Design and Implementation of a Remote Computer Monitoring System

David Kučera, Matúš Suský, Michal Kvet
University of Žilina
Žilina, Slovakia
Michal.Kvet@fri.uniza.sk

***Abstract*—This paper presents the design and implementation of an information system for the real-time monitoring and management of remote student computers. With the increasing digitalization of education, there is a growing need for effective tools that support supervision and help ensure academic integrity during online examinations. The proposed solution is based on a client-server architecture implemented using the .NET platform. The client component provides continuous communication with the server and executes remote monitoring and management commands, while the server component offers teachers a web-based interface for centralized control and oversight. The resulting system represents a modern, user-friendly, and robust alternative to existing similar solutions, with a focus on scalability, reliability, and ease of use in educational environments.**

I. Introduction

The integration of computer technology into the educational process brings numerous advantages; however, it also introduces new challenges for educators. During lessons conducted in computer classrooms, it is often difficult for teachers to maintain students' attention and ensure that they remain focused on assigned tasks rather than engaging in unrelated activities such as browsing social networks or playing online games. This issue becomes even more critical during electronic assessments, including quizzes and examinations, where it is essential to prevent cheating and unauthorized access to external sources of information.

Existing solutions, such as the open-source Veyon system [1], provide some monitoring and control functionality, but they frequently suffer from limitations including outdated user interfaces, restricted multi-platform access for teachers, or complex configuration and deployment processes. Moreover, many of these systems lack effective mechanisms for preventing or deterring academic dishonesty during examinations.

The objective of this work was to design and implement a comprehensive information system that addresses these shortcomings. The proposed solution provides robust real-time remote computer monitoring and introduces advanced mechanisms for student assessment through an integrated *Exam Mode*. The system is built using a modern .NET 10 technology stack [2], with an emphasis on usability, extensibility, and suitability for contemporary educational environments.

II. System Analysis and Design

The system was designed as a distributed application based on client-server architecture. The primary design goals were security, stability, and low communication latency between client and server. This chapter will reveal key functions of each component of the system and technologies used.

A. Solution Architecture

The system consists of three main logical components:

1) Client component (Student PC): An application running on the Windows operating system that receives messages from the server and executes predefined commands accordingly. This component is composed of two parts: a Windows Service application responsible for background processing and continuous communication, and a WinForms application visible in a system tray interface.

2) Server component: A centralized component for managing client connections and providing a web user interface for teachers. This part of the system is implemented using Blazor Server technology and enables centralized monitoring, configuration, and control of connected student computers, which information is stored in a database.

3) Communication interface: A proprietary communication layer implementing bidirectional data transfer using WebSocket technology. This layer includes client-side, server-side and shared components, such as message definitions, serialization mechanisms, and encryption algorithms.

The solution diagram can be seen in Fig. 1. The client application (student PC) communicates with a server containing a database. Then a web application running on a teacher's device communicates with the server to perform actions on client PCs.

Fig. 1. Diagram of solution

B. Used Technologies

The C# .NET platform (version 10, LTS – Long-Term Support) was selected for the development of all system components due to its strong type safety, performance characteristics, and mature ecosystem.

1) Client: The client application is implemented as a combination of Windows Service Worker [3] for continuous background communication and a WinForms application [4] for visibility in the system tray [5] and execution of UI and user related tasks.

2) Server: The server application is implemented as a Blazor Web Application [6], enabling server-side rendering and an interactive user interface for teachers. This application is accessible through any modern web browser on both desktop and mobile devices.

3) API: The server exposes an application programming interface (API) [8] used for registering new client computers into the system during installation and unregistering them upon uninstallation. Additionally, the API provides a set of predefined operations for external applications, such as starting and stopping monitoring through the *Exam Mode*. These operations can be utilized by an external system *EXAM*, which is used at the faculty for student assessments in the courses *Database Systems* and *Advanced Database Systems*.

4) Database: An SQLite database [9] is used for storing data on the server for its simplicity and practicality. It stores data related to system users (teachers), rules, rule sets, client computers (students). Entity Framework [10] is employed as the object-relational mapping (ORM) tool. The database includes indexes to optimize data retrieval.

5) Messages: All data are transferred in a binary format over TCP/IP using WebSocket and HTTP/S protocols [7]. Each message contains information about the sender, the receiver, serialized binary data, and a creation timestamp. To serialize any type of data into binary form a NuGet packaged MessagePack is used [11]. To ensure data protection, all messages are encrypted using the AES algorithm. Upon receipt, messages are decrypted and deserialized.

