BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2025
http://excel.fit.vutbr.cz

Web Application for Smart Device Management and Responsive IoT Data Visualization

Marek Joukl, Marko Olešák

Abstract

The rise of IoT devices and smart homes has created a need for their management, monitoring, and analysis. This paper presents a newly developed frontend application built on the existing Real-Time IoT (RIoT) system that fulfills these requirements. The new solution replaces the original frontend application, which suffered from poor responsiveness, several user experience issues, and a complete lack of dashboard or data visualizations, with a fully modular and mobile-friendly web interface. The application is capable of dynamic device type definition, customizable dashboard with real-time data visualization, Key Performance Indicators (KPIs) management through an integrated editor, process automation using a Visual Programming Language (VPL) editor, and newly introduced support for device commands.

*xjoukl00@vutbr.cz, xolesa00@vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

The rapid development of smart home technologies and IoT ecosystems has created a strong demand for unified management and monitoring platforms. However, many existing solutions remain tied to specific brands, lack flexibility, or struggle with poor responsiveness and usability. Addressing these challenges, this project focuses on the redesign and implementation of a new frontend application built on the RIOT system [1], a platform offering a Go-based GraphQL backend for real-time device management.

The original RIoT frontend exhibited several limitations, including a non-responsive design, inconsistent user experience, and missing features crucial for effective device configuration and system scalability. The newly developed frontend application introduces a fully modular, mobile-first architecture based on modern web development principles. It supports dynamic device type definition, enabling users to flexibly define new devices with custom parameters and commands. Furthermore, the system incorporates a customizable dashboard with real-time data visualization, a Key Performance Indicator (KPI) editor for monitoring performance metrics, and a Visual Programming Language (VPL) editor to automate smart device behavior. Moreover, devices can be organized into groups for enhanced control.

2. Current Solutions

There are many other existing solutions, notable examples include Home Assistant[1], OpenHAB[2], or Google Home[3]. However, they often fall short in terms of general usability, customization, scalability, or responsiveness. For instance, OpenHAB does not provide a user-friendly interface and can be truly challenging for beginners to navigate or set up. In terms of responsiveness, Home Assistant largely mirrors the desktop application, and Google Home does not support any form of historical data storage or preview.

The implemented application thus seeks to bridge this market gap by creating a solution that is both user-friendly and has the ability to cater to even more advanced users, with its powerful data visualizations, KPIs, device, and group management.

3. System Architecture

The developed frontend application is built using React combined with TypeScript. Communication with the backend is handled entirely through Apollo Client, which manages queries, mutations, and subscriptions

[1] Home Assistant homepage - https://www.home-assistant.io/

[2] OpenHAB homepage - https://www.openhab.org/

[3] Google Home homepage - https://home.google.com/welcome/

via a GraphQL API. This approach abstracts away direct network communication and provides robust tools for state management, error handling, and caching.

The backend system is based on the RIOT platform, utilizing a Go-based GraphQL server connected to PostgreSQL and InfluxDB databases. All device configurations, including dynamically defined device types and their associated parameters or commands, are stored and retrieved through this API.

The application also incorporates internationalization support through the i18next library, allowing users to dynamically switch between English and Czech languages. In addition, the interface supports both dark and light modes, enabling users to customize the application's appearance.

4. Key Features

The application as a whole was split into two main parts: data visualization and device management. Starting with the dashboard shown in Screen 1, it is fully responsive and features an independent layout for each defined breakpoint, currently four, while following the principles of the book [2]. The layout features drag-and-drop resizing and moving of individual cards, with dedicated buttons for better accessibility on mobile devices. The dashboard unlocks for changes upon entering edit mode and can be reverted while inside. Furthermore, the dashboard is split into multiple sections or categories for better organization using tabs, where the same applies, and each tab is entirely independent of the rest and features layouts for each separate breakpoint.

Visualizations, shown in Figure 2, can be added using the '+' button and further adjusted at any time in the edit mode. Their color schemes were carefully selected to be colorblind friendly, following the book's [3] guidelines.

Since dashboards cannot display all information at once, detailed views of devices and groups are closely integrated with the dashboard and can be essential in complex smart homes. These views are then suitable for drill-down operations or similar tasks.

The device management and configuration part of the application is demonstrated in Screen 3. This screen shows the overview page for device types, allowing users to view, search, and manage existing device definitions. Each device type is presented in a card layout, showing the MQTT denotation and offering quick actions such as viewing associated instances or deleting the type. Importantly, the new solution introduces the ability to edit device types individually, improving on the previous system where only creation and deletion were possible. The process of dynamically defining new device types and integrating them into the system is visualized in Figure 1. The figure illustrates how a user can configure a new device type through the frontend, which sends the configuration in JSON format via GraphQL to the backend.

The application also offers the ability to define KPIs through an integrated editor, allowing users to monitor important metrics related to their devices. This KPI editor was adapted and integrated from the original system, with minor adjustments to align with the new architecture and styling. In addition, a VPL editor is included, enabling users to create automation programs that control device behavior based on defined logic.

5. Challenges and Limitations

During the development of the new frontend application, several challenges and limitations were encountered. One significant limitation is related to backend readiness. Some features, such as the members management functionality, are not fully operational because the backend API does not yet support all necessary operations for managing user groups. While the frontend components for listing and viewing members have been prepared, they currently rely on mock data and will require further integration once the backend side is completed. Similarly, command actions and their backend handling are still in the development process. Nevertheless, the frontend components are prepared for their integration.

References

[1] Michal Bureš. Systém pro zpracování dat z chytrých zařízení. Bachelor's thesis, Brno University of Technology, Faculty of Information Technology, Brno, 2024.

[2] Stephen Few. *Information Dashboard Design: The Effective Visual Communication of Data*. O'Reilly Media, Inc., 2006.

[3] Claus O. Wilke. *Fundamentals of Data Visualization*. O'Reilly Media, Inc., Sebastopol, CA, 2019, 2019.