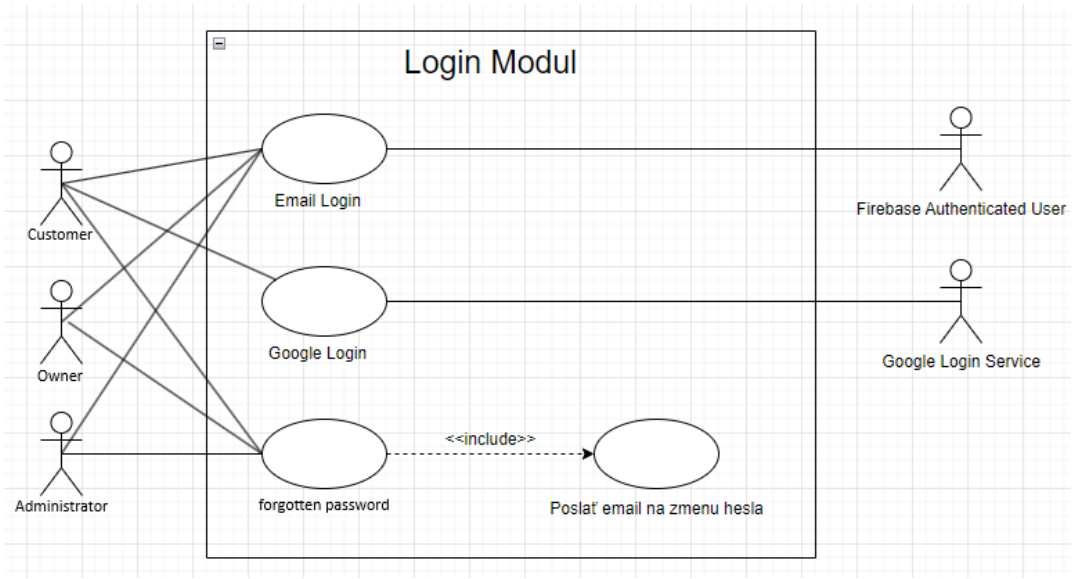

Fig.3. Use case diagram for Login Module (text: *Send e-mail to change password*).

No server-side session management is required; role-based access is handled via authenticated requests and Firestore's rule engine.

d) User Roles and Functionality

The system defines three primary user roles:

- **Customer**: Can register, log in, create one or more vehicles, and reserve available parking slots for specific time periods. Reservations are visualized in the map interface and stored in Firestore.
- **Owner**: Has access to a custom dashboard for managing created parking lots. Owners may create new lots, define availability, and visualize their area using MapTiler. They can temporarily deactivate lots or edit their configuration.
- **Administrator**: Manages the approval process for new parking lots, monitors reservations system-wide, and controls visibility or status of entries.

e) Frontend Implementation and UX Design

The entire frontend is implemented in **Angular 15**, taking advantage of reusable components, routing modules, and reactive forms. Authentication and data services are implemented as shared modules, while feature-specific components are organized by user role. Map interaction is handled using **MapTiler Maps API**, integrated with drawing tools and custom layers. Users can interact with markers, boundary outlines, and tooltips rendered directly from Firestore data.

The application supports responsive design, ensuring usability across both desktop and mobile browsers. For payment integration, the **Stripe** API is used. After reserving a parking space, users are directed to the payment module where they complete the transaction.

f) Functional Implementation and Use Case Mapping

The application supports use cases for each role, validated via diagrams:

- **Customer Use Case**: Includes registration/login, car creation, reservation flow and interaction with map view.
- **Owner Use Case**: Covers parking lot creation, visualization, availability updates and status management.
- **Admin Use Case**: Focuses on review and approval processes, reservation monitoring and map-based moderation tools.

Each of these roles accesses different interface modules and components based on authentication state and role flags. All logic was implemented directly in the Angular frontend, using **RxJS** for observable state handling and **AngularFire** for interfacing with Firestore and Firebase Auth. Application structure emphasizes modularity, with logical separation between shared services, form components, and user-specific views.

g) Evaluation and Validation

The developed application underwent functional validation across all defined roles and use cases. The validation was primarily scenario-based, covering:

- **Customer workflows**: successful registration, email verification, car creation and editing, map interaction, time-slot-based reservation, and online payment completion.
- **Owner actions**: creation of parking lots using interactive drawing tools, saving metadata and coordinates, activating and deactivating parking areas, and monitoring usage.
- **Administrator processes**: reviewing pending parking entries, inspecting layout on map with zoom-to-feature functionality and approving or disabling selected lots.