6) Records and Data Storage: Client configurations, logs, screenshots and recordings are stored locally on client computers by default. Teachers may download these data from the client when necessary. Recordings are retained for a maximum of 14 days, while logs are stored for up to 30 days. After these retention periods expire, all data are permanently deleted from the client device.

III. Implementation

This chapter describes the key implementation aspects of the individual system components in more detail.

A. Client Component

The client component is divided into two cooperating modules.

The first module is a Windows Service Worker running under the *LocalSystem* account. This ensures that the service is active immediately after the system startup, regardless of whether a user is logged in, and that it has sufficient privileges to perform administrative operations. This module is responsible for executing system-level commands such as computer restart, shutdown, user logout, website blocking, and rule enforcement. The service starts automatically during system boot and does not require any manual intervention. All service logs can be viewed in the Event Viewer application (a Windows tool).

The second module is a WinForms system tray application, which is started after a user logs into the computer. It serves as a visual indicator and acts as an intermediary for user interface-related tasks, including displaying notifications and capturing screen content. The correct operation of the client component is indicated by the presence of a system tray icon. This application is automatically launched by the Windows Service when any user logs in. The application icon can be seen in Fig. 2.

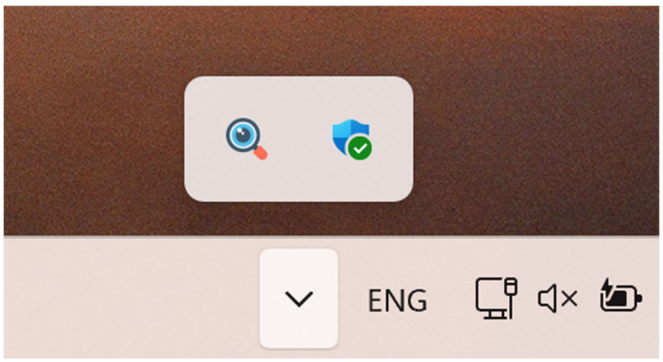

Fig. 2. A view at tray icon (NotifyIcon component) in Windows system tray

The two client modules communicate with each other using the same communication solution employed between server and client. In this context, the Windows Service acts as the primary client. If the service is unable to execute a command directly, it forwards the message to the tray application for execution. Since the tray application does not maintain a direct connection to the server, all outbound messages are first sent to the Windows Service, which then relays them to the server.

Protection against unauthorized manipulation by students is achieved by installing executable files in the *Program Files* directory on the system drive and storing other files such as configuration files, logs and records in *ProgramData* directory. As students do not possess administrator privileges on school computers, they are unable to terminate the service or modify application files. Installation is performed using a dedicated installer application that deploys both Windows Service and the WinForms tray application. The Windows Service is registered using the Sc command [12] during this process.

B. Server Component

The server component functions as a centralized control center for teachers. The user interface is designed with an emphasis on simplicity and usability. After successful authentication, users can navigate through several functional pages:

1) Dashboard: The primary monitoring interface, allowing users to select a specific group of clients which they want to look at and perform monitoring and control actions. The view

displays only client thumbnails, with their size adjustable using a slider located in the page footer. A detailed view with additional client information can be opened when required. Dashboard page can be seen in Fig. 3.

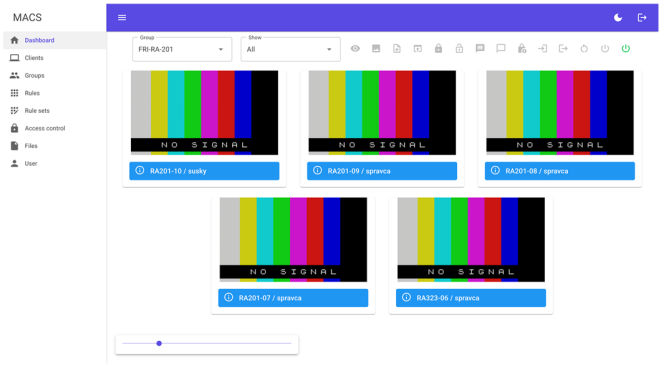

Fig. 3. Dashboard in light mode

All application pages can be also viewed in dark mode. You can see the same dashboard page from Fig. 2 in dark mode in Fig. 4.

Fig. 4. Dashboard in dark mode

The detailed view (see Fig. 5) provides an enlarged live screen preview of the selected client and allows the user to switch between available monitors if the client device is equipped with multiple displays. Additional features include access to the client's webcam stream (if available), visibility of currently running programs, and real-time keystroke monitoring. The user can remotely terminate any running program and, if necessary, take remote control of mouse and keyboard input.

