VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

GRAFICKÉ ROZHRANÍ PRO NASTAVOVÁNÍ A SPRÁVU INTELIGENTNÍCH DOPRAVNÍCH SYSTÉMŮ

GRAPHICAL INTERFACE FOR CONFIGURING AND MANAGING INTELLIGENT TRANSPORTATION SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **VOJTĚCH KUCHAŘ**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Dr. Ing. OTTO FUČÍK**
SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

164481

Ústav:	Ústav počítačových systémů (UPSY)
Student:	**Kuchař Vojtěch**
Program:	Informační technologie
Název:	**Grafické rozhraní pro nastavování a správu inteligentních dopravních systémů**
Kategorie:	Webové aplikace
Akademický rok:	2024/25

Zadání:

1. Prostudujte problematiku grafických rozhraní pro nastavování a správu inteligentních dopravních systémů (ITS) s důrazem na moderní webové technologie, ergonomii a podporu jejich vývoje.
2. Předpokládejte, že GUI bude použito u produktů, které běží na různých platformách (ARM, Intel, mikrokontrolery).
3. Předpokládejte použití vhodných frameworků, které mají kvalitní podporu vývoje a je u nich předpoklad, že budou mít dlouhou životnost.
4. S využitím vybraného frameworku navrhněte a realizujte knihovnu komponent specifických pro ITS produkty.
5. S využitím knihovny komponent navrhněte a implementujte aplikaci pro vybraný produkt, která bude realizovat GUI pro jejich nastavování, sběr dat a monitorovací funkce (např. online obraz z kamery).
6. Implementovanou aplikaci otestujte na reálném produktu.
7. Vyhodnoťte dosažené výsledky.

Literatura:
Dle doporučení vedoucího práce.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Fučík Otto, doc. Dr. Ing.**
Konzultant:	Ing. Filip Kadlček, PhD
Vedoucí ústavu:	Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání:	1.11.2024
Termín pro odevzdání:	14.5.2025
Datum schválení:	31.10.2024

Abstrakt

Práce se zaměřuje na tvorbu grafického uživatelského rozhraní s důrazem na ergonomii a uživatelskou přívětivost. Hlavním cílem bylo analyzovat stávající webová GUI a identifikovat klíčové výzvy a možnosti pro zlepšení. Na základě této analýzy byly stanoveny požadavky, podle kterých byla navržena a implementována knihovna GUI komponent specifických pro nastavování a správu produktů inteligentních dopravních systémů. Knihovna byla následně demonstrována na aplikaci implementované pro vybraný ITS produkt, která zohledňuje moderní webové technologie a přibližuje se optimálnímu uživatelskému zážitku.

Abstract

The work focuses on the creation of a graphical user interface with an emphasis on ergonomics and user-friendliness. The main objective was to analyse existing web GUIs and identify key challenges and opportunities for improvement. Based on this analysis, the requirements were determined, according to which a library of GUI components specific to the setup and management of Intelligent Transportation Systems products was designed and implemented. The library was then demonstrated on an application implemented for the selected ITS product, taking into account modern web technologies and approaching an optimal user experience.

Klíčová slova

grafické uživatelské rozhraní, webové technologie, ergonomie, analýza gui, produktové weby, knihovna komponent, blazor, inteligentní dopravní systémy

Keywords

graphical user interface, web technologies, ergonomics, gui analysis, product websites, component library, blazor, intelligent transportation systems

Citace

KUCHAŘ, Vojtěch. *Grafické rozhraní pro nastavování a správu inteligentních dopravních systémů.* Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Dr. Ing. Otto Fučík

Grafické rozhraní pro nastavování a správu inteligentních dopravních systémů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Dr. Ing. Otta Fučíka. Další informace mi poskytl pan Ing. Filip Kadlček, PhD. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Vojtěch Kuchař
13. května 2025

Poděkování

Chtěl bych poděkovat vedoucímu své bakalářské práce, panu doc. Otto Fučíkovi, za vedení a hodnotné rady při konzultacích. Dále pak panu Ing. Filipu Kadlčkovi, PhD. a společnocti CAMEA za zpřístupění kamery, poskytnutí potřebných dat a odborné konzultace.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V současném rychle se rozvíjejícím technologickém světě hrají inteligentní dopravní systémy (ITS) zásadní roli v zlepšování dopravní bezpečnosti, efektivity a pohodlí. Tato bakalářská práce se zaměřuje na studium a návrh grafického uživatelského rozhraní pro ITS produkty, jako jsou chytré kamery, dopravní senzory a zařízení internetu věcí. S důrazem na moderní webové technologie, ergonomii a podporu vývoje se práce věnuje výzvám spojeným s tvorbou uživatelsky přívětivých a efektivních rozhraní pro různorodé platformy. V rámci práce byla provedena analýza současného stavu webových GUI pro ITS produkty. Na základě této analýzy a analýzy konkurenčních produktů a jejich webů byly stanoveny požadavky na nové webové GUI, podle kterých byla navržena a implementována knihovna komponent. Knihovna byla následně demonstrována na aplikaci pro manipulaci a sběr dat dopravní kamery UC-SCA.

Kapitola 2

Analýza současného stavu

Tato kapitola se zaměřuje na rozbor aktuální situace v oblasti inteligentních dopravních systémů a zkoumá jejich software, dostupné technologie a konkurenční řešení. Cílem kapitoly je nastínit současný stav uvedené problematiky a přiblížit čtenáři technologie, relevantní pro následný návrh a realizaci knihovny komponent pro nastavování a správu produktů inteligentních dopravních systémů. Na základě analýzy bude sestaven soubor požadavků na výsledné grafické uživatelské rozhraní.

2.1 Inteligentní dopravní systémy

ITS, neboli inteligentní dopravní systémy, představují integraci pokročilých informačních a komunikačních technologií do dopravních systémů a infrastruktury. Do této kategorie spadají například senzory a řídicí technologie, včetně internetu, a jsou aplikovány s cílem zvýšit bezpečnost, udržitelnost, efektivitu a pohodlí v dopravě. ITS jsou blíže popsány standardem ISO/TR 17465-1:2014 [3].

Přehled ITS produktů

Inteligentní dopravní systémy zahrnují širokou škálu produktů a technologií zaměřených na zlepšení vlastností dopravních systémů. Důležitou součástí těchto systémů jsou grafická uživatelská rozhraní (GUI), která umožňují uživatelům interagovat s ITS zařízeními a softwarem. Tato sekce poskytuje přehled hlavních ITS produktů, jejich výstupů a charakteristiku jejich GUI.

Chytré kamery

Jedním ze základních prvků inteligentních dopravních systémů jsou chytré kamery. Nejsou určeny pouze pro pouhý záznam obrazu, ale také využívají pokročilé algoritmy zpracování obrazu, které umožňují automatickou optimalizaci výstupu podle aktuálních podmínek [29]. Například díky inteligentnímu přepínání mezi denním a nočním režimem dokáží systémy dynamicky upravovat nastavení, jako jsou expozice, kontrast či ostrost, což značně zvyšuje kvalitu získaného obrazu. Nastavení těchto kamer je většinou možné vzdáleně upravovat.

GUI těchto kamer obvykle nabízí:

- **Živý přenos**: Zobrazení živého videa z kamery. V systémech pracujících s více kamerami lze zobrazovat více pohledů najednou.

- **Nastavení parametrů kamery**: Možnosti upravit nastavení jako expozice, ostření, ale pokročilejší parametry zařízení, které se mohou měnit v určitých podmínkách. Příkladem může být změna způsobu zpracování obrazu ve dne a v noci.
- **Analýza a záznam**: Funkce pro analýzu videa, ukládání a přehrávání záznamů.
- **Integrace s ostatními systémy**: Možnosti integrace s jinými ITS komponentami. Nejčastěji se jedná o systémy pro rozpoznávání SPZ a měření provozu, případně rozšíření o více kamer.

Obrázek 2.1: Chytrá kamera UC-SCA společnosti CAMEA, spol. s r.o.

Inteligentní senzory

Senzory v ITS shromažďují data o dopravních podmínkách, jako je hustota provozu a rychlost vozidel [4]. Určité druhy inteligentních senzorů jsou schopny detekovat
a analyzovat míry koncentrace škodlivin v ovzduší nebo monitorovat povětrnostní podmínky. Produktové weby pro tyto senzory obvykle umožňují:

- **Dashboardy**: Grafické zobrazení shromážděných dat. Obvykle je možné v reálném čase zobrazit statistiky a identifikovat trendové vzorce. Další možnosti zahrnují zobrazení historických dat a vytváření vlastních dashboardů.
- **Nastavení Senzorů**: Konfigurace parametrů senzorů. Může se jednat o nastavení parametrů jako je rozsah detekce, citlivost. V oblasti chytrých kamer může jít o nastavení parametrů jako je expozice, ostření a další.
- **Výstrahy a Oznámení**: Nastavení upozornění na neobvyklé události v provozu. Konkrétně se může se jednat o upozornění na přetížení silnice, neobvyklé chování řidičů a podobné neobvyklé či jinak zajímavé situace.

IoT zařízení

Internet věcí (IoT) v kontextu inteligentních dopravních systémů hraje nemalou roli, propojuje různá zařízení a umožňuje sběr a analýzu dat [8]. GUI pro různá IoT zařízení z pravidla zahrnují rozhraní pro přidávání, konfiguraci a monitorování IoT zařízení. Dále bývá možná vizualizace dat a identifikace trendů v reálném čase, případně možnost integrace s jinými ITS komponentami.

Produktové weby

Produktové weby jsou součástí ITS produktů, které poskytují uživatelům informace o produktech, jejich funkcích a v některých případech umožňují produkty nastavovat. Pro takové produktové weby je důležité, aby jejich GUI splňovalo určitá kritéria. Mezi taková kritéria může patřit například uživatelská přívětivost, efektivita práce se zdroji zařízení nebo snadnost použití. Při výběru těchto vlastností je nutné zohlednit různorodé uživatele těchto webů, včetně technicky méně zdatných uživatelů, kteří by mohli mít problémy s používáním složitých GUI. Je proto nutné najít balanc mezi funkcionalitou a jednoduchostí, aby bylo dosaženo optimálního uživatelského zážitku [34].

2.2 Technologie

V této sekci popisuji nabyté teoretické znalosti o technologiích, na které dále navazuji v dalších částech práce. Jedná se o stručný přehled webových technologií, frameworků, jejich významu v kontextu vývoje Blazor[1] aplikací a webových aplikací obecně.

2.2.1 Webové technologie

S vývojem webových aplikací nutně souvisí základní webové technologie, na které Blazor navazuje a se kterými úzce spolupracuje. HTML a CSS tvoří základní kámen struktury a vzhledu každé webové aplikace, a to i při práci v Blazoru.

HTML, CSS a responsivní design

HTML (Hypertext Markup Language) a CSS (Cascading Style Sheets) jsou technologie pro tvorbu webových stránek a aplikací.

HTML definuje obsah stránky pomocí tzv. tagů, které označují jednotlivé části dokumentu a určují jejich strukturu. Mezi základní tagy patří například `<h1>` pro nadpisy, `<p>` pro odstavce, `<a>` pro odkazy, `<ul>` a `<li>` pro seznamy, nebo `<div>` pro blokové prvky. Každý tag může mít své atributy, které specifikují jeho vlastnosti, mezi nejpoužívanější patří class, id, src, či href [23].

CSS na stránce definuje vzhled. Pro jednotlivé tagy lze pomocí CSS nastavit vlastnosti, jako jsou barvy, velikost, mezery mezi prvky, atd. Pomocí selektorů jako `.trida`, `:hover` nebo `div > p` lze cílit konkrétní části rozhraní. Flexbox a Grid umožňují pokročilé rozvržení stránky. Pro přizpůsobení vzhledu různým zařízením se v CSS používají tzv. `@media` dotazy, které umožňují měnit styly podle šířky obrazovky či typu zařízení, což umožňuje tvořit stránky s responzivním designem [22].

[1] Více o frameworku Blazor viz sekce 2.2.5.

V rámci Blazor aplikací hraje CSS důležitou roli při stylování komponent a vytváření konzistentního a responzivního uživatelského rozhraní. Přestože Blazor umožňuje programování uživatelského rozhraní v jazyce C#, samotný vzhled je stále definován prostřednictvím standardních webových technologií, mezi které CSS neodmyslitelně patří. Blazor poskytuje několik možností, jak styly aplikovat [14]:

- **Globální styly** – Definované v souborech jako `wwwroot/css/app.css` nebo v jiných externích stylech připojených ke stránce. Tyto styly ovlivňují celé uživatelské rozhraní.
- **Styly specifické pro komponenty** – Pro každou komponentu lze definovat vlastní styly v souboru s příponou `.razor.css`, například `ComponentName.razor.css`. Tyto styly jsou aplikovány pouze na danou komponentu a zamezují nechtěnému ovlivnění ostatních částí aplikace. Blazor v tomto případě využívá tzv. CSS izolaci.
- **Inline styly a třídy** – Přímo v komponentách lze pomocí atributů jako `class` nebo `style` nastavovat vzhled jednotlivých prvků. Tento přístup je vhodný pro dynamické změny stylů podle stavu komponenty.

Bootstrap

Bootstrap je populární open-source framework pro vývoj responzivních a mobilně přívětivých webových stránek. Poskytuje sadu nástrojů v HTML, CSS a JavaScriptu, které usnadňují tvorbu konzistentních a esteticky příjemných uživatelských rozhraní napříč různými zařízeními. Bootstrap byl původně vyvinut ve společnosti Twitter a poprvé vydán v roce 2011. Od té doby se stal standardem v oblasti front-end vývoje díky své jednoduchosti a rozsáhlé komunitní podpoře. Více o Bootstrapu naleznete zde [13].

Figma

Figma je cloudový nástroj, umožňující vývojářům vytvářet a testovat interaktivní prototypy webů, GUI a mobilních aplikací. Nabízí nástroje pro vytváření grafických návrhů, vektorových ilustrací a interaktivních prototypů, které mohou být snadno sdíleny a upravovány v reálném čase [1].

JavaScript a jeho využití s Blazorem

JavaScript je skriptovací jazyk, který se běžně používá k tvorbě interaktivních webových stránek. V rámci Blazor aplikací je hlavní motivací pro jeho použití možnost interakce s DOM (Document Object Model) prostřednictvím volání JavaScriptových funkcí ze C# kódu. To značně usnadňuje práci s interaktivními prvky v komponentách, jejichž správné fungování nelze zaručit bez přístupu k DOM. Příkladem takové situace může být například potřeba zjistit finální velikost vykresleného HTML prvku.

Dalším praktickým využitím JavaScriptu je možnost volat funkce JavaScriptových knihoven, jako jsou například Chart.js, RGraph, jQuery, Polymer a mnoho dalších, které mohou být pro účely práce přínosné. Provádění JavaScriptového kódu z Blazorových aplikací probíhá za pomocí knihovny `JSRuntime` z balíčku `Microsoft.JSInterop` [33].

2.2.2 Komunikace se serverem

V následující sekci jsou představeny technologie, umožňující webovým aplikacím komunikaci se serverem.