Each workflow was tested across multiple devices and browsers, ensuring responsive design and data consistency. Real-time updates were specifically evaluated by performing concurrent operations in two browser sessions—e.g., booking a parking space and observing the immediate update in the map view.

The system also proved robust in offline/online switching scenarios, thanks to Firestore's built-in cache and sync capabilities. All user actions are either instantly synchronized or queued for execution once connectivity is restored.

h) Architecture and Scalability

The architecture was designed to remain scalable and maintainable. By relying entirely on **serverless architecture** with Firebase and Firestore, the system avoids backend server maintenance, database provisioning, or manual API scaling. Cloud functions can be introduced later if custom server-side logic is needed (e.g., automatic cleanup of expired reservations or analytics). The frontend is fully modular, with Angular routing separating logical modules:

- /login – login and registration interface,
- /customer – dashboard for users with role Customer,
- /owner – dashboard and parking management tools for Owners,
- /admin – administration panel with approval interface and full-map access,

- /map – public interactive view with parking data from Firestore,
- /reservation – reservation form and payment process.

Lazy-loading and shared modules ensure that unused components are not loaded unnecessarily. Firebase hosting with CDN ensures fast delivery of static assets. The app structure allows for future expansion, including:

- internationalization (i18n),
- mobile-native wrapper using Ionic or Capacitor,
- analytical dashboards for usage statistics,
- parking prediction models via machine learning.

i) User Experience and UI Design Considerations

From the start, the system was designed with simplicity and intuitiveness in mind. Map-based selection is central to the experience. Users can click on parking lots, view real-time occupancy, and start a reservation in a few clicks.

Forms were kept minimal and responsive. Input validation, auto complete, and visual feedback mechanisms guide users through actions such as car registration or booking selection. Administrators benefit from embedded previews and click-to-center map features during approval. Owners can visualize the boundaries they drew using intuitive polygon overlays.

Stripe integration was structured as a simple multi-step flow. After selecting a parking time window, the user is prompted to enter payment details and is notified upon confirmation. Color scheme, contrast, and layout were chosen to ensure readability and accessibility, following WCAG 2.1 Level AA principles.

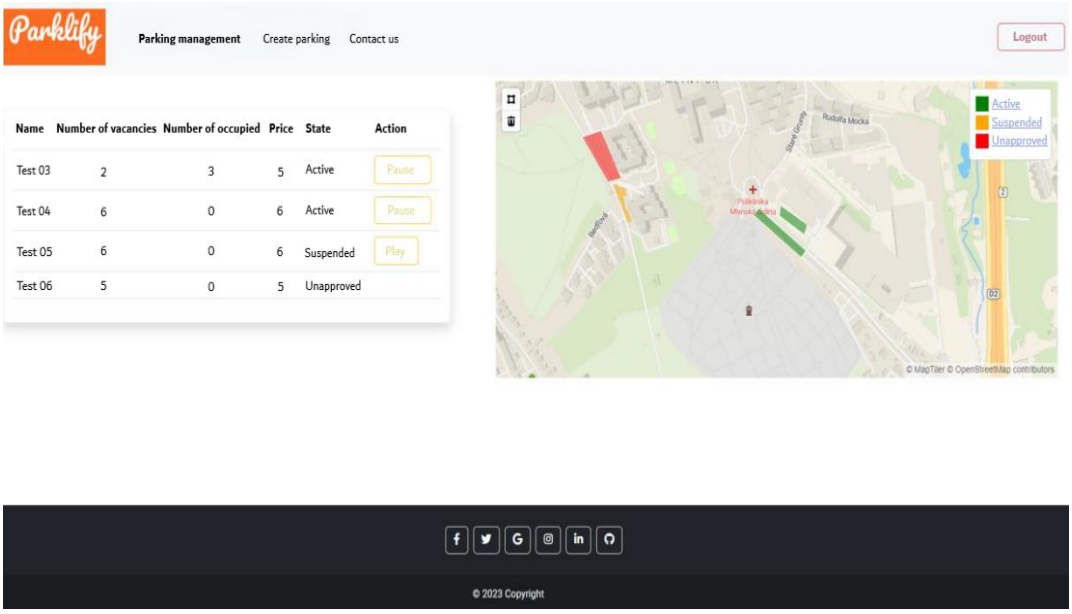

Fig.4. Parking lot management page from the parking lot owner's perspective.