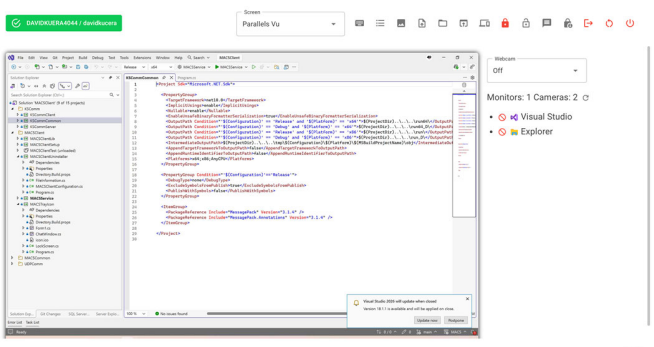

Fig. 5. Detailed view of a connected client

Furthermore, the interface enables browsing files on the client's local storage drives, allowing users to download or delete files when necessary. This dialog is visible in Fig. 6.

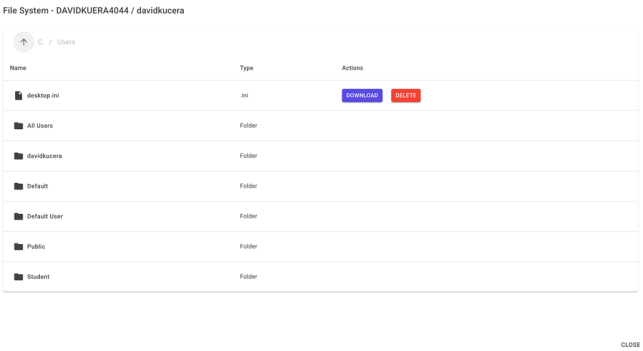

Fig. 6. Browse client files dialog window

2) Clients: A page displaying a list of all registered client computers in the system, including information such as unique identifier, computer name, IP address, MAC address and assigned group. This page is visible in Fig. 7.

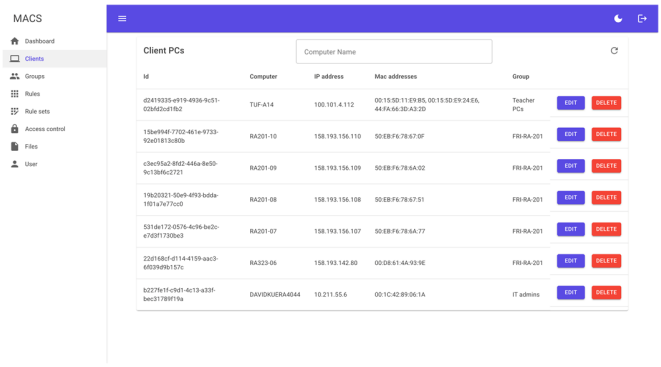

Fig. 7. Clients' page

3) Groups: An interface for creating and managing groups that represent a real-world classrooms. Each group can be assigned a unique name (e.g., RA-201, RA-013) and may contain any number of client computers, including none. See Fig. 8.

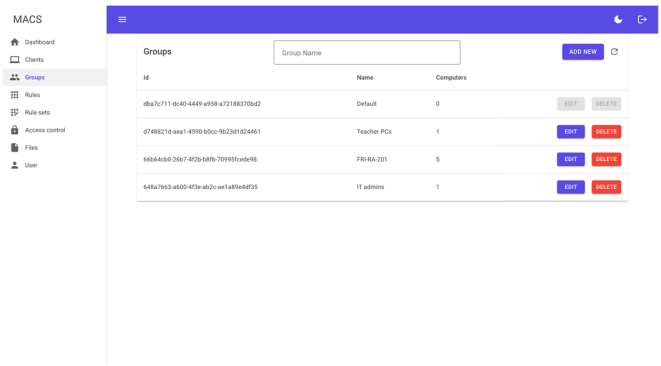

Fig. 8. Groups page

4) Rules: A section listing all user-defined rules, such as blocking specific browsers, applications or websites.

5) Rule sets: Rule sets are used to define configurations for different *Exam Modes*. Individual rules are first created in the

Rules section and then combined into rule sets. When an exam session is initiated, the teacher selects the appropriate rule set to be applied to the monitored clients. Let's say a teacher wants to block a browser, an application and a website together during exam taking. He will need to create individual rules in *Rules* page and then assign them to a rule set created in *Rule sets* page. Then when starting the *Exam Mode*, he can select which rule set to use for that monitoring.