REST API

REST (Representational State Transfer) API je architektonický styl pro navrhování webových služeb, který byl popsán Royem Thomasem Fieldingem v jeho disertační práci Architectural Styles and the Design of Network-based Software Architectures [19].

REST API využívá standardní HTTP metody, kterými může klient spravovat zdroje umístěné na serveru. Zdroje jsou identifikovány pomocí jejich URI (Uniform Resource Identifiers[2]), což jsou jednoznačné adresy, které určují umístění konkrétního zdroje na serveru. Jednotlivé HTTP metody pak nad touto adresou provádějí příslušné operace. Úspěšnost operací se udává v odpovědi na požadavek v čísle Response Status Code.

HTTP požadavky mohou být několika typů:

- **HTTP POST**: Je používán pro vytvoření položky (create).
- **HTTP GET**: Je používán pro čtení položky (read).
- **HTTP PUT**: Je používán pro aktualizaci položky (update).
- **HTTP DELETE**: Je používán pro smazání položky (delete).

Tyto čtyři základní operace odpovídají tzv. CRUD modelu (Create, Read, Update, Delete), který představuje standardní přístup k manipulaci s daty v aplikacích.

REST je bezstavový, to znamená, že každý požadavek od klienta obsahuje veškeré informace potřebné k jeho zpracování. Díky své jednoduchosti a nezávislosti na konkrétních programovacích jazycích se REST API široce uplatňuje při komunikaci webových aplikací se serverem.

JSON

JSON (JavaScript Object Notation) je lehký formát pro výměnu dat. Při REST komunikaci se JSON používá pro serializaci požadavků i odpovědí a umožňuje přenos komplexních datových struktur přes HTTP ve formě textu [5]. Data zapsaná do JSON formátu mohou vypadat například takto (viz 2.2):

```
{"name":"Marie", "age":20, "date":"06-09-2025"}
```

Obrázek 2.2: Ukázka dat serializovaných v JSON formátu

HttpClient v Blazoru

V Blazor aplikacích se pro komunikaci se serverem standardně používá třída `HttpClient`, která je součástí .NET. S jejím prostřednictvím lze asynchronně odesílat HTTP požadavky

[2]Každý typ zdroje (například uživatelé, produkty nebo objednávky) má svou specifickou URI strukturu, která umožňuje přístup k jednotlivým instancím těchto dat. Například URI `/users/123` může označovat konkrétního uživatele s ID 123.

a přijímat serializovaná data v odpovědích. `HttpClient` podporuje konfigurovatelné hlavičky, zpracování chybových statusů a integrované připojení k autentizačním mechanismům (např. JWT[3]).

2.2.3 WebAssembly

WebAssembly (wasm) je otevřený standard vyvíjený pod záštitou W3C, který definuje formát nízkoúrovňového kódu určeného pro efektivní a rychlé provádění v různých prostředích. Původně byl navržen s cílem umožnit běh výkonných aplikací přímo v prohlížeči, ale není omezen pouze na web – neobsahuje totiž žádné specificky webové funkce ani závislosti. Díky své přenositelnosti, bezpečnosti a kompaktnosti je vhodný také pro další scénáře mimo webové aplikace [9].

Výhody

Zásadní výhodou použití WebAssembly je jeho výkon. Wasm dosahuje výkonu téměř na úrovni nativního kódu díky tomu, že kód je před spuštěním zkompilován do binárního formátu [10], který se velmi blíží strojovému kódu. Tento přístup umožňuje využít pokročilé optimalizace. Jak uvádí článek na Mozilla Hacks [17], WebAssembly se vyhne časovým nákladům spojeným s parsingem a dynamickým zjišťováním typů, které jsou nezbytné u JavaScriptu, a tím pádem odstraňuje potřebu opakovaného optimalizačního cyklu (tzv. reoptimizing), který je běžný v JIT kompilaci JavaScriptu. Díky těmto mechanismům je WebAssembly schopno efektivněji využívat hardwarové zdroje, což vede ke konzistentně rychlejšímu provádění kódu.

Tyto vlastnosti jsou velmi cenné při provádění výpočetně náročných úloh – například při 3D renderování, fyzikálních simulacích, komplexních matematických výpočtech či manipulaci s videi s velkým rozlišením [30]. I když je potřeba počítat s minimálními režijními náklady způsobenými sandboxovaným prostředím a občasným zprostředkováním komunikace s webovými API, tato mírná ztráta výkonu je většinou zanedbatelná ve srovnání s hlavní výhodou, kterou WebAssembly nabízí.

Další výhodou WebAssembly je, že běží ve většině moderních prohlížečů (např. Chrome, Firefox, Safari, Edge) bez ohledu na operační systém [26]. Díky tomu lze kód napsat jednou a spustit jej napříč různými platformami – ať už se jedná o Windows, Linux, macOS nebo mobilní systémy jako Android a iOS. Tato nezávislost výrazně usnadňuje nasazení a údržbu aplikací.

Omezení

Jednou z nevýhod použití wasm může být nadměrná velikost stahovaných balíčků. Aplikace založené na WebAssembly mají obvykle nezanedbatelnou počáteční velikost a jejich stahování tím pádem může prodloužit dobu prvotního načtení.

Dalším omezením, se kterým je při práci s wasm potřeba počítat, je omezený přístup k nativním webovým API. Přímý přístup k DOM není v rámci wasm podporovaný a vyžaduje využití interakce s JavaScriptem. Wasm tedy není vhodný pro aplikace, založené na časté a přímé interakci s DOM.

[3] JWT (JSON Web Token) je otevřený standard (RFC 7519) pro bezpečný přenos informací mezi stranami ve formátu JSON. Více informací o JWT zde: https://datatracker.ietf.org/doc/html/rfc7519

2.2.4 Frameworky pro frontend vývoj

Tato sekce popisuje dva alternativní frameworky pro vývoj webových aplikací: React a Angular, a nastiňuje jejich odlišnosti v porovnání s frameworkem Blazor.

React

React je open-source knihovna vyvinutá společností Meta (dříve Facebook) pro tvorbu uživatelských rozhraní. Umožňuje vytvářet komponentově orientované aplikace, kde každá část rozhraní je definována jako samostatná komponenta. Díky využití virtuálního DOM React efektivně aktualizuje pouze ty části stránky, které se změnily, což zvyšuje výkon aplikací. React je deklarativní, což znamená, že vývojáři definují, jak by mělo rozhraní vypadat v různých stavech, a React se postará o aktualizace. [7]

Angular

Angular je open-source framework vyvíjený společností Google pro tvorbu webových aplikací. Je postaven na jazyce TypeScript a poskytuje kompletní sadu nástrojů pro vývoj, včetně směrování, formulářů, HTTP služeb a dalších. Angular využívá komponentově orientovanou architekturu a dvoucestné datové vazby, což umožňuje synchronizaci dat mezi modelem a pohledem. Díky vestavěnému systému pro vkládání závislostí (dependency injection) podporuje modularitu a testovatelnost kódu. [11]

Porovnání s Blazorem

Na rozdíl od Reactu a Angularu, založených na JavaScriptu (resp. TypeScriptu v případě Angularu), Blazor nabízí integraci s ekosystémem .NET, včetně přístupu k existujícím knihovnám a nástrojům. Blazor eliminuje potřebu JavaScriptu pro mnoho scénářů a umožňuje vývojářům využívat jednotný jazyk napříč celou aplikací, což je výhodné zejména pro vývojáře, kteří jsou zaměřeni na práci v .NET.

Co se týče komunity a dostupnosti knihoven, React a Angular mají výhodu díky své delší existenci a širokému přijetí v průmyslu. Blazor však rychle získává na popularitě, zejména mezi vývojáři, kteří hledají alternativu k JavaScriptovým frameworkům a chtějí využít své znalosti C#.

Celkově lze říci, že volba mezi Blazorem, Reactem a Angularem závisí na konkrétních potřebách projektu a stávajícím technologickém stacku. Pro vývojáře se znalostmi .NET může Blazor představovat efektivní řešení pro vývoj moderních webových aplikací [32].

2.2.5 Blazor

Blazor je webový framework od společnosti Microsoft, který umožňuje vývojářům psát webové aplikace s využitím jazyka C#. Poskytuje možnost tvořit klientskou i serverovou logiku v jednotném jazyce C# s využitím Razor syntaxe (viz kapitola 2.2.5), čímž odpadá nutnost využívat JavaScript. Blazor rozšiřuje platformu ASP.NET. Při práci s Blazorem je možné čerpat z výhod, jako je podpora asynchronního programování[4] pro efektivní zpraco-

[4]Pro více informací o asynchronním programování v .NET viz https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/.

vání požadavků, či Dependency Injection[5] pro přehlednější správu závislostí mezi komponentami.

Blazor Server vs Blazor WebAssembly

Blazor nabízí dvě hlavní varianty pro vytváření aplikací [14]:

- **Blazor Server**: V této architektuře běží aplikace na serveru a interaguje s uživatelským rozhraním ve webovém prohlížeči pomocí SignalR, což je knihovna pro real-time webovou komunikaci. Klientová strana je tedy relativně lehká, jelikož většina zpracování dat probíhá na serveru.
- **Blazor WebAssembly**: Tato verze umožňuje spouštění C# kódu přímo v prohlížeči pomocí technologie WebAssembly. V tomto případě je celá aplikace, včetně business logiky, nasazena a spuštěna v prohlížeči uživatele, což umožňuje vytvářet plně offline aplikace. Tato varianta je ideální pro tvorbu aplikací, které provádějí operace s vyššími nároky na výkon, jako jsou například úpravy snímků videa.

Komponentový model

Podobně jako u Reactu jsou základními stavebními jednotkami aplikací, vytvářených v Blazoru komponenty. Komponenty se zapisují jako soubory s příponou `.razor`. Razor je značkovací syntaxe, která umožňuje definovat chování Blazor komponent. K rozlišení mezi HTML a C# kódem se používají značky `@`, které označují začátek C# kódu. Tato vlastnost odlišuje způsob zápisu Razor syntaxí od jiných frameworků, které často používají složitější způsoby zápisu, jako `{{...}}` v šablonovacím jazyce HandleBars [24] nebo `<%...%>` ve starších verzích ASP [20]. Pro názornost je zde uvedeno porovnání jednotlivých způsobů zápisu zobrazení proměnné v HTML šabloně 2.3:

```
Razor syntax: <p>@Name</p>
HandleBars syntax: <p>{{Name}}</p>
ASP syntax: <p><%=Name%></p>
```

Obrázek 2.3: Porovnání syntaxí

Způsob zápisu komponent do souborů je popsán v oficiální dokumentaci Blazor frameworku takto [14]:

> Třída komponenty je obvykle napsána ve formě stránky jazyka značek Razor s příponou souboru `.razor`. Komponenty v Blazor jsou formálně označovány jako komponenty Razor, neformálně jako komponenty Blazor. Razor je syntaxe pro kombinování jazyka značek HTML s kódem jazyka C#, která zvyšuje produktivitu vývojářů. Razor umožňuje přepínat mezi jazykem značek HTML a jazykem C# ve stejném souboru...

Častým motivem pro využití komponent je možnost komponenty znovu využít v různých částech aplikace. Komponenty v Blazoru toto umožňují několika způsoby.

[5] Pro více informací o Dependency Injection v .NET viz https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection.

Každá komponenta může přijímat vstupní parametry z nadřazené komponenty pomocí atributu `[Parameter]`, čímž dochází ke snadnému předávání dat v rámci komponentové hierarchie. Pokud je potřeba předat parametr více komponentám vnořeným hlouběji ve stromu, lze využít tzv. `[CascadingParameter]`, který umožňuje šířit hodnotu napříč více úrovněmi bez nutnosti explicitního předávání skrze každý mezičlánek. V případech, kdy má být parametr povinný, může být označen atributem `[EditorRequired]`, což zajistí upozornění při kompilaci, pokud parametr nebyl komponentě předán.

Data binding a události

Kromě parametrů mohou komponenty reagovat na uživatelské akce prostřednictvím událostí. Událostmi mohou být například kliknutí na tlačítko, změna hodnoty ve formulářovém poli nebo interakce s dalšími prvky uživatelského rozhraní. Tyto události se v Blazoru obvykle obsluhují pomocí direktiv jako `@onclick`, `@onchange` či `@oninput`, které umožňují propojit prvky rozhraní s metodami definovanými v jazyce C#.

Životní cyklus komponent

Komponenty v Blazoru procházejí definovaným životním cyklem, který umožňuje efektivní správu jejich stavu, načítání dat a interakce s uživatelským rozhraním. Porozumění jednotlivým fázím tohoto cyklu je velmi důležité pro optimalizaci výkonu aplikace a správné načasování asynchronních operací. Následující přehled životního cyklu komponent v Blazoru vychází z oficiální dokumentace Microsoftu [15].

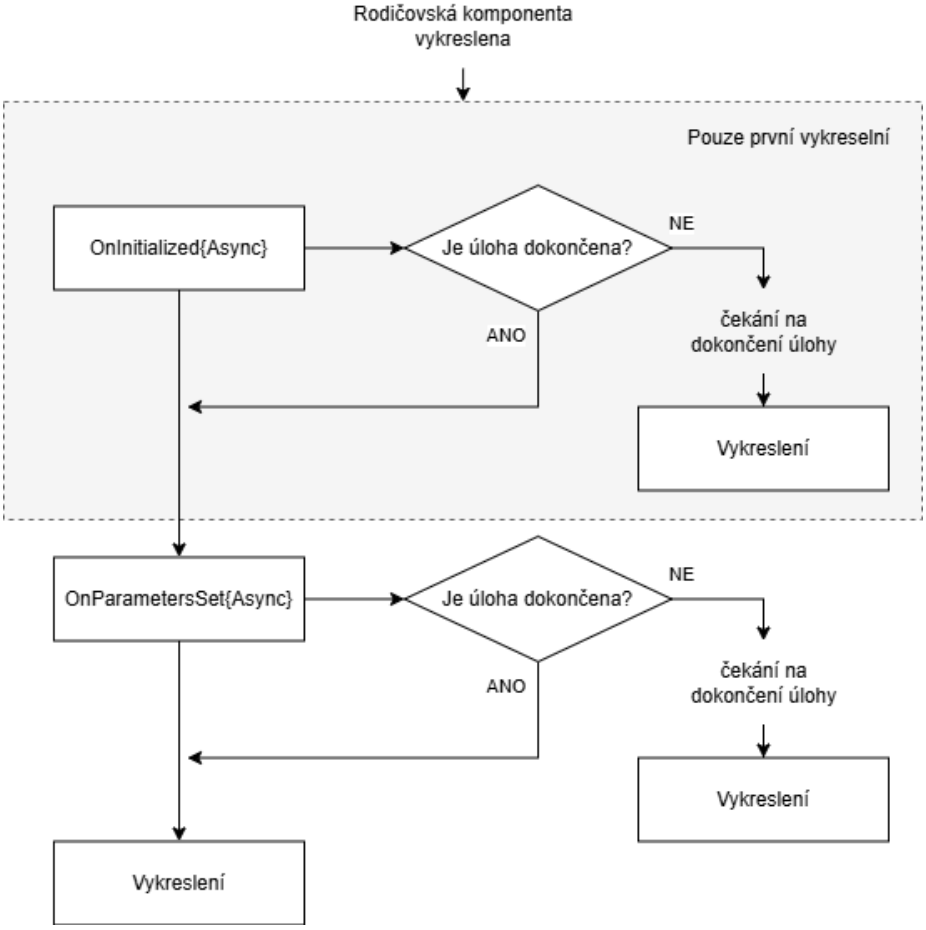
Inicializace komponenty: Při prvotní inicializaci komponenty se volají metody `OnInitialized` a její asynchronní varianta `OnInitializedAsync`. Metody jsou použity v případech, kde je potřeba provést nastavení počátečních hodnot komponenty, případně načtení dat z externích zdrojů. Použití asynchronní varianty je výhodné při operacích, které mohou trvat déle a nemají okamžitý výsledek.