j) Technical Challenges Encountered

Although Firebase and Angular offer deep integration, several technical challenges were encountered:

- **Security rule debugging**: Complex Firestore rules required repeated testing to strike a balance between safety and flexibility, especially for nested access (e.g., ensuring owners cannot access each other's lots).
- **Stripe API mismatch**: Initial problems with missing client-side Stripe scripts and environment variables caused payment issues until the full integration flow was corrected and npm install stripe was applied.
- **MapTiler API limits**: The free tier of the map service imposed certain constraints, key-based authentication and rate-limiting required code optimizations to avoid overuse.
- **Data structure design**: Embedding availability slots directly within ParkPlaces documents saved reads but made updates more complex. A trade-off was made between normalization and performance.

Despite these issues, all problems were resolved using built-in debugging tools, Firestore emulators, and online documentation.

3 Discussion of Results and Experiences

The implemented web application successfully fulfills the initial design goals, offering a comprehensive and user-friendly system for parking management. During development and testing, the system proved effective across all defined user roles, enabling customers to reserve parking spaces, owners to manage their lots, and administrators to control visibility and compliance.

One of the major advantages observed was the seamless integration of Firebase with Angular. Thanks to real-time data synchronization provided by Firestore, any changes to parking space availability are immediately reflected in the user interface. This capability is particularly useful in dynamic environments where multiple users interact with the system concurrently. The ability to manage state in real time significantly improves the user experience and reduces friction in the reservation process.

The modular structure of the applications supported by Angular's component-based architecture-facilitated clean separation of concerns and reuse of code. For instance, form components used for login, registration, and reservation share similar logic while remaining easily customizable. This approach proved valuable during iterative development and testing, as functionality could be isolated, improved, or replaced without affecting the rest of the system.

From the end-user perspective, the use of a map-based interface (powered by MapTiler) allowed for intuitive interaction. Users can locate nearby parking areas, visualize their boundaries, and select available spaces with minimal effort. The experience closely mimics popular mapping applications, reducing the learning curve for first-time users.

Some challenges were encountered during the implementation phase. The configuration of Firebase security rules required careful planning to ensure role-based access without compromising performance. Additionally, while Stripe integration was straightforward due to extensive documentation, deployment required attention to security keys and API configuration both on the client and Firebase backend.

Despite these challenges, the final solution demonstrated robustness and extensibility. The application's responsiveness and compatibility across devices (desktop and mobile) make it suitable for real-world deployment. Moreover, its reliance on cloud infrastructure ensures that the system can scale without the need for complex server-side management.

In summary, the application shows promise not only as a school project but also as a practical prototype that could be developed further into a production-grade platform. Real-world deployment would require enhancements such as localization, accessibility improvements, parking analytics, and integration with municipal databases or IoT devices.

Conclusion

This paper presents the design and implementation of a web-based parking management system that addresses key shortcomings identified in both international and Slovak parking applications. By integrating modern web technologies such as Angular, Firebase, Firestore, and Stripe, the system supports multi-role access, real-time synchronization, and secure online transactions within a cohesive user experience.

The proposed application provides an intuitive interface and role-specific functionalities that collectively enhance the efficiency of parking operations. Customers are able to register vehicles, view real-time parking availability, and complete reservations using cashless payments. Parking lot owners can register and manage their properties, while administrators maintain oversight through approval workflows and visibility controls.

The main contribution lies in the creation of a modular, scalable, and cloud-native system architecture. Unlike many existing Slovak applications, this solution supports persistent user accounts, map-based visualization of parking lots, dynamic data updates, and real-time status monitoring—all implemented using open and widely adopted technologies.

The practical outcomes demonstrate that it is possible to build a fully functional system with minimal backend overhead, leveraging serverless infrastructure and modern frontend development practices. The application offers a foundation for further development, including advanced features such as predictive occupancy analytics, integration with public transport APIs, and deployment as a Software-as-a-Service (SaaS) platform for municipalities and private operators.