6) Access control: An administrative page for managing API keys and system users. API keys can be created, modified, or revoked. User accounts can be created, edited, have their password reset, have two-factor authentication (2FA) removed, or be deleted.

7) Files: This page provides access to all stored files created during *Exam Mode*, such as screenshots containing prohibited activities and reports generated after the termination of an exam session. Reports are produced in PDF format and contain a chronological list of detected violations made by the student during session with corresponding timestamps, facilitating post-exam review.

8) User Settings: A personal settings page allowing teachers to configure their first and last name, username, and changing a password. It also provides an option to enable two-factor authentication (2FA) using time-based one-time passwords (TOTP), adding an additional layer of security to the authentication process. When enabled, user will need to provide a code from an authentication app for login.

C. Communication Interface

A proprietary communication interface based on WebSocket technology was implemented to meet the specific requirements of the system. This approach was chosen to enable efficient transmission of diverse data types, ranging from simple textual commands to image data and complete files of various formats. The protocol operates on raw byte arrays, allowing arbitrary data to be transmitted.

```
using MessagePack;

namespace KSCommCommon
{
    [MessagePackObject]
    99+ references
    public class Message
    {
        Constants

        #region Properties
        [Key(0)]
        99+ references
        public string Source { get; set; } = string.Empty;
        [Key(1)]
        99+ references
        public string Destination { get; set; } = string.Empty;
        [Key(2)]
        99+ references
        public string ResponseToken { get; set; } = string.Empty;
        [Key(3)]
        99+ references
        public MessageAction Action { get; set; } = MessageAction.None;
        [Key(4)]
        99+ references
        public MessageType Type { get; set; } = MessageType.None;
        [Key(5)]
        99+ references
        public byte[] Data { get; set; } = [];
        #endregion //Properties

        Constructor

        Public functions
    }
}
```

Fig. 9. Definition of Message class

Each message (see class definition in Figure 9) includes a defined type (e.g., *StartRecording*, *CreateScreenshot*, *File*, *Rule*), payload data, sender information (Source), target recipient (Destination), and an optional response token used for messages that expect instant reply. A built-in serializer converts any type of object into a byte array and back. Prior to transmission, all data are encrypted using the AES algorithm to ensure confidentiality. The communication layer supports operation over both HTTP and HTTPS protocols, ensuring secure and reliable real-time communication between server and clients.

Every client connects to the server via *ClientWebSocket* object [13] and maintains a heartbeat to detect disconnections. Upon detecting a disconnection, the client automatically attempts to reconnect until the connection is re-established. Events are triggered for incoming messages, replies, disconnections, and timeouts. The interface supports standard actions such as connect, disconnect, heartbeat, as well as message-related functions such as send message, send message with response.

D. Database

The system uses a class built on EntityFramework Core to interact with SQLite database. Each entity – clients, users, groups, rules, rule groups, and API keys – is represented as a *DbSet<TEntity> [14]*, with relationships and cascading behaviors defined to ensure data consistency. The class provides methods for creating, reading, updating, and deleting records, as well as specialized operations such as assigning rules to groups, importing/exporting rules, managing TOTP secrets, and handling API keys. The database consists of six tables (see Figure 10): *ApiKeys*, *Clients*, *Groups*, *RuleGroups*, *Rules*, and *Users*. To improve query performance for frequently accessed data, three indexes are defined: *Clients_Group*, *RuleGroups_UserId* and *Rules_UserId*, which significantly enhance retrieval speed compared to non-indexed queries.

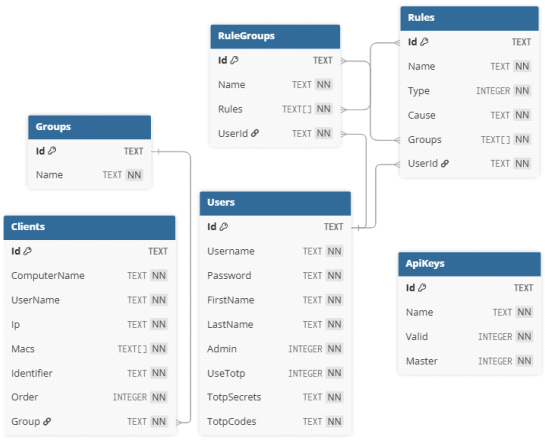

Fig. 10. Data model

E. API

The system provides a RESTful API implemented using ASP.NET Core, which exposes endpoints for managing clients, users, and API keys, as well as controlling monitoring and recording features. The API is structured around a controller, which handles all requests under the */api/* route and returns responses in JSON format.