Nastavení parametrů: Když komponenta dostane nové vstupní parametry nebo se změní jejich hodnota, zavolají se metody `OnParametersSet` a `OnParametersSetAsync`. Pomocí těchto metod může komponenta reagovat na změny dat, která jí byla předána z nadřazené komponenty přes atributy jako `[Parameter]` nebo `[CascadingParameter]`. Díky tomu komponenta pracuje vždy s aktuálními hodnotami a může podle nich upravit svůj stav nebo chování. Celý tento proces, od inicializace komponenty až po její vykreslení, znázorňuje obrázek 2.4.

Řízení renderování: Metoda `ShouldRender` umožňuje vývojáři určit, zda by se komponenta měla znovu vykreslit, a tím zabraňuje zbytečným aktualizacím uživatelského rozhraní. Tímto způsobem je možné vynutit opětovné renderování komponenty pouze v případě, kdy se změnila podstatná data.

Post-renderovací logika: Po vykreslení komponenty do DOM se zavolají metody `OnAfterRender` a její asynchronní verze `OnAfterRenderAsync`. Tyto metody se používají pro akce, které závisí na tom, že je komponenta úplně vykreslená – například pro inicializaci JavaScriptových funkcí pomocí JS interopu nebo pro jiné úpravy, které potřebují viditelný prvek na stránce.

Uvolnění zdrojů: Pokud komponenta pracuje s externími zdroji nebo se například registruje na nějaké události, měla by implementovat rozhraní `IDisposable`. Metoda `Dispose` zajistí, že když se komponenta odstraní z uživatelského rozhraní, všechny používané zdroje se uvolní, což pomáhá předcházet problémům, jako jsou paměťové úniky.



Obrázek 2.4: Průběh inicializace a vykreslení komponenty v Blazoru

Explicitní aktualizace zobrazení: V některých případech se může stát, že se změní stav komponenty, ale systém automatického renderování tyto změny nezachytí. V takových situacích lze použít metodu `StateHasChanged()`, která ručně vyvolá aktualizaci uživatelského rozhraní.

2.3 Analýza konkurence

Pro dosažení balancu mezi funkcionalitou a jednoduchostí je vhodné se inspirovat již existujícími řešeními, využít jejich výhod a zároveň se vyhnout jejich nedostatkům. Na trhu se v dnešní době vyskytuje mnoho konkurenčních ITS produktů, které nabízejí různé typy GUI, většinou s podobnými společnými rysy. Následující kapitola se zaměřuje na analýzu současného stavu těchto konkurenčních produktových webů a jejich vlastností.

Metodologie

Pro analýzu konkurenčních produktových webů byly použity metody srovnávací analýzy a SWOT analýzy. Konkurencí jsou v tomto případě myšleny ostatní ITS produkty, jejich produktové weby a jiný software nabízející podobné funkce a služby.

Srovnávácí metoda

Komparační nebo srovnávací metoda, jak je popsáno v Sociologické encyklopedii Sociologického ústavu AV ČR, je široce používána ve vědách, včetně sociologie, historiografie, kulturní a sociální antropologie, demografie a jazykovědy. Tato metoda zahrnuje specifikaci předmětu srovnání, vymezení srovnávaných vlastností, posouzení komparability, určení konkrétních technik srovnávání a způsob zhodnocení získaných informací [28]. Metoda je použita pro různé účely, včetně deskripce, generalizace, klasifikace, hledání kauzálních a funkčních souvislostí, predikce a testování hypotéz.

SWOT analýza

Pro analýzu konkurenčních produktových webů byla dále použita metoda SWOT analýzy. SWOT analýza je strategický nástroj používaný k hodnocení silných a slabých stránek, příležitostí a hrozeb (Strengths, Weaknesses, Opportunities, Threats) organizace, projektu nebo osobní kariéry. Tento model pomáhá identifikovat faktory, které ovlivňují úspěch dané entity. Metodika analýzy je blíže popsána v knize *Business policy: text and cases.* [25]

Přehled konkurenčních produktových webů

Následuje stručný přehled analyzovaných konkurenčních produktových webů a společností, které je vyvinuly.

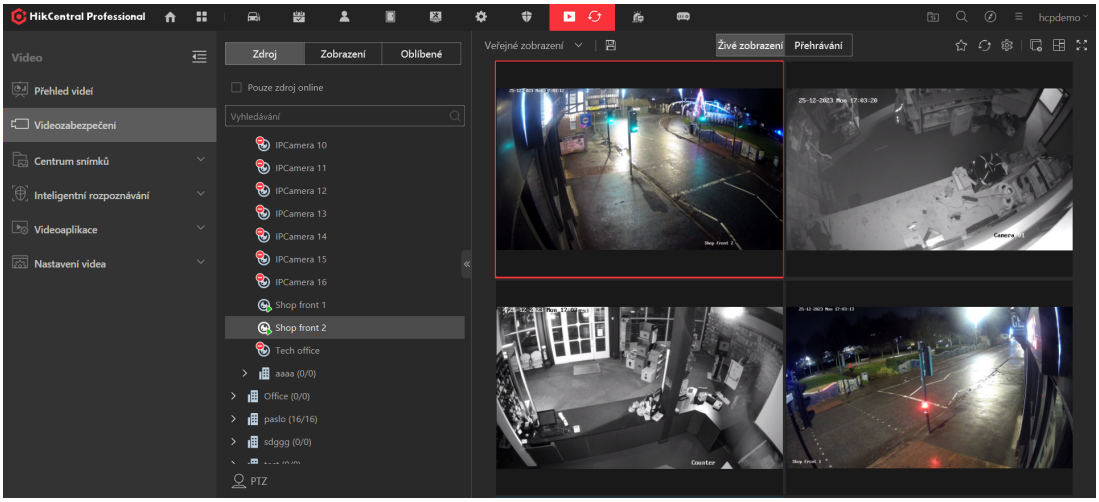
- **HikCentral Professional:** Modulární softwarová platforma pro integraci a správu bezpečnostních systémů společnosti Hikvision.
- **Tattile Pathfinder:** Softwarový nástroj vyvinutý společností Tattile, který slouží k detekci, konfiguraci a správě jejich zařízení v rámci lokální sítě.

HikCentral Professional

HikCentral Professional [2] je centralizovaná platforma pro správu bezpečnostních systémů společnosti Hikvision. Tento systém umožňuje monitorování, správu a analýzu dat z různých zařízení, jako jsou kamery, kontrolní body a alarmy. HikCentral Professional je známý svým intuitivním, uživatelsky přívětivým grafickým uživatelským rozhraním, které poskytuje snadný přístup k hlavním funkcím a údajům. Rozhraní zahrnuje funkce jako živý přenos z kamer, přehrávání nahrávek, pokročilé vyhledávání a analýzu, což umožňuje uživatelům rychle a efektivně reagovat na bezpečnostní události. HikCentral Professional také nabízí pokročilé funkce, jako je rozpoznávání obličejů, analýza davu a integraci s různými typy senzorů a systémů [2].

Živý přenos obrazu kamery

Společnost Hikvision nabízí zkušební demo verzi aplikace HikCentral Professional, která je na omezený čas volně dostupná na jejich webových stránkách. Tato demo verze umožňuje uživatelům dočasně si vyzkoušet všechny funkce HikCentral Professional, včetně živého přenosu z kamer, přehrávání nahrávek a pokročilé analýzy. Z obrázku 2.5 je patrné, že aplikace je členěna do několika hlavních částí, které jsou přístupné z hlavního menu (nahoře). Zbytek zobrazené stránky je rozdělen do tří sloupců, které obsahují různé informace a ovládací prvky. V levém sloupci je umístěn ovládací panel, který umožňuje uživatelům přepínat mezi jednotlivými funkcemi zpracování dat. Prostřední sloupec obsahuje seznam kamer,

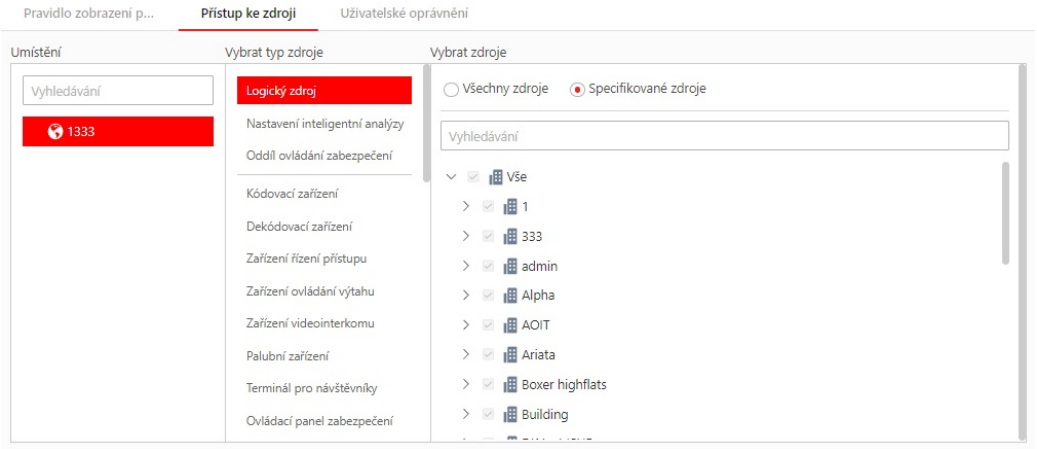


Obrázek 2.5: Demo aplikace HikCentral Professional společnosti Hikvision

které jsou připojeny k systému. V pravém sloupci jsou zobrazeny informace o vybraných kamerách, včetně živého přenosu nebo přehrávání nahrávek. Aktuálně aplikace zobrazuje živý přenos kamerových dat.

Uživatelské role

HikCentral Professional nabízí možnost nastavovat několik uživatelských rolí, které jsou přiřazeny jednotlivým uživatelům. Podle role je možné nastavit uživateli přístup k určitým funkcím a datům a vytvořit tak uživateli pohled na míru. V rámci nastavení rolí je možné změnit účinné období dané role, přiřazení sad či konkrétních pohledů kamer. Je tedy možné mít roli, kde jeden uživatel má například přístup ke kamerovým pohledům, zatímco jiný uživatel může mít pouze nastavené notifikace v případě, že se v budově spustí alarm. Aplikace dále nabízí možnost vytvářet šablony rolí, které je možné přiřadit více uživatelům najednou, viz 2.6.



Obrázek 2.6: Nastavení oprávnění zdrojů

Zhodnocení silných a slabých stránek

Software HikCentral Professional má několik výhod, které by měly být při návrhu nového uživatelského rozhraní zohledněny. Tyto výhody lze nazvat SWOT silnými stránkami:

- **Rozvržení stránky**: Aplikace je přehledně rozdělena do několika částí, které jsou přístupné z hlavního menu. Tento přístup umožňuje uživatelům snadno přepínat mezi jednotlivými funkcemi.
- **Manipulace pohledů kamer**: Další výhodou je možnost intuitivního přidávání a odebírání kamer, jejichž pohledy chce uživatel zobrazit. Kamery lze mezi sebou snadno přesouvat a přeskupovat (přetažením na místo určení), díky čemuž je aplikace intuitivní a ergonomická.

Na druhou stranu, aplikace HikCentral Professional má několik nedostatků, které by měly být při návrhu nového uživatelského rozhraní zohledněny. Tyto nedostatky lze nazvat SWOT hrozbami:

- **Přehlednost na malých obrazovkách**: Aplikace obsahuje mnoho informací, které mohou být pro uživatele matoucí. Obzvláště na mobilních zařízeních, kde je omezená velikost obrazovky, by byla tato aplikace téměř nepoužitelná. Hikvision tento problém vyřešila implementací mobilní aplikace HikCentral Mobile, která je optimalizována pro mobilní zařízení a tablety. Pro návrh nového uživatelského rozhraní je proto nutné zohlednit velikost obrazovky a přizpůsobit GUI tak, aby na malých obrazovkách zobrazovalo méně dat, případně je zobrazovalo jiným způsobem.

Pathfinder

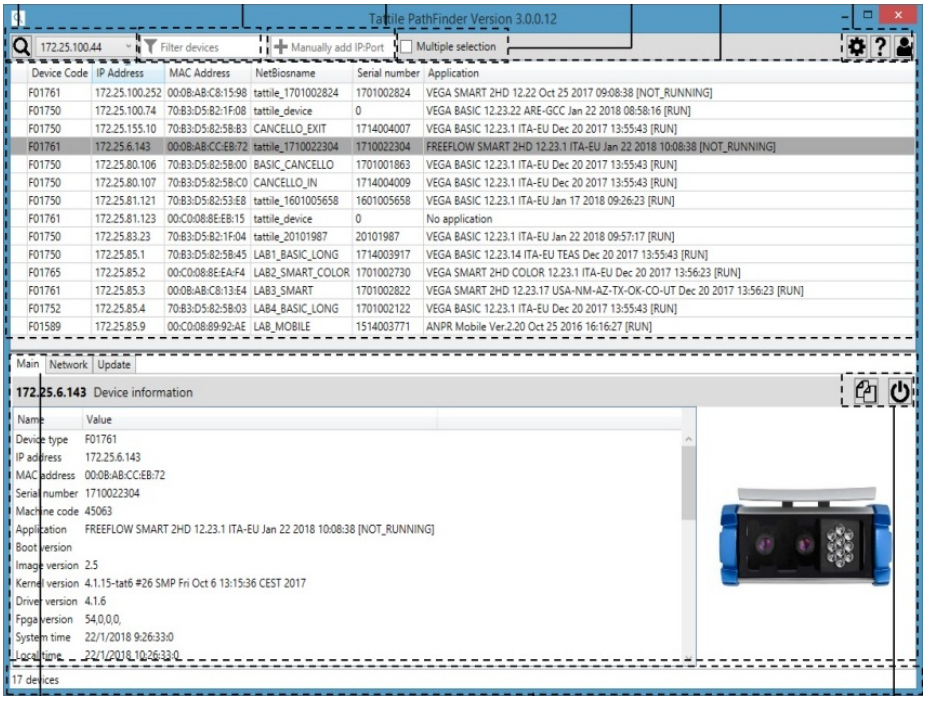
Pathfinder je desktopová aplikace pro správu zařízení, v čele s chytrými kamerami společnosti Tattile. Aplikace převážně slouží k objevování a základní správě zařízení v síti a je volně dostupná na webových stránkách společnosti Tattile [31]. Aplikace slouží pouze k části správy zařízení, která je základní a neposkytuje mnoho funkcí. Umožňuje například zařízení vzdáleně konfigurovat nebo aktualizovat firmware. Aplikace zdaleka není tak funkčně rozsáhlá jako její konkurence společnosti Hikvision. Nabízí však základní funkce, které jsou pro správu zařízení nezbytné. Aplikace obsahuje dvě hlavní okna a lišty s prvky určenými pro vyhledávání a filtrování sledovaných zařízení (viz. obrázek 2.7). První okno zobrazuje seznam nalezených zařízení, informace o nich a jejich stav. Druhé okno zobrazuje detaily o vybraném zařízení.