This work also highlights the importance of usability and accessibility in urban technology systems. Future iterations could incorporate user feedback, expand to mobile-native platforms, and integrate IoT sensors for automatic status detection.

In conclusion, the system proves that real-time, role-based parking management is technically feasible and that open technologies provide a viable foundation for scalable deployment in local and regional contexts. The project bridges a functional gap in the Slovak digital services ecosystem and opens new avenues for smart urban mobility solutions.

Acknowledgement

This research was funded by the Slovak Agency for Research and Development - grant No. APVV-21-0125, by the Cultural and Educational Grant Agency of the Ministry of Education, Research, Development and Youth of the Slovak Republic KEGA 021STU-4/2024, and by the Scientific Grant Agency of the Ministry of Education, Research and Sport of the Slovak Republic No. 1/0637/23 and 1/0045/25.

References

[1] Parclick. Online parking reservation system. Retrieved online, https://www.parclick.com

[2] ParkingTag. Mobile parking payments in Ireland. Retrieved online, https://www.parkingtag.ie

[3] Upay Slovakia. Mobilná platba za parkovanie. (*Mobile payment for parking*) Retrieved online, https://www.upay.sk

[4] BPS - Bratislavský parkovací systém. Parkovanie v Bratislave. (*Parking system in Bratislava. Parking in Bratislava*) Retrieved online, https://paas.sk

[5] Google Developers. (2023). Firebase Documentation. Retrieved online, https://firebase.google.com/docs

[6] M. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash: "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications". IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015.

[7] F. T. Sheldon, K. M. Weber and S. C. Conrad, "A Software System Engineering Framework for Secure Smart Parking". in: Proc. IEEE Int. Conf. SERA, Las Vegas, USA, 2014, pp. 210–217.

[8] L. Liu, R. Jain, and Y. Wang: "Smart Parking Cloud Services and a System for Location-Based Parking Reservation". in: Proc. IEEE World Congress on Services, New York, USA, 2013, pp. 124–129.

[9] M. Collotta and G. Pau: "A novel energy management approach for smart parking applications in IoT environments". IEEE Internet of Things Journal, vol. 3, no. 6, pp. 806–813, Dec. 2016.

[10] A. Mohan, R. Ramesh and V. Ranganathan: "Design and Implementation of an Intelligent Parking System Using IoT". in: International Journal of Engineering Research & Technology, vol. 6, no. 6, pp. 340–344, 2017.

[11] M. Amine and B. Souhail: "A Smart Parking System Based on IoT and Cloud Computing". in: Proceedings of the 2020 6th International Conference on Control, Decision and Information Technologies (CoDIT), Prague, Czech Republic, 2020, pp. 1040–1045.

[12] R. Goyal and A. Agrawal: "Implementation of Smart Parking System Using Mobile Application". in: International Journal of Computer Applications, vol. 179, no. 27, pp. 22–25, 2018.

[13] C. Pham, T. D. Nguyen and M. Alazab: "A Secure and Efficient Cloud-Based Parking System for Smart Cities". IEEE Transactions on Sustainable Computing, vol. 6, no. 4, pp. 614–623, 2021.

[14] M. D. Bui, T. P. Hong and H. T. Nguyen: "An Intelligent Parking System Integrating IoT and Microservices Architecture". in: Journal of Ambient Intelligence and Humanized Computing, Springer, vol. 13, pp. 823–834, 2022.

Authors

Ing. Filip Žemla, PhD.
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava, Slovakia
filip.zemla@stuba.sk
Researcher and full-stack developer. He received the diploma and PhD. degree in Automatic Control from the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology (FEI STU) in 2023. His current work focuses on the virtualization and optimization of modern manufacturing processes, with a strong background in SCADA systems, database technologies, back-end development and front-end development. In addition to his academic background, he actively works as a full-stack developer, combining industrial automation with advanced software engineering skills.

Ing. Ján Cigánek, PhD.
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology in Bratislava,Slovakia
jan.ciganek@stuba.sk
He was born in 1981 in Malacky, Slovakia. He received the diploma and PhD. degree in Automatic Control from the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology (FEI STU) in Bratislava, in 2005 and 2010, respectively. He is now Assistant Professor at Institute of Automotive Mechatronics FEI STU in Bratislava. His research interests include optimization, robust control design, computational tools, SCADA systems, big data, and hybrid systems.