- User Management: *POST /user/add* and *GET /user/{id}* for creation and retrieval of user accounts.
- Client Management: *POST /client/register*, *DELETE /client/unregister* and *GET /client/{id}* for registering, unregistering and retrieving clients' details.
- Recording Control: *GET /recording/start/{gId}* and *GET /recording/stop/{gId}* for starting and stopping client computer screen and webcam recording.
- API Key Management: *POST /keys/generate*, *GET /keys/{id}* and *POST /keys/invalidate/{kId}* for managing authentication keys – generating, retrieving and invalidating.

All endpoints include logging, input validation, and appropriate HTTP response codes to facilitate error handling and ensure reliable operation.

IV. System Functionality

A. Real-Time Monitoring

The core functionality of the system is real-time visualization of student computer activity.

- Dashboard View: Previews of all student computers within a selected group are displayed and updated at a frequency of one frame per second (1 FPS). These preview windows can be rearranged within the interface as needed. It is possible to show only client computers that are in use, hiding the turned off ones.
- Detailed View: When a specific online client computer is selected, the system switches to a higher-priority monitoring mode. In this mode, the streamed image is presented at higher resolution and quality, and additional control actions become available, such as mouse and keyboard control. The system supports multi-monitor environments and webcams, allowing the user to switch between available image sources. In addition to screen streaming, real-time visualization of keystrokes is also supported, as well as browsing client files.

B. Remote Management and Intervention

The system provides teachers with a set of tools for active interaction and intervention:

- File Transfer: Files can be sent directly to the student's Desktop or Download directory.
- Application Launch: Applications can be started remotely by specifying either the program name or the path to executable file.
- Open Website: A specified website can be opened remotely in a predefined web browser.
- Power Management: Supported operations include Wake-on-LAN (WOL) for remotely starting computers connected via an Ethernet cable, restarting the computer, logging off the current user, locking the workstation, and shutting down the system.

C. Exam Mode

The most significant innovation is the module used when writing tests or exams. Prior to starting an exam session, the teacher can define a set of rules and group them into a rule set, which can be then applied to selected group at the beginning of the exam.

If a student launches a prohibited application during exam, the following actions are performed automatically:

1) The teacher receives a notification, and the application is immediately terminated on the student's computer (optional).
2) The client automatically captures a screenshot of the incident and sends it to the server, where it is stored and made available for review. A local copy is also saved on the client's computer.
3) The incident is logged onto the server. Upon completion of the exam, a PDF report containing these logs is generated.

If a student attempts to access a prohibited website, the request is blocked and the website does not load. In this case, no event is reported to the server.

During *Exam Mode*, it is possible to also enable screen and camera recording during the whole session. These recordings are stored locally on the client computer for a limited retention period of several weeks.

After the exam is terminated, the server generates a PDF report (see Figure 11) containing a chronological timeline of all detected unauthorized activities, including precise timestamps for each client.

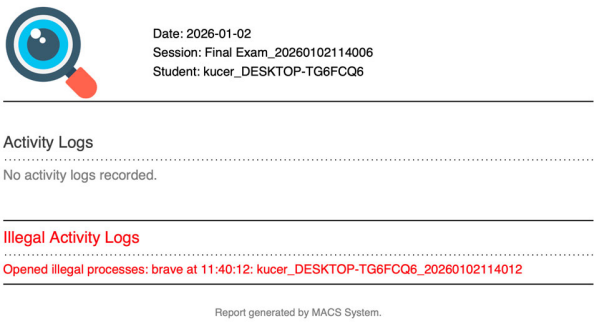

Fig. 11. Generated PDF report

The evidence (screenshot) of these detected unauthorized activities is also saved on the server and client computer.

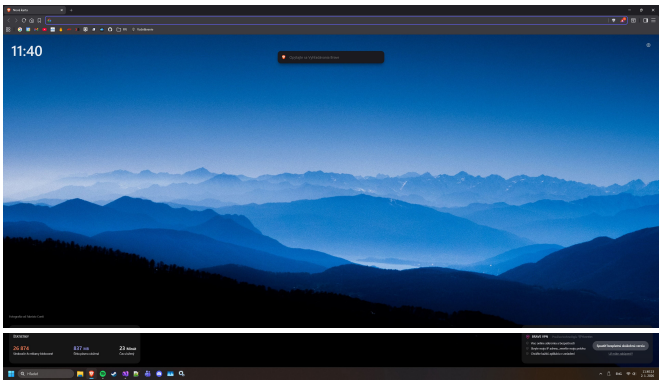

Fig. 12. Evidence of brave process running at recorded time in report

V. Discussion and Comparison

Compared to the Veyon system, the proposed solution provides several significant improvements.