Zhodnocení SWOT

Aplikace nabízí některé vlastnosti, které za určitých okolností lze nazvat SWOT příležitostmi:

- **Dělení stránky**: Aplikace je dělená na seznam zařízení a detaily o vybraném zařízení. Při výběru zařízení je možno (i když v tomto konkrétním provedení ne úplně přehledně) logicky oddělit, o kterém zařízení chceme vidět detail, případně které zařízení chceme nastavovat. Určitou formu této vlastnosti v lepším provedení lze vidět i u aplikace HikCentral Professional 2.3.

Na druhou stranu, ale obsahuje řadu nedostatků, které při návrhu nového uživatelského rozhraní je nutné zohlednit:



Obrázek 2.7: Aplikace Pathfinder společnosti Tattile

- **Nepřehlednost**: Aplikace je nepřehledná a tím pádem těžko použitelná. Pro uživatele to pak znamená, že mají problém s orientací v aplikaci a hledáním informací o zařízeních. Tato nepřehlednost je způsobena především snahou zobrazit co nejvíce informací zaráz. Najdou se případy užití, kdy je tato vlastnost žádoucí, avšak většinou je to na úkor přehlednosti. O potřebě přehlednosti dále pojednává kapitola 2.4.1.
- **Podpora mobilních zařízení**: Aplikace není optimalizována pro mobilní zařízení. Při návrhu nového uživatelského rozhraní je potřeba zohlednit podporu mobilních zařízení, protože část uživatelů produktových webů může preferovat používat mobilní zařízení.

2.4 Shrnutí požadavků

Tato kapitola nepřímo navazuje na předchozí kapitolu, ve které byly analyzovány současné ITS produkty a jejich grafická rozhraní. Z pozorování v předchozí kapitole vyplývá, že existující produktové weby mají několik výhod a nevýhod, které by měly být při návrhu nového uživatelského rozhraní zohledněny. Tato kapitola se proto zabývá shrnutím požadavků na nové uživatelské rozhraní.

Na požadavky na produktové weby ITS produktů lze nahlížet z několika různých úhlů pohledu. V této kapitole jsou shrnuty požadavky z pohledu obecných vlastností, kterých by mělo dané rozhraní dosahovat. V další části kapitoly je řešena problematika z pohledu konkrétních funkcí, kterých by měla výsledná aplikace dosahovat. Tím získáme bližší obraz toho, jak by mělo výsledné rozhraní vypadat a jaké funkce by mělo obsahovat.

Následuje výčet požadavků na webové GUI pro ITS produkty. Tyto požadavky odrážejí vlastnosti, kterých by mělo GUI dosahovat podle předchozí analýzy současného stavu ITS produktů včetně analýzy konkurence.

2.4.1 Obecné požadavky

Tato podkapitola čerpá z informací získaných z knihy *Strategický marketing* docentky doc. Ing. Heleny Horákové, CSc. z Jihočeské univerzity v Českých Budějovicích [21], která poskytuje užitečné informace o návrhu uživatelských rozhraní. Další sekce se zabývají obecnými požadavky na výslednou aplikaci.

Uživatelská přívětivost

První z požadavků pro výslednou aplikaci je dosti intuitivní. Jedná se o obecnou uživatelskou přívětivost. Výsledný web by neměl obsahovat přehršel funkcí a informací, které uživatelé v danou chvíli nevyužijí. Web by měl působit intuitivním a snadno použitelným dojmem.

Adaptabilita

Schopnost rozhraní přizpůsobit se různým typům uživatelů a zařízení. Toto zahrnuje alespoň omezenou podporu na zařízeních s menšími nebo dotykovými obrazovkami, jako jsou mobilní telefony či tablety. Je vhodné, avšak ne zcela zásadní, využít nástrojů pro tvorbu responzivních grafických uživatelských rozhraní [27] jako je například framework Bootstrap[6].

2.4.2 Specifické funkční požadavky

Tyto sekce se zabývají specifickými požadavky na funkce výsledné aplikace.

Multiplatformnost

V rámci zadání byl stanoven předpoklad, že GUI bude použito u produktů, které běží na různých platformách (ARM, Intel, mikrokontrolery). GUI proto musí být navrženo a implementováno tak, aby bez úprav běželo na všech cílových platformách produktu. Předpokládá se použití přenosných knihoven a nástrojů, které zajišťují konzistentní vzhled i chování ovládacího rozhraní napříč různými hw architekturami.

Živý přenos obrazu kamery

Při nastavování kamer a dalších zařízení je důležité mít možnost sledovat živý přenos obrazu z kamer. Pro tyto účely je vhodné podporovat některé funkce spojené s obrazem (přehrávání/pozastavení videa, fullscreen).

Nastavení parametrů zařízení

Další funkcí, kterou by měla aplikace obsahovat, je možnost nastavení parametrů zařízení. V kontextu chytrých kamer se jedná o dálkové nastavování parametrů, jako je expozice, ostření, rozlišení, ořez obrazu a další. Cílem této funkce je umožnit uživatelům nastavit

[6] Více o Bootstrap viz 2.2.1.

parametry kamer bez nutnosti fyzického přístupu k nim. Ideálně by mělo být možné spravovat všechny parametry a konfigurace, které jsou dostupné v rámci daného ITS produktu. Avšak vzhledem k množství těchto parametrů a konfigurací je vhodné zohlednit přehlednost a uživatelskou přívětivost.

Při návrhu výsledné webové aplikace je nutné také zohlednit bezpečnostní aspekty, jako je omezení přístupu k určitým funkcím neautorizovaným osobám. Možnost editace parametrů zařízení by mělo být možné omezit podle přístupnosti daného zdroje.

Client-side

Výsledná aplikace by měla být implementována tak, aby většina výpočetních operací probíhala převážně na straně klienta. To je zapříčiněno tím, že CAMEA produkty nedisponují dostatečným výpočetním výkonem pro provádění výpočetně náročných operací, jakým může být například komplexnější analýza či manipulace s videem v reálném čase. Výpočetně náročné operace by tedy měly být prováděny na straně klientské aplikace.

Výběr frameworků s dlouhodobou podporou

Podle bodu 3 zadání je pro vývoj GUI třeba zvolit takové vývojové frameworky a knihovny, které mají předpoklad dlouhé životnosti a podpory v budoucnu. Je proto potřeba vyhnout se použití technologií, které jsou již zastaralé a nemají zajištěnou dlouhodobou životnost.

2.4.3 Požadavky na vizuál a barvy

Tato kapitola se zabývá vhodným použitím barev v uživatelském rozhraní. Při tvorbě této kapitoly bylo čerpáno z knihy *Colour Psychology* [12], která se zabývá vlivem barev na lidskou mysl a podvědomí. Dále pak z knihy Česko-anglický slovník praktické globální vizuální komunikace doktora Tomáše Fassatiho, která se zabývá interpretací vizuálních symbolů jako formy komunikace [18].

Správné použití barev je důležitou součástí návrhu uživatelského rozhraní, protože výběr vhodných barev může mít pozitivní vliv na uživatelský zážitek. Vhodně zvolené barvy mohou u potenciálních zákazníků ovlivnit výběr produktu a zvýšit tak jeho prodejnost. Výběr barev by měl být proveden tak, aby byl zohledněn cílový trh a cílová skupina uživatelů.

Základní kombinace modré barvy

Pro ITS produkty a obecně pro produkty technického ražení je výhodnější použít chladné barvy, které jsou spojovány s bezpečností, spolehlivostí, logikou, loajalitou a udržovatelností, jako je například modrá a její odstíny. Modrá barva je také často používána v oblasti automatizace, což je oblast, ve které se produkty inteligentních dopravních systémů pohybují. Dosavadní weby společnosti CAMEA využívají modrou barvu, která je použita i v logu společnosti. Modrou dále kombinuje s oranžovou barvou, která bývá spojována s inovací a energií. Dále lze v téměř jakémkoliv případě použít bílou barvu, která je spojována s čistotou, jednoduchostí a dokonalostí. Případně je možné použít černou barvu, která je spojována s bezpečností, sofistikovaností a autoritou. Obě tyto barvy jsou často používány, díky možnosti je snadno kombinovat s jinými barvami a vysokému kontrastu, který obecně zvyšuje čitelnost textu a tím zlepšuje uživatelský zážitek.

Alternativní kombinace k modré barvě

Kromě hojně používané kombinace modré a bílé barvy je možné použít i jiné kombinace, které by daný web odlišily od ostatních technologických společností. Jednou z možností je použití zelené barvy v případě, že by společnost chtěla působit ekologickým či udržitelným dojmem. Další možnou alternativou je použití žluté barvy, která je spojována s energií a optimismem, případně hnědou barvu, která reprezentuje spolehlivost, přírodu a podporu. V případě použití hnědé barvy je však nutné zohlednit nízký kontrast s černou barvou, která je často používána pro text.

Barvy jako prostředek pro zvýraznění důležitých informací

Mimo odstíny barev je možné využít vlastností kontrastu komplementárních barev pro zvýraznění ovládacích prvků aplikace či důležitých informací. V případě, že by byla použita modrá barva pro pozadí, je vhodné použít žlutou či oranžovou barvu pro zvýraznění důležitých informací. Podobně pro naléhavá upozornění nebo varování je vhodné použít červenou barvu, na kterou lidská mysl přirozeně reaguje se zvýšenou pozorností. Tato vlastnost by nebyla použitelná v případě, že by byla použita červená barva pro pozadí. Rozumné použití červené barvy můžeme vidět na obrázku 2.5. Červená barva je použita pro zvýraznění tlačítek, případně aktuálního výběru pohledu kamery či prvku v menu. Na obrázku 2.8 je vidět použitá barevná paleta aplikace HikCentral Professional.

Obrázek 2.8: Barevná paleta aplikace HikCentral Professional

Kapitola 3

Návrh uživatelského rozhraní

Následující kapitola se zabývá samotným návrhem uživatelského rozhraní produktového webu demonstrovaného na inteligentní dopravní kameře. V této kapitole je popsán přístup k návrhu a návrhy jednotlivých částí rozhraní. Výsledný návrh staví na požadavcích, které byly shrnuty v předchozí kapitole.

Tvorba návrhu probíhala zpočátku náčrtem wireframů a následně v aplikaci Figma, kde byly vytvořeny prototypy jednotlivých částí aplikace. Ty pak byly testovány dobrovolníky, z jejichž zpětné vazby byly prototypy dále upravovány, aby splňovaly požadavky na uživatelskou přívětivost (viz 2.4.1). V případě, že prototyp ve Figmě nebyl pro účely testování dostatečný, byla vytvořena prototypická razor komponenta v Blazoru, na které byl daný prototyp testován.

3.1 Přístup k návrhu

V rámci zadání práce bylo navrhnout a realizovat knihovnu komponent specifických pro ITS produkty, za pomocí níž bude následně implementována demonstrační aplikace. Vzhledem k potřebě uvažovat o knihovně jako o samostatném a znovupoužitelném celku byl návrh rozdělen na části: návrh knihovny a návrh aplikace. Návrh knihovny se zaměřuje na to, jak komponenty společně fungují a jak je snadno znovu použít. Určuje jednotný vzhled a způsob, jak předávat parametry a nastavovat jejich chování. Součástí jsou i pravidla pro komunikaci rodič–potomek (např. přes `CascadingValue` nebo události). Návrh aplikace pak demonstruje možnosti kombinace několika komponent knihovny dohromady v uceleném uživatelském zážitku.

Dělení aplikace do komponent

Pro zjednodušení návrhu rozsáhlejšího GUI do přehlednějších menších celků je vhodné rozdělit aplikaci do komponent. Komponenty budou představovat samostatné části aplikace, které mohou obsahovat vlastní logiku a vzhled. Jedna z hlavních výhod takového přístupu při návrhu aplikace je možnost znovupoužití komponent v různých částech aplikace. Díky tomu je možné snadno upravovat a rozšiřovat aplikaci bez nutnosti zásadních změn v kódu. To je samo o sobě dobrý důvod pro použití Blazor frameworku 4.1, jelikož umožňuje tvorbu razor komponent. Některé komponenty budou využívány opakovaně v různých částech aplikace, zatímco některé budou specifické pouze pro určitou část aplikace. Mezi častěji používané komponenty patří například tlačítka, vstupní textová, číselná a datová pole obsažená ve formulářích, menu, informační panely a různé kontejnery pro zobrazení obsahu. Tyto

komponenty nazývám základními komponentami. Dále bude potřeba vytvořit specializované komponenty, které budou svojí funkcí a vzhledem specifické pro dané použití. Mezi tyto komponenty patří například zobrazení obrazu z kamery, ořez části obrazu kamery nebo jiné, v uživatelském rozhraní málo opakované, komponenty. V knihovně se budou tyto specializované komponenty ve velké míře skládat z komponent základních.

3.2 Návrh knihovny

Návrh knihovny byl úzce spjat s návrhem aplikace, protože probíhal nejdříve testováním prototypů demonstrační aplikace. Tyto prototypy byly následně od aplikace odděleny a postupně implementovány do knihovny v dynamických variantách. Tyto nové univerzální komponenty pak bylo možné znovu používat i v dalších částech aplikace. Během prototypování bylo patrné, že některé prototypy knihovních komponent, tak jak byly vytvořeny, nedávaly ve výsledném použití smysl. To zapříčinilo, že se do knihovny implementovalo pár komponent, které ve výsledné aplikaci nebyly použity.

3.2.1 Základní komponenty

V následující kapitole jsou popsány základní komponenty, jejich specifické funkce a způsoby integrace do uživatelského rozhraní jako celku. V rámci práce bude vytvořeno několik typů datových vstupů, které se budou lišit vzhledem a funkcí. Nejčastěji půjde o interakci s datovým modelem systému, ale přesto je dobré definovat některé společné vlastnosti. Aby mohly být komponenty pro vstup dat opravdu intuitivní, budou měnit svůj stav v závislosti na kontextu. V případě, že má například tlačítko potvrzovat nějakou akci, nesmí být přístupné uživateli, dokud nebude akce provedena. Toto vyžaduje, aby rodičovská komponenta, ve které je tlačítko umístěno, mohla ovlivnit jeho stav (například z neaktivní na aktivní). Následně musí být možné z rodičovské komponenty získat informaci o stisknutí tlačítka. Podobně je potřeba informovat rodičovské komponenty i o dalších možnostech, jako je načítání obsahu komponenty, odesílání nových dat nebo čekání na potvrzení akce.

Tlačítka

V rámci aplikace bude vytvořeno několik typů tlačítek, z nichž některá budou mít opakující se funkci. Pro tato využití dává smysl vytvořit komponenty pro jeden specifický účel (například navigaci nebo volání metod rodičovských komponent). Pro tlačítka s obecnější funkcí bude vytvořena komponenta, která bude schopna přijímat parametry a podle nich měnit svůj vzhled a funkci. Vzhled tlačítka by měl jasně indikovat jeho funkci, případně vztah k datovému modelu. Pokud například tlačítko slouží k rušení předchozí akce, je potřeba to uživateli jasně vizuálně sdělit například barvou či ikonou[1]. Základními parametry pro ovládání tlačítek tedy budou aktivita (možnost tlačítko stisknout), zaneprázdněnost (vizuální náznak, že požadovaná akce se již provádí), ikona a případně text pro vysvětlení méně intuitivní akce. Dalším parametrem bude odkaz na metodu, zodpovědnou za provádění akce.

[1] Více informací o využití barev a jejich významu na uživatelský prožitek viz kapitola 2.4.3.

Slidery

Komponenty, které umožňují uživateli nastavit hodnotu v určitém rozsahu. Pro slider dává smysl vytvořit jednu univerzální komponentu, která bude nastavena pomocí parametrů, jako jsou minimální a maximální hodnota, krok, výchozí hodnota a další. V rámci slider komponenty dává smysl připojit volitelné zobrazení označení názvu slideru, hodnotu a případně jednotky, ve kterých daná hodnota operuje. Protože akce slideru se většinou nepotvrzují prováděním dalších akcí (stisknutí tlačítka pro potvrzení), bude každou změnu hodnoty slideru možné zaznamenat a případně ignorovat, dokud nebude akce potvrzena rodičovskou komponentou. Tím je možné přizpůsobit chování slideru konkrétním potřebám cílové aplikace.

Formuláře

Jeden z nejuniverzálnějších způsobů pro získání dat od uživatele jsou formuláře. Obvykle v ergonomických uživatelských rozhraních chceme použití obyčejných formulářových vstupů minimalizovat, ale vzhledem k velkému množství parametrů, které lze v rámci systému nastavit, se bez určitého množství formulářů neobejdeme. Vstupy, které lze v rámci spravovaného ITS systému nastavovat, jsou:

- **Textové pole**: Bude použitý pro zadávání plain textových dat. V kontextu ITS se jedná o například název zařízení, sériové číslo dílu, či název podporovaného software. Pro tento účel byla zvolena komponenta `TextBox` knihovny Radzen, která oproti standardnímu input elementu nabízí pokročilejší možnosti kontroly vstupů.
- **Časové vstupy**: V rámci systému je možné nastavovat pár časových parametrů, které jsou předávány v hodnotě `string` datového typu. Pro vkládání časových údajů bude použita kalendářová Radzen komponenta, která umožňuje nastavení přesnosti časového vstupu a implementaci vlastních tlačítek do stejné komponenty.
- **Číselné vstupy**: V systému je třeba zadávat různé číselné hodnoty, například identifikátory, porty a množství parametrů nastavení jednotlivých kamerových vstupů. Tyto vstupy se dělí podle očekávaného typu na celá čísla se znaménkem (`int`) a bez znaménka (`uint`). Pro zajištění správnosti vstupu a eliminaci neplatných hodnot byla zvolena komponenta `Numeric` z knihovny Radzen. Tato komponenta umožňuje nastavení minimální a maximální hodnoty, krokování (increment/decrement), a zároveň poskytuje možnost validace vstupních hodnot na základě datového typu. V případě požadavku na nezáporné hodnoty (např. počty, indexy) je nastaveno minimum povolené vstupní hodnoty.

  Kromě celých čísel systém operuje také s reálnými hodnotami typu `float` a `double`, které jsou nejčastěji využívány při nastavování optických kamerových parametrů (např. nastavení clony, světlosti, atd.). I pro tyto typy čísel je možné využít komponentu `Numeric`, která podporuje desetinné vstupy a umožňuje definovat přesnost pomocí parametru `Step`.

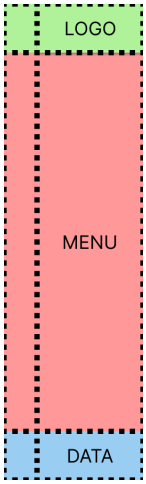
3.2.2 Specializované komponenty

V následující kapitole jsou popsány návrhy specializovaných komponent a jejich požadovaných funkcí:

Navigační panel

Knihovna nabídne responzivní boční nebo horní navigaci, která se automaticky přizpůsobuje různým rozlišením. Na větších obrazovkách zůstává panel ve zúženém stavu, ale při najetí kurzoru myši se rozvine tak, aby nezasahoval do hlavního obsahu. V mobilním zobrazení je panel skryt za tlačítkem hamburger menu a otevře se na požádání.

Detailnější náhled způsobu, jakým se bude moct komponenta rozvinout, je vidět na obrázku 3.1. Obsahuje tři hlavní sekce: **logo**, reprezentující danou společnost, **menu**, obsahující odkazy na stránky, a **data**, která bude moci zobrazovat (systémový čas a stav připojeného zařízení).



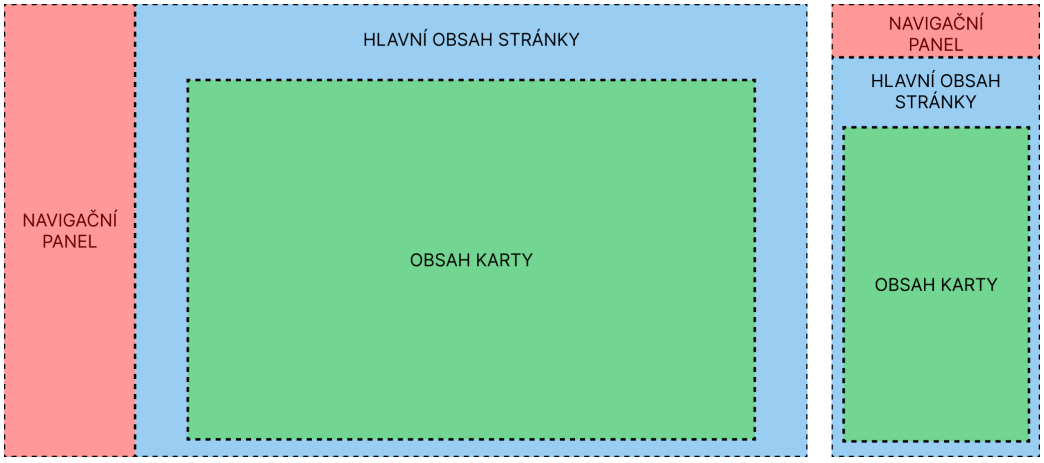
Obrázek 3.1: Wireframe bočního navigačního panelu

Kontejnery

Pro správné rozložení komponent po stránce je potřeba definovat, jakým způsobem se mají komponenty zarovnávat v závislosti na jejich velikosti, množství, dostupném místě na stránce a rozměrech obrazovky zařízení. Komponenty, zajišťující tuto funkci, nazývám obecně kontejnery. Mimo zarovnávání obsažených komponent kontejnery plní funkci vizuálního oddělení obsahu na stránce a tím vedou koncového uživatele aplikace k prvkům aplikace, se kterými chce v danou chvíli interagovat.

Zde jsou některé hlavní kontejnerské komponenty, které v systému využiji:

- **Rozložení stránky (`PageContent`)**: Komponenta kontejneru pro rozložení stránky bude sloužit pro zarovnání hlavního obsahu na stránce spolu s navigačním panelem. Jedním z úkolů této komponenty bude řešení zarovnávání obsahu na různých zařízeních. Komponenta bude nabízet dva hlavní pohledy. První pohled bude nabízet rozložení stránky na zařízeních s větším obrazovkou a druhý pohled tvoří responzivní rozložení pro menší obrazovky převážně mobilních zařízení. Komponenta bude obsahovat vstupní parametry `ShowNavMenu`, `ShowBottomPanel` a `Title` pro nastavení viditelnosti navigačního menu a spodního informačního panelu. Návrh komponenty stránky můžete vidět na obrázku 3.2 v pohledu s viditelným navigačním panelem na větších obrazovkách (vlevo) a v mobilním zobrazení (vpravo).



Obrázek 3.2: Hlavní rozložení stránky

- **Karta (`Card`)**: Komponenta karty bude sloužit jako vizuální oddělení barevného pozadí od specifického obsahu. Je navržena tak, aby seskupovala související prvky a zajišťovala jejich přehledné uspořádání. Uvnitř komponenty se bude obsah zarovnávat pomocí flexibilního rozložení (`flexbox`), které umožňuje responzivní přizpůsobení rozložení podle dostupného prostoru. Tato komponenta bude široce využívána například pro zobrazování živého náhledu, nastavení zařízení nebo přehledy teplot. Jak bude komponenta fungovat na různých velikostech zařízení, je vidět na obrázku zde 3.2. Mimo vizuálních vlastností bude komponenta komunikovat s dětskými komponentami, sbírat data o stavech jejich načítání a případně na popud jiné dětské komponenty (např. tlačítka) aktualizovat jejich hodnoty.
- **Skupina vstupních polí (`Fieldset`)**: Tato komponenta slouží k logickému seskupení souvisejících datových editorů. Těm nabídne jednotné záhlaví, čímž pomůže uživateli nastínit kontext jednotlivých editovaných položek. Záhlaví bude tvořeno textem a případně volitelnou ikonou. Během inicializace bude předán kontext dětským komponentám, které se při inicializaci registrují a sdílejí svůj stav (například informaci o probíhající aktualizaci). Fieldset bude podporovat přepínání mezi zobrazením a editačním režimem s možností hromadného potvrzení změn a zároveň sledovat stav registrovaných `ItemDataEditor` komponent, aby podle potřeby aktivoval nebo deaktivoval ovládací prvky. Tím se zvyšuje přehlednost rozsáhlejších formulářů a zlepšuje uživatelská zkušenost při hromadných změnách dat.

Prvotní nastavení (`InitialSetup`)

Některá zařízení disponují velkým množstvím parametrů, které je třeba nastavit. Při nastavování zařízení uživatel často provádí stejnou sekvenci kroků na různých místech. Nezkušený uživatel se pak může ocitnout v situaci, kdy neví, zda opravdu nastavil vše potřebné.

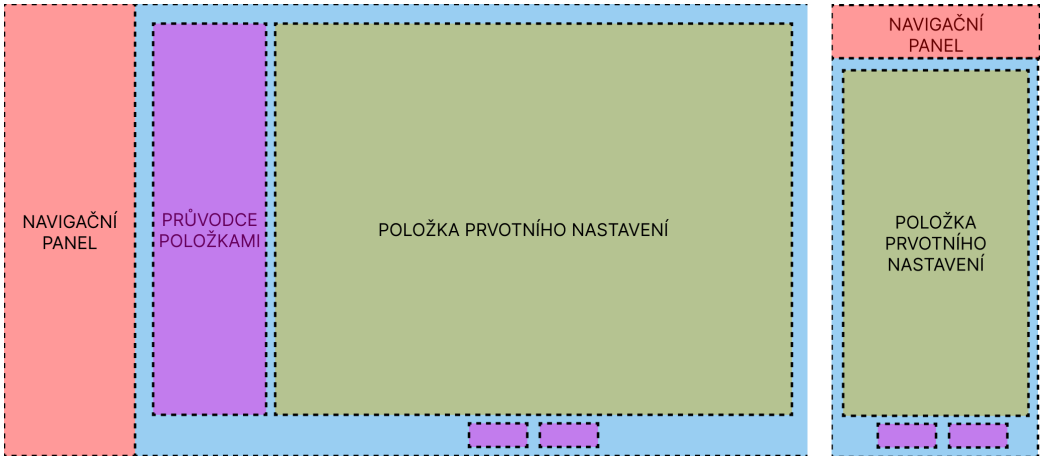
Komponenty `InitialSetupItem` a `InitialSetupWindow` tuto problematiku řeší tak, že umožní předdefinovat rutinní průchody aplikací a intuitivním způsobem zajistí nastavení správných parametrů ve správném pořadí. Tímto způsobem je možné vytvářet rutiny pro například nastavení pouze parametrů souvisejících s připojením k lokální síti nebo pouze některých parametrů obrazu. Mimo to lze tímto způsobem navrhnout celé průchody aplikací v případě, že chce uživatel pouze zařízení nastavit a poté s ním již nekomunikovat.

- **Položka prvotního nastavení (`InitialSetupItem`)**: Slouží jako jeden krok v průvodci prvotního nastavení, ve kterém se seskupují související datové editory a živé náhledy. Každá položka při inicializaci předává kontext rodičovskému oknu, sama se registruje a sleduje stav načítání svých podřízených komponent. Ve chvíli, kdy se načítání dokončí, položka odhalí svůj obsah.

  Součástí položky je také navigační zápatí s tlačítky „Previous“, „Next“ a „Done“, která řídí tok průvodce, deaktivují se během načítání a signalizují uživateli aktuální pozici v sekvenci kroků.

- **Okno prvotního nastavení (`InitialSetupWindow`)**: Hlavní kontejner průvodce, který v levé části zobrazuje kroky jako lištu s vyznačením aktuálního kroku a vedle ní ukazuje oblast s nadpisem a právě aktivním obsahem. Při načtení sbírá informace o všech vložených `InitialSetupItem` komponentách, sestavuje jejich pořadí a řídí přepínání viditelnosti položek podle uživatelských akcí, až po závěrečné přesměrování na cílovou stránku po dokončení průvodce.

Na obrázku 3.3 vidíme, jakým způsobem se bude pár `InitialSetup` komponent chovat v kombinaci s výše zmíněným kontejnerem `PageContent` a navigačním menu.



Obrázek 3.3: InitialSetup navbar

Editor datových položek

Komponenta bude sloužit jako editor jednotlivých parametrů definovaných v DataStructure v konfiguračním souboru (viz 4.1). Cílem je vytvořit prvek, který dokáže v různých kontextech zobrazovat data, umožnit jejich úpravu a zároveň respektovat oprávnění a typovou bezpečnost. Komponenta proto bude podporovat alespoň režimy: běžné zobrazení a editaci parametru. V režimu zobrazení je hodnota prezentována bez možnosti úpravy. V režimu editace komponenta zobrazí vstupní pole odpovídající danému datovému typu a umožní změnu hodnoty. Tento režim bude vhodný tam, kde má uživatel potřebná oprávnění.

Editor dat endpointů

Komponenta bude sloužit jako souhrn pro editaci a zobrazení datových struktur spojených s konkrétním API endpointem. Na základě zadaného identifikátoru načte strukturu dat

a dynamicky generuje odpovídající editační prvky `ItemDataEditor`. Dále umožní indikaci načítání a hromadnou aktualizaci hodnot.

Obraz a editor obrazu

Komponenta `LiveImageEditor` zobrazí živý přenos snímků obrazu kamery. Nabídne základní možnosti editace parametrů, souvisejících přímo s obrazem, jako jsou ořez obrazu, rozlišení nebo úroveň komprese. Živé video samotné bude možné zobrazit ve fullscreen. V rámci vstupních parametrů komponenty bude možnost volby přepnout mezi editačním a zobrazovacím módem, kdy zobrazovací mód obsahuje pouze snímky obrazu.