The primary advantage is the web-based nature of the server component, which allows teachers to monitor classrooms without being tied to a specific "master" computer. Monitoring can be performed using their mobile devices, desktops or laptops, even remotely from the comfort of their home, providing greater flexibility and accessibility.

Another notable improvement is the granularity of settings in Exam Mode and the automated generation of reports, which substantially reduces the time teachers need to evaluate exam integrity. Additionally, the proprietary communication interface based on WebSocket technology has demonstrated effective performance for low-latency image transmission within a local network, ensuring smooth real-time monitoring and interaction with client computers.

Overall, the system offers a modern, flexible, and efficient alternative to existing solutions, combining enhanced usability, real-time monitoring, and robust exam supervision features.

Table I. presents a comparison between the proposed remote computer monitoring system with the open-source Veyon system [1].

VI. Conclusion

The objective of this paper was to present the design and implementation of a system developed at our faculty over the past year. The project aimed to create a modern and efficient tool for classroom monitoring. The resulting system successfully integrates real-time monitoring, remote management, and Exam Mode functionality into a single, user-friendly platform.

Operational testing has demonstrated the stability of the client components, as well as sufficient network performance for transmitting screen previews in real time.

Future work includes expanding the system to support additional client platforms, such as Linux and macOS, thereby increasing its applicability and flexibility in diverse educational environments.

Acknowledgment

This paper was supported by the **VEGA 1/0192/24** project - Developing and applying advanced techniques for efficient processing of large-scale data in the intelligent transport systems environment.

TABLE I. Comparison to Veyon system

Feature/Aspect	Proposed System	Veyon
Server Access	Web-based server; accessible from any device (desktop, mobile, remote)	Desktop-based master required; limited remote access
User Interface	Modern Blazor web interface; dark/light mode; mobile-friendly	Classic desktop UI; tied to master PC; limited themes
Exam Mode	Advanced rule sets; automatic application and website blocking; screenshots and logging; PDF report generation	Limited blocking; no automated reporting
API	RESTful API for client/user management and integration with external systems	Limited API
Security	Password protection; Two-factor authentication (TOTP) optional, AES encryption	Password protection; encryption optional
Platform Support (Client)	Windows	Windows, Linux

References

[1] Veyon official website, Cross-platform computer control and classroom management, Web: https://veyon.io/.
[2] Microsoft Learn, What's new in .NET 10, Web: https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-10/overview/.
[3] Microsoft Learn, Worker Services in .NET, Web: https://learn.microsoft.com/en-us/dotnet/core/extensions/workers.
[4] Microsoft Learn, Windows Forms Overview, Web: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/.
[5] Microsoft Learn, How to: Add Application Icons to the TaskBar with the Windows Forms NotifyIcon Component, Web: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/controls/app-icons-to-the-taskbar-with-wf-notifyicon.
[6] .NET official website, Blazor | Build client web apps with C#, Web: https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor.
[7] Microsoft Learn, WebSockets support in .NET, Web: https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/websockets.
[8] .NET official website, APIs with ASP.NET Core — Build secure REST APIs on any platform with C#, Web: https://dotnet.microsoft.com/en-us/apps/aspnet/apis.
[9] Microsoft Learn, Microsoft.Data.Sqlite overview, Web: https://learn.microsoft.com/en-us/dotnet/standard/data/sqlite/.
[10] Microsoft Learn, Entity Framework documentation hub, Web: https://learn.microsoft.com/en-us/ef/.
[11] NuGet Gallery, MessagePack NuGet package, Web: https://www.nuget.org/packages/messagepack.
[12] Microsoft Learn, Sc, Web: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc754599(v=ws.11).
[13] Microsoft Learn, ClientWebSocket Class (System.Net.WebSockets), Web: https://learn.microsoft.com/en-us/dotnet/api/system.net.websockets.clientwebsocket?view=net-10.0.
[14] Microsoft Learn, DbSet<TEntity> Class (System.Data.Entity), Web: https://learn.microsoft.com/en-us/dotnet/api/system.data.entity.dbset-1?view=entity-framework-6.2.