Výběr zařízení

Komponenta `DeviceSelection` nabídne nástroj pro správu zařízení, načtených z konfiguračního souboru 4.1. Zařízení bude možno dle libosti vybrat, přidávat a editovat. V rámci editace bude možné zařízení vybrat název, ikonu a danou IP adresu, na kterou zařízení odpovídá. V případě výběru zařízení bude automaticky otestováno připojení a zobrazí se zpráva o úspěšnosti připojení zařízení.

Výběr aplikačního módu

Po úspěšném připojení k zařízení bude možno přejít do výběru aplikačního módu. Komponenta `AppModeSelection` nabídne uživateli možné aplikační módy, ze kterých si uživatel může vybrat následující průchod aplikací. `AppModeSelection` tedy bude sloužit jako takové úvodní menu určující další průchod aplikací.

3.3 Návrh aplikace

Návrh aplikace je založen na komponentách knihovny, které byly popsány v předchozí sekci 3.2. Aplikace je koncipována jako demonstrace fungování těchto komponent v jednom celku a skládá se ze stránek, které dávají smysl pro použití na inteligentní kameře UC-SCA[2] společnosti CAMEA. Jedná se o jedno z mnoha možných využití knihovny pro tvorbu Blazor aplikací.

3.3.1 Stránky

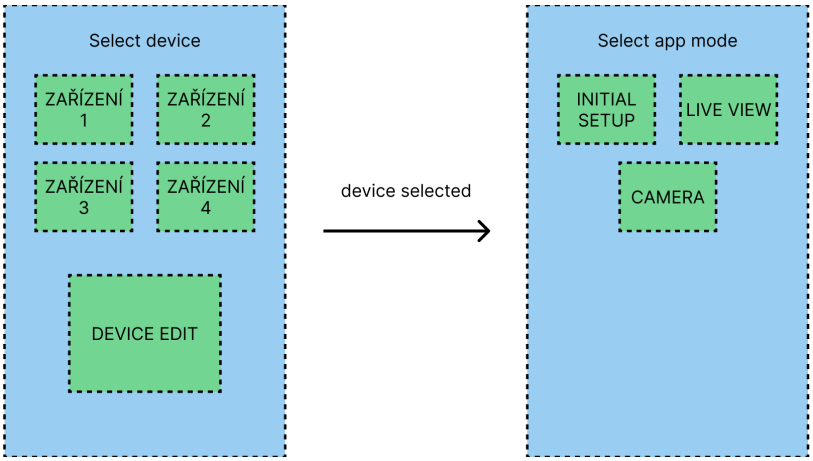
Stránky jsou navrženy s ohledem na uživatelský zážitek a průchod aplikací. Během návrhu byly několikrát měněny podle uživatelského testování jejich prototypů, tak aby ulehčily uživateli práci a pokud možno co nejvíce práce "dělaly za něj".

Výběr zařízení a aplikačního módu

První stránkou, se kterou se uživatel setká, bude výběr zařízení a aplikačního módu. Stránka bude dostupná na routě `/` (nebo alternativně `/manage-devices`) a pomocí komponent `DeviceSelection` a `AppModeSelection` nabídne seznam předkonfigurovaných zařízení s možností přidat nové nebo upravit existující. Po úspěšném výběru zařízení a aplikačního módu bude uživatel odkázán na stránku jemu příslušející: prvotní nastavení, domovská

[2]Kameru můžete vidět na obrázku 2.1, případně její detailní technická specifikace viz https://www.cameatechnology.com/data/files/UC-SCA-dat-en-20200427.pdf.

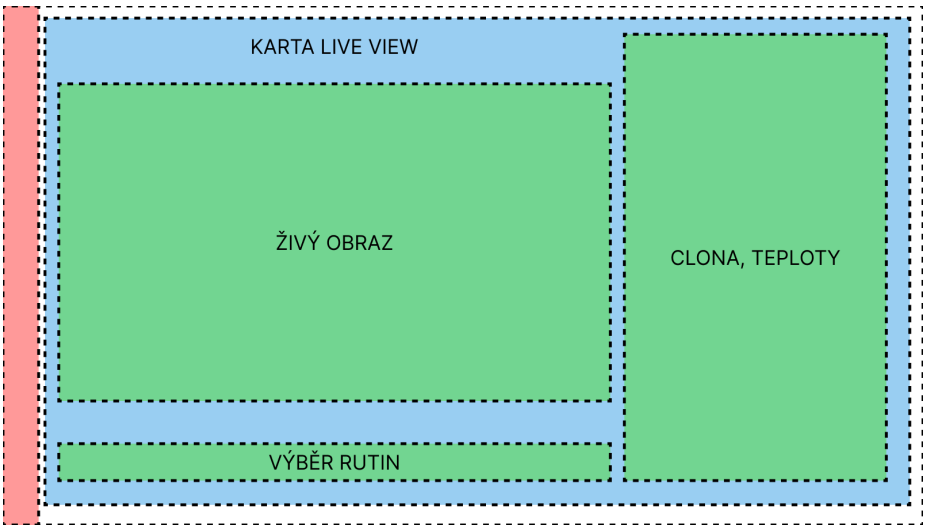
stránka (live view) nebo editace obrazu. Tímto uspořádáním je zajištěno úspěšné připojení na zařízení a intuitivní navigace do dalších částí aplikace. Wireframe stránky je vidět na 3.4.



Obrázek 3.4: Wireframe výběru zařízení a aplikačního módu

Domovská stránka

Domovská stránka aplikace bude sloužit jako hlavní vstupní bod a centrální hub pro sledování živého obrazu z kamery a některých jejich nastavení. Bude se nacházet na routě `/live-view` a poskytovat informace o ovladači clony a jeho aktuálním nastavení, živý přenos kamery s možností přechodu do fullscreen a aktuální informace o teplotách jednotlivých částí zařízení. Dále stránka nabídne navigační panel a tři tlačítka vedoucí na: zobrazení základních informací o zařízení, rutiny prvotního nastavení obrazu a prvotního nastavení celku. Rozložení všech těchto prvků je patrné na obrázku 3.5. Mobilní zobrazení těchto částí je dáno návrhem jednotlivých subkomponent: karta, navigační panel a rozložení stránky (viz obrázek 3.2).



Obrázek 3.5: Wriframe domovské stránky

Stránky nastavovacích rutin

Následující stránky tvoří rutiny pro nastavení vybraných parametrů v předdefinovaném pořadí. Skládají se z velké části z komponent `InitialSetupItem` a `InitialSetupWindow` které udávají průchod nastavením.

Prvotní nastavení zařízení

Stránka `Initial setup` dostupná na routě `/initial-setup` provádí uživatele osmi kroky základní konfigurace. Na stránce bude skryt navigační panel, čímž bude uživatel naveden nejdříve konfiguraci dokončit. Kroky konfigurace budou následující:

1. **Nastavení data a času** (`Time settings`): Uživatel nastaví místní datum, čas a časové pásmo, případně povolí NTP a nakonfiguruje synchronizaci s NTP servery.
2. **Nastavení sítě** (`Network`): Konfigurace výstupní rychlosti UDP a parametrů LAN (IP, MAC, subnet mask, brána), přičemž jistá omezení zabraňují nechtěnému odpojení.
3. **Identifikace výrobce** (`Manufacturer`): Kontrola a případná úprava údajů výrobce (název, stát, adresa, město, PSČ, e-mail, telefon) a základních informací o zařízení (part number, sériové číslo, název).
4. **Snímací profil 0** (`Capturing profile 0`): Konfigurace prvního snímacího profilu.
5. **Snímací profil 1** (`Capturing profile 1`): Konfigurace druhého snímacího profilu.
6. **Editor obrazu** (`Image editor`): Interaktivní úprava snímku s definicí výřezu a kvality obrazu.
7. **Nastavení clony** (`Iris settings`): Podrobná konfigurace clony – režim (UNKNOWN, Manual, Auto), skutečná a požadovaná hodnota, expozice a zesílení.
8. **Náhled výsledků** (`Preview`): Zobrazení finálního snímku pro ověření před přechodem na živý přenos.

Nastavení obrazu

Stránka `Image setup` dostupná na routě `/image-setup` je rozdělena do šesti kroků. Navigační panel na stránce bude viditelný, protože následující kroky konfigurace nejsou pro nastavení zařízení nezbytné:

1. **Konfigurace clony** (`Iris`): Uživatel volí režim (UNKNOWN, Manual, Auto), nastavuje aktuální hodnotu clony, cílovou jasnost a rozsahy expozice a zesílení.
2. **Přehled profilů snímání** (`Capturing profiles`): Informativní krok s upozorněním, že profily je vhodné připravit předem a změny se nemusí projevit okamžitě.
3. **Snímací profil 0** (`Capturing profile 0`): Konfigurace prvního snímacího profilu.
4. **Snímací profil 1** (`Capturing profile 1`): Konfigurace druhého snímacího profilu.
5. **Úprava snímku** (`Set image`): Interaktivní editor snímku, kde lze definovat výřez, kvalitu a další parametry související přímo s obrazem.

6. **Náhled snímku** (`Preview`): Zobrazení náhledu výsledného snímku, které ověřuje finální nastavení před návratem na domovskou stránku.

Průchody všech rutin uzavírá tlačítko „Done“, které přesměruje zpět na domovskou stránku.

Snímací profily

Stránka `Capturing profiles` s route `/capturing-profiles` poskytne uživateli přehled všech dostupných snímacích profilů a nástroje pro jejich správu. V horní části se pomocí `<MainButtonContainer>` zobrazí tlačítka pro přepínání mezi jednotlivými profily (`Profile 0`, `Profile 1`). Po výběru se pod seznamem objeví panel obsahující akční tlačítka:

- **Testing capture**: Zobrazí testovací snímek s vybraným profilem
- **Update profile**: Uloží změny vybraného profilu.
- **Edit profile** / **Cancel Edit**: Přepíná režim úprav, ve kterém je profil možné měnit.
- **Delete profile**: odstraní vybraný profil a automaticky zvolí další.

Ve střední části stránky pak budou zabaleny jednotlivé parametry daného profilu v `<Card>` v režimu editace.

Informace o zařízení

Stránka `Device Info` na adrese `/device-info` bude sloužit k prohlížení a úpravě základních informací o vybraném zařízení. Nabídne k nastavení následující parametry:

- **Device name** (`DeviceName`): Název zařízení.
- **Part number** (`PartNumber`): Číslo dílu zařízení.
- **Serial number** (`SerialNumber`): Sériové číslo zařízení.
- **Manufacturer** (`ManufacturerIdentification.Name`): Název výrobce.
- **Manufacturer information** (`ManufacturerIdentification`): Detailní údaje o výrobci.

Všechna pole budou ve výchozím režimu editace, což umožní okamžité změny a uložení prostřednictvím tlačítek vestavěných v komponentách editorů.

Kamera

Stránka `Camera` na routě `/camera` bude sloužit k detailnímu nastavení parametrů obrazu pomocí komponenty `<LiveImageEditor>`. Uživatel zde může:

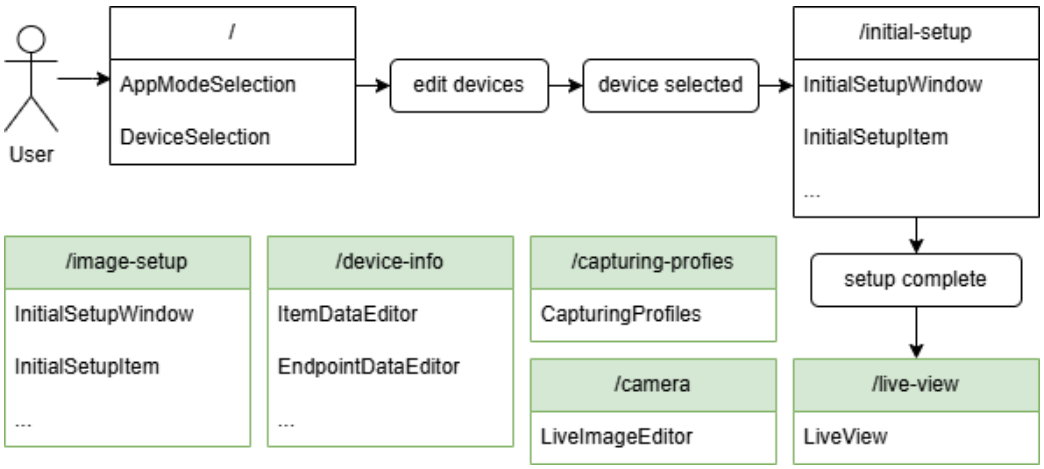
- **Nastavit kompresi** (Slider „Compression“): Ovlivňuje kvalitu snímku.
- **Změnit rozlišení** (Slider „Resolution“): Volba procentuálního zmenšení oproti maximálním rozměrům.
- **Upravit jas** (Slider „Brightness“): Přímá změna požadovaného jasu senzoru.

- **Vybrat oblast zájmu** (Crop): Vstupem do módu ořezávání, potvrdit výběr nebo obnovit původní oblast.
- **Přechod na správu profilů** („Edit“ pod „Capturing profiles“): Navigace na stránku snímacích profilů.

Ve výchozím režimu je editor `editMode="false"` s možností přechodu do módu úprav. Komponenta bude zachovávat stav obrazu (kvalitu, rozlišení i ořez) mezi relacemi pomocí `Blazor sessionStorage`.

3.3.2 Průchod aplikací

Na následujícím obrázku je vidět jeden z možných průchodů aplikací. Uživatel v něm nejdříve upraví a vybere zařízení. Pak se rozhodne pro aplikační mód `/initial-setup` a po dokončení vstupní konfigurace zařízení přejde do `/live-view`, ze kterého se může pohybovat do dalších částí aplikace. Zelená barva značí přítomnost navigačního panelu na stránce, který umožňuje další navigaci v aplikaci.



Obrázek 3.6: Ilustrační průchod aplikací

Kapitola 4

Implementace

Tato kapitola se zabývá výslednou implementací knihovny Blazor komponent[1] a demonstrační Blazor WebAssembly aplikace[2]. Je v ní popsáno využití zvolených technologií, implementované služby, komponenty a stránky. Výsledná aplikace byla implementována pro inteligentní dopravní kameru UC-SCA[3]. Implementace probíhala postupně tvorbou zprvu jednodušších komponent. V případě, že některé komponenty spolu měly komunikovat, byla jim tato funkcionalita později přidána.

4.1 Zvolené tehnologie

Cílem práce bylo vytvořit aplikaci, která bude schopna nastavovat různé typy zařízení bez ohledu na jejich výpočetní zdroje. Z toho důvodu byl jako základ pro vývoj uživatelského rozhraní zvolen framework Blazor ve variantě WebAssembly. Tento přístup zajišťuje, že aplikace bude provozuschopná v běžných webových prohlížečích bez závislosti na konkrétní platformě. Následuje kompletní výčet zvolených technologií, které jsou pro účely práce z implementačního hlediska důležité, a odůvodnění jejich využití:

Komunikace s kamerou

Komunikace s kamerou probíhá přes firemní API společnosti CAMEA založeném na architektonickém stylu REST API, obohaceném o funkce pro správu inteligentních zařízení a případně middleware, který celý přenos dat zabezpečí (záleží na nastavení API). Komunikace s API probíhá formou HTTP požadavků, posílaných klientem na stranu serveru a následně odpovědí serveru s dotazovanými daty. Server v tomto případě tvoří backend, který sbírá informace o zařízeních a zprostředkovává je klientské aplikaci. Požadavky se od standardních REST API požadavků liší v těchto bodech:

- **HTTP POST**: Kromě toho, že je používán pro vytváření položek (create), byl využit pro výměnu kamerových snímků, které nelze zasílat metodou GET. Backend

[1] Více o projektech ASP.NET Core Razor SDK viz https://learn.microsoft.com/en-us/aspnet/core/razor-pages/sdk?view=aspnetcore-6.0.

[2] Více informací o Blazor aplikacích viz kapitola 4.1.

[3] Detailní technická specifikace viz https://www.cameatechnology.com/data/files/UC-SCA-dat-en-20200427.pdf.

na vstupu v těle požadavku očekává JSON strukturu s parametry na požadovaný kamerový snímek[4], který následně vrací v obdobné struktuře v JSON formátu[5].

- **HTTP PUT**: Je používán pro aktualizaci existujících položek (update) a případně pro vytváření zatím neexistující položky.
- **HTTP OPTIONS**: Je používáno pro zjištění metadat (meta). Podpora metadat je vyžadována vždy, kromě rozhraní, která jsou určena jen pro čtení. U rozhraní pouze pro čtení jsou metadata použita pouze v případech, kdy nesou důležitou informaci o povoleném rozsahu hodnoty. Například pokud je třeba indikovat povolený teplotní rozsah a jeho případné překročení.

Zvolený způsob komunikace vychází z doporučení zaměstnanců společnosti CAMEA, jelikož kamera UC-SCA je proprietárním zařízením této firmy a uvedený způsob byl označen za nejvhodnější ze všech dostupných možností.

Blazor

V rámci doplňujícího zadání bylo doporučeno vytvořit knihovnu ve frameworku Blazor.

Vzhledem k potřebě vytvořit client-side aplikaci, která bude fungovat na různých typech zařízení, byla zvolena varianta Blazor WebAssembly. Některá ITS zařízení s nižším výpočetním výkonem (například menší vestavěná zařízení) nemusí být schopná zvládat spouštění Blazor Server aplikace, což by v lepším případě znamenalo značné zpomalení odezvy a doby zpracování dat a v horším případě neschopnost zprovoznění aplikace Blazor typu server na těchto zařízeních.

Multiplatformnost

Blazor WebAssembly poskytuje univerzální běhové prostředí, které umožňuje spouštění aplikací přímo v moderních webových prohlížečích bez ohledu na architekturu procesoru (x86, x64, ARM) nebo operační systém. Mezi podporované webové prohlížeče patří Apple Safari, Google Chrome (a prohlížeče založené na Chromiu), Microsoft Edge a Mozilla Firefox [16]. Tím je zajištěn předpoklad pro podporu zařízení, běžících na různých platformách. Více informací o WebAssembly, jeho výhodách a souvislostech s Blazorem viz 2.2.3 a 2.2.5.

Podpora vývoje a životnost

Pro potřeby práce byl tedy zvolen Blazor v rámci verze .NET 6, která byla v době návrhu stále podporovaná. Vzhledem k tomu, že Long Term Supported verze .NET (.NET 6, .NET 8 a nadcházející .NET 10) jsou podporované minimálně 3 roky a Microsoft jim zaručuje zpětnou kompatibilitu, je možné později přejít na novější Long Term Supported verzi .NET 8, která má být podporována až do listopadu 2026 [6]. Poté lze opět díky zpětné kompatibilitě přejít na verzi .NET 10. Tím lze zaručit dlouhodobou životnost výsledné knihovny.

Radzen

Pro implementaci některých komponent byla využita knihovna Radzen. Radzen je bezplatná open-source knihovna komponent pro Blazor. Obsahuje své vlastní implementace tlačítek,

[4] Více informací o předávané datové struktuře snímků viz 4.12.
[5] Více informací o JSON viz kapitola 2.2.2.

vstupních polí, menu, panelů a dalších komponent. V některých případech bylo vhodné upřednostnit Radzen komponenty před vlastními implementacemi, jelikož jsou již otestované a optimalizované pro použití v Blazor aplikacích. Vývojáři plánují Radzen podporovat i v nadcházejících verzích .NET Blazoru[6].

4.1.1 Použité knihovny a balíčky

Kromě jádra Blazor WebAssembly a vlastních služeb stojí za zmínku následující externí knihovny a balíčky:

- `System.Net.Http`: Klient pro HTTP komunikaci s REST API a JSON serializaci.
- `Microsoft.JSInterop`: Volání JavaScriptových funkcí z Blazoru.
- `Radzen & Radzen.Blazor`: Předpřipravené UI komponenty.

4.2 Implementace knihovny

V rámci knihovny byla implementována sada komponent v `.razor` souborech adresáře `/Components` v projektu knihovny. Většina komponent obsahuje styly, definované izolovanými styly v `.razor.css` souborech umístěných vedle `.razor` souborů v daném adresáři.

4.2.1 Konfigurace

Aby knihovnu bylo možné znovu použít ve více projektech, byla implementována takzvaná kontrola systémového nastavení (KSN). Jedná se o pár: soubor `ApiConfiguration.json`, který definuje komunikaci s daným REST API[7] a službu `ApiService`, která během inicializace tuto konfiguraci nahraje. Pro jednoduchost použití bylo vhodné konfigurační soubor umístit do projektu aplikace, a to ve webovém adresáři `wwwroot/config/`[8]. Konfigurační soubor obsahuje informace o dostupných endpointech a volitelně i informace o zařízeních v následujícím formátu viz 4.1. Cesta k souboru je knihovně předána v metodě `AddLibraryServices` (viz 4.2).

[6] O této skutečnosti se vyjádřil vývojář Radzen týmu na https://forum.radzen.com/t/radzen-blazor-studio-and-net-8/15678.

[7] Více teoretických informací o REST API naleznete obecně zde 2.2.2. Případně v kapitole 4.1 je popsáno konkrétní API použité v demonstrační aplikaci.

[8] Adresář `wwwroot/` je výchozím kořenem pro statické soubory v Blazor (i obecně ASP.NET) aplikacích.

```
{
  "Devices": [
    {
      "Name": "string",          // Název zařízení
      "Address": "string"        // IP adresa a port zařízení
    }
  ],
  "Endpoints": [
    {
      "Name": "string",          // Název endpointu
      "LeafNode": "string",      // Název klíče s daty
      "BasePath": "string",      // Cesta k API
      "DataStructure": {
        "Parameter": "type"      // Např. "PartNumber": "string"
      },
      "Actions": ["string"]      // "Read"/"Write"/"Options"
    }
  ]
}
```

Obrázek 4.1: Struktura konfiguračního souboru

4.2.2 Služby

Služby knihovny jsou do `ServiceCollection` projektu aplikace přidány pomocí třídy `ServiceCollectionExtensions`, která umožňuje komponentám v aplikaci tyto služby injektovat. UML diagram registrace služeb viz 4.3. Služby jsou do `ServiceCollection` přidány v souboru `Program.cs` (v projektu aplikace) takto:

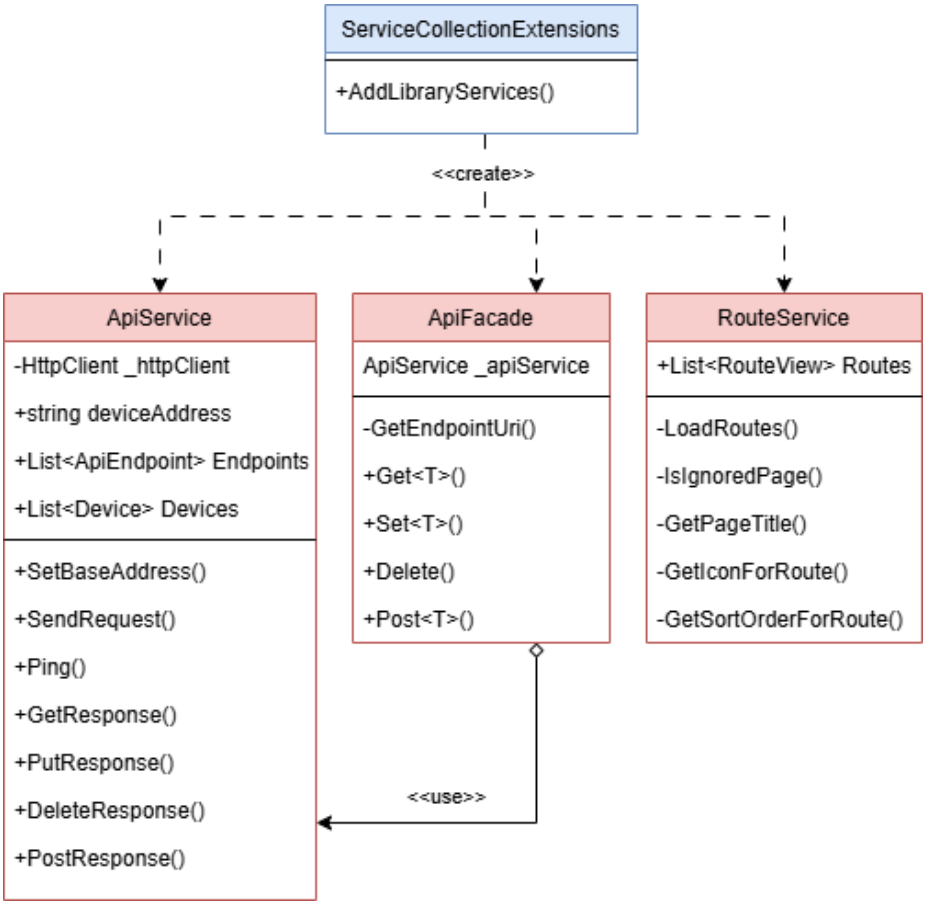
```
var builder = WebAssemblyHostBuilder.CreateDefault(args);
builder.Services.AddLibraryServices(builder.HostEnvironment.BaseAddress);
```

Obrázek 4.2: Přidání služeb do ServiceCollection

Routování

Pro správu navigačního menu byla zavedena služba `RouterService`, která automaticky načte všechny Blazor stránky z assembly daného projektu a vygeneruje pro ně položky menu. Pro některé routy nedávalo smysl, aby byly viditelné přímo z navigačního panelu, a proto bylo možné dané položky ignorovat. U stránek je nyní možné jednoznačně identifikovat routu i její alias (zobrazovaný v navigačním menu) přímo z daného `.razor` souboru stránky. Díky reflexi[9] se nevyžaduje ruční údržba seznamu stránek a nové `.razor` soubory s direktivou (viz 4.4) se do menu automaticky přidají a zároveň se z nich převezme jak URL, tak alias zobrazený v sidebaru. Výsledkem je tedy seznam objektů `RouteView{Url, Name, Icon, SortOrder}`

[9]Informace tom jak reflexe v C# funguje viz https://www.miroslavholec.cz/blog/csharp-reflexe-prakticke-snippety



Obrázek 4.3: UML diagram služeb

v `RouterService.Routes`, již seřazený podle `SortOrder`. Přidání nové stránky proto vyžaduje pouze použití `@page` a volitelně atributu `[Display]` v jejím `.razor` souboru, bez jakékoli další konfigurace.

```
@page "/nějaká-cesta"
@attribute [Display(Name = "Zobrazovaný název")]
```

Obrázek 4.4: Direktivy umožňující automatické routovaní

ApiService

Služba `ApiService` představuje nízkoúrovňovou vrstvu pro veškerou komunikaci s backendem. Její hlavní odpovědností je načtení konfigurace ze souboru `ApiConfiguration.json` obsahujícího seznam zařízení a endpointů. Podle přítomnosti zařízení (`Devices`) v konfiguračním souboru nastaví IP adresu prvního zařízení jako výchozí. Dále služba zprostředkovává konstrukci a odesílání HTTP požadavků (viz ukázka 4.5).

```
... SendRequest(string uri, HttpMethod method, HttpContent? content)
{
var requestUri = new Uri(new Uri(deviceAddress), uri);
var request = new HttpRequestMessage(method, requestUri){
              Content = content };
var response = await _httpClient.SendAsync(request);
...
```

Obrázek 4.5: Konstrukce a odesílání HTTP požadavku

ApiFacade

Na vrcholu `ApiService` stojí `ApiFacade`, která poskytuje vyšší, doménově orientované API pro zbytek aplikace. Podle zadaného `leafNode` (a volitelného indexu) vyhledá odpovídající položku v `ApiConfiguration.json` a sestaví relativní cestu. Služba se dále stará o (de)serializaci přijímaných a odesílaných dat na API. To zajišťují metody `Get<T>()`, `Set<T>()` a `Post<T>()`, které převádějí .NET objekty na JSON a zpět. Jejím hlavním přínosem je, že komponenty, které jí používají, nemusí řešit formát přijímaných dat ani konstrukce URI – stačí pouze znát klíč (leafNode) definovaný v konfiguraci. Použití služby v komponentě k získání hodnoty klíče pak vypadá například takto:

```
@code {
protected override async Task OnInitializedAsync(){
...
var result = await ApiFacade.Get<
  Dictionary<string, JsonElement»(LeafNode, Index);
if (result != null && result.TryGetValue(Item, out var element))
  currentValue = element.ToString();
...
```

Obrázek 4.6: Použití služby ApiFacade

Tímto rozdělením zodpovědností `ApiService` zůstává univerzální a nezávislá na konkrétních datech, zatímco `ApiFacade` definuje „business“ způsob, jak s těmito daty pracovat. Výsledkem je snadno udržovatelná architektura, kde změna v komunikaci se serverem nevyžaduje úpravu obchodní logiky a naopak.

4.2.3 Implementace komponent

Všechny komponenty se nacházejí v souboru `[AnglickyNazevKomponenty].razor` v adresáři `ibtlib/Components` projektu knihovny. Některé komponenty jsou rozděleny do podsložek podle jejich významu v projektu, ale všechny sdílí stejný namespace `ibtlib.Components`. V této kapitole popisuji implementaci většiny z nich. V rámci práce sice bylo vytvořeno komponent více, ale jejich implementace buď není nějak zajímavá, nebo se velmi podobá již popsaným komponentám, a proto je nezmiňuji.

4.2.4 Komunikace dítě-rodič

Vzájemná komunikace mezi komponentami probíhá pomocí kaskádových parametrů, ve kterých nadřazená komponenta (rodič) na sebe vkládá referenci podřazeným komponentám (dětem), které se přes tuto referenci registrovaly v listu dětských komponent v rodiči. Vytváření tohoto vztahu ukazuji na příkladu editoru datových položek, který se registruje do nadřazené skupiny vstupních polí, vypadá takto:

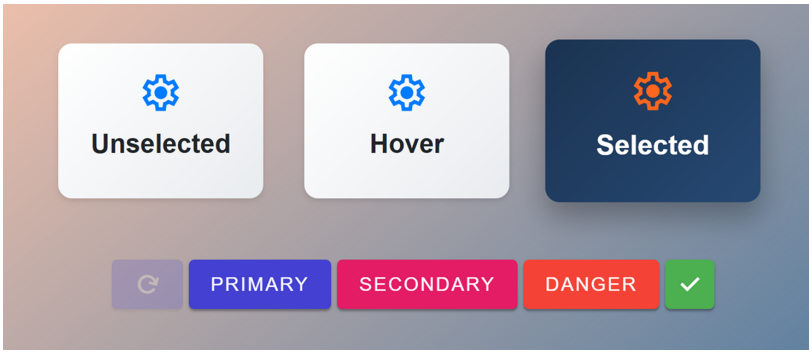
```
<!--ItemDataEditor.razor-->
@code {
[CascadingParameter] public Fieldset? FieldsetParent { get; set;}
...
  void RegisterParents(){
    FieldsetParent?.AddItemDataEditor(this);
  }
}

<!--Fieldset.razor-->
<CascadingValue Value=this>
  @childContent
</CascadingValue>
@code {
private List<ItemDataEditor> itemDataEditors = new();
...
  public void AddItemDataEditor(ItemDataEditor editor){
    if (!itemDataEditors.Contains(editor))
      itemDataEditors.Add(editor);
  }
}
```

Obrázek 4.7: Registrace dětských a rodičovských komponent

Toto řešení umožňuje rodičovským i dětským komponentám vzájemně přistupovat k jejich veřejným metodám a proměnným a vlastnostem.

Základní komponenty

Pro knihovnu byla použita tlačítka z knihovny Radzen (viz 4.1), která podporují editaci jejich stylů, vkládání textu a ikon. Mimo tato tlačítka bylo implementováno tlačítko `MainButton`, které používám v komponentách `InitialSetupWindow`, `DeviceSelection`, `AppModeSelection` a `CapturingProfiles` pro zdůraznění výběru a navigaci v aplikaci. Na ukázce 4.8 můžeme vidět implementovaná tlačítka `MainButton` obalená v kontejnerech `MainButtonContainer` a `MainDialogWindow` nahoře a tlačítka knihovny Radzen dole.



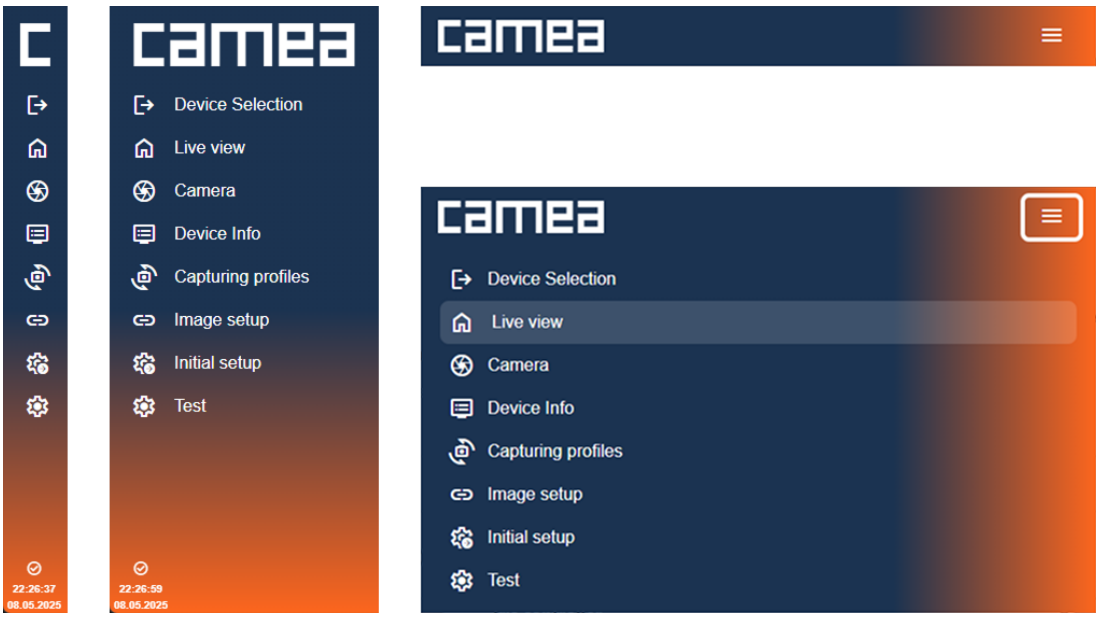
Obrázek 4.8: Použitá tlačítka

Rozložení stránky

Komponenta `PageContent` zajišťuje rozložení obsahu stránky a řídí zobrazování navigačních panelů pomocí parametru `showNavMenu`. V případě rozložení parametrů na velkých obrazovkách komponenta automaticky zarovnává její obsah tak, aby boční navigační panel nepřekrýval, a tím zachovává viditelné množství obsahu na stránkách. Dále definuje parametry `title` (volitelný nadpis stránky) a `showBottomPanel`, který určuje, zda se má zobrazit info na spodní části bočního panelu.

Navigační panel

Navigační panel tvoří komponenty `SideNavMenu` a `TopNavMenu`, které jsou aplikovány na všech stránkách kromě `Index` a `InitialSetup`. Obsah menu komponenta generuje skrz službu `RouterService`, ze které získává informace o možných routách `RouterService.Routes`. Samotný panel obsahuje loga společnosti, odkazy s ikonami a data o systémovém čase a stavu zařízení. Ukázku komponent můžete vidět na obrázku viz 4.9.



Obrázek 4.9: Ukázka komponent navigačního panelu

Editor datových položek

Editor datových položek `ItemDataEditor` je nejčastěji používanou komponentou knihovny pro zadávání vstupních dat. Podle návrhu podporuje zobrazování a editaci několika typů dat (konkrétně `uint`, `int`, `float`, `double`, `bool`, `enum`, `string` a typu `DateTime` ukládaného ve `string`). Zde jsou vypsány některé implementačně zajímavé vstupní parametry komponenty:

Parametr	Povinný	Význam
`LeafNode`	Ano	Identifikátor endpointu, podle kterého se načte JSON struktura.
`Item`	Ne	Klíč v JSON; pokud není zadán, použije se `LeafNode`.
`Name`	Ne	Popisný text pro štítek; výchozí je `Item`, případně název endpointu.
`Index`	Ne	Řetězec pro vícenásobné instance (např. pole); přidává se do URI.
`editMode`	Ne	Režim editace vs. čtení (default `false`).
`showUpdateButton`	Ne	Samostatné tlačítko pro odeslání změny.

Na základě těchto parametrů `LeafNode` (a případně `Item`) určí typ položky `ItemType`, načte aktuální hodnotu přes `ApiFacade.Get<>()`, vykreslí buď formátovaný text (v režimu čtení) nebo odpovídající vstupní ovládací prvek (číslo, checkbox, textbox, dropdown, datepicker) s obsluhou změny a odesláním aktualizace zpět na server. Komponenta dále obsahuje `public` příznak `isLoading`, který udává stav načítání dat. Tento příznak využívají rodičovské komponenty, do nichž je komponenta registrována. Příklad použití viz 4.10.

```
<ItemDataEditor LeafNode="SensorConfig"
                Item="ExpositionTime"
                Index="@Id"
                Name="Exposure time"
                editMode="@editMode"
                showUpdateButton="true"/>
```

Obrázek 4.10: Příklad použití editoru datových položek

Editor dat endpointů

Komponenta `EndpointDataEditor` zprostředkovává hromadné načtení a editaci všech nevnořených klíčů z `DataStructure` jednoho endpointu. Po inicializaci načte datovou strukturu nastaveného `LeafNode` a pro každý klíč v `DataStructure` vytvoří instanci `ItemDataEditor`. Pro zjištění stavu načítání dětských `ItemDataEditor`ů přistupuje k jejich `isLoading` příznaku a podle toho nastavuje vlastní načítání (viz ukázka 4.11). Pokud je povolen parametr `showUpdateButton`, komponenta zobrazí tlačítko `UpdateButton`, které volá metodu `UpdateAll()` na všech dětech. Díky tomu se EndpointDataEditor chová jako obal obsahující jednotlivé editory pro daný endpoint. Možné rozšíření by bylo podporovat i všechny vnořené datové struktury v `DataStructure`.

```
public void CheckLoading(){
  ...
  bool anyChildLoading = ChildList.Any(child => child.isLoading);
  if (isLoading != anyChildLoading){
    isLoading = anyChildLoading;
    if (!isLoading){
      Parent?.CheckLoading();
    }}}
```

Obrázek 4.11: Ukázka načítání editoru dat endpointů

Skupina vstupních polí

Komponenta `Fieldset` má velmi podobnou funkci jako editor dat endpointů, s tím rozdílem, že umožňuje vkládání vlastních komponent typu `EndpointDataEditor` a `ItemDataEditor` a update jejich položek. Podle hodnoty parametru `IsEditable` zobrazuje `Edit/UpdateAll` tlačítka umožňující editaci a aktualizaci dětských položek. Použití komponenty můžete vidět například na stránce prvotního nastavení zde 4.20.

Karta

Komponenta slouží k vizuálnímu oddělení hlavního obsahu stránky od barevného pozadí. Podobně jako `EndpointDataEditor`, `Fieldset` nebo `InitialSetupItem` podporuje komunikaci s jejím obsahem a dokáže tak volat update metody jejich dětských komponent pomocí `CascadingParameter`. Je využívána ve většině stránek a její využití můžete vidět například v 4.19 nebo 4.20.

Prvotní nastavení

Komponenta `InitialSetupWindow` vytváří krokovací rozhraní pro postupné nastavení zařízení. Po inicializaci v `OnAfterRender` načte své registrované potomky typu `InitialSetupItem`, vygeneruje z nich navigační tlačítka a udržuje pořadí i popisy jednotlivých kroků.

Při přepnutí mezi kroky metodou `ChangeItem()` komponenta skryje dosud viditelný `InitialSetupItem`, nastaví nový jako aktivní a aktualizuje záhlaví i stav tlačítek Next, Previous nebo Done. Díky kaskádovým parametrům a metodám `AddInitialSetupItem` každé dítě předává odkaz na sebe (viz 4.2.4), což umožňuje, aby si jednotlivé kroky navzájem sdílely informace o načítání (`isLoading`) a vyvolávaly aktualizace svých komponent přes `UpdateValue()`. Komponenty `InitialSetupWindow` a `InitialSetupItem` využívají sdílený model `ItemInfo`, držící informace o jednotlivých krocích. Definice modelu se nachází v adresáři `/Models` v projektu knihovny.

Obraz

Komponenta `LiveImage` zprostředkovává periodické vykreslování JPEG snímků z kamery do `<canvas>`. Po inicializaci vytvoří `DotNetObjectReference` pro zpětné volání metod z JavaScriptu, načte počáteční parametry snímků z API a spustí `PeriodicTimer` s intervalem `InitialUpdateTime`.

V metodě `RefreshFrame()` posílá přes POST požadavky[10] na endpoint `Frame` s šablonou snímku (obsahující informace o rozměrech, kvalitě, kódování, viz 4.12) a získá base64 data, která předá JS metodě `drawImage()`. Svůj příznak `isLoading` nastavuje pouze do doby, než se načte první snímek. Data samozřejmě načítá i pro každý nový snímek, ale během načítání dalších snímků používá předchozí snímek jako placeholder.

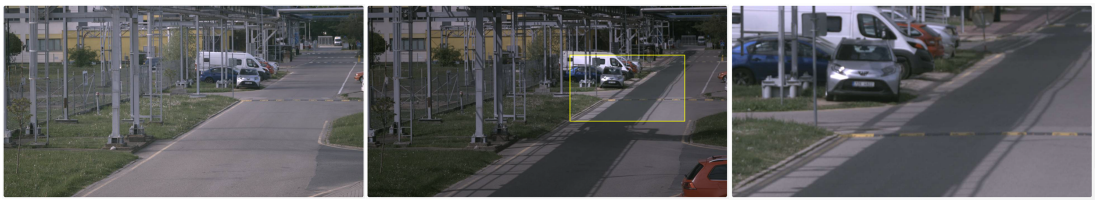
```
{
  "Width":901,
  "Height":477,
  "Encoding":0,
  "Quality":50,
  "AreaOfInterrest":{
    "Top":0,
    "Left":0,
    "Width":0,
    "Height":0}
}
```

Obrázek 4.12: Šablona snímku

Důvodem pro využití JavaScriptového interopu v této komponentě byla potřeba přímé manipulace s elementem `<canvas>`, kterou Blazor samotný nativně nepodporuje. Ta byla potřeba pro dynamické vykreslování obrazových dat a překreslování ořezových rámečků. Metody `SetCropArea` a `NotifyCropSelected` přijímají z JS informace o vybrané oblasti ořezu, přepočítávají souřadnice na originální rozlišení a předávají je zpět do .NET. Při zrušení ořezu `CancelCrop` JS funkce `clearCropBox()` smaže vizuální overlay a komponenta v C# obnoví celý snímek bez ořezu. Komponenta umožňuje přechod do fullscreen pomocí JS funkce `requestFullscreen`.

Editor obrazu

Komponenta `LiveImageEditor` obaluje `LiveImage` a přidává prvky pro nastavení kvality, rozlišení a jasu přes `Slider` komponenty. Na základě změn hodnot sliderů upravuje šablonu snímku v `BuildAndApplyFrameTemplate()` a ukládá ji do `sessionStorage`[11]. Tlačítkem `Crop` volá JS funkce `enableCropping()`/`clearCropBox()` pro výběr oblasti ořezu viz 4.13.



Obrázek 4.13: Ukázka ořezu snímku

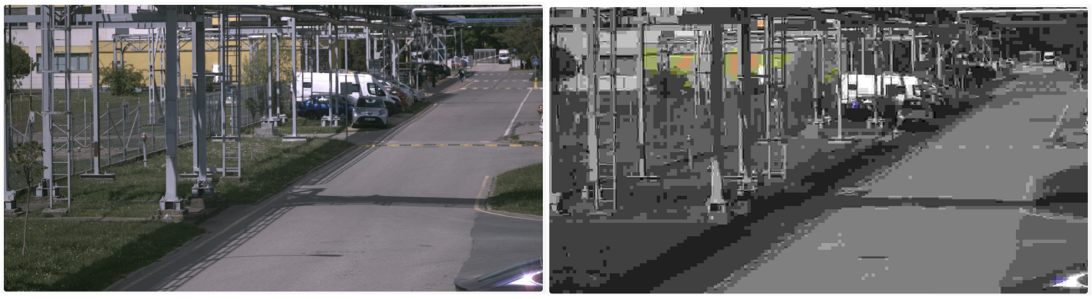
[10] Jedná se o jednu z vlastností, kterou se použité firemní API liší od klasického REST API modelu (viz sekce 4.1).

[11] Více o správě stavu Blazor WebAssembly aplikací viz https://learn.microsoft.com/en-us/aspnet/core/blazor/state-management?view=aspnetcore-6.0&pivots=webassembly

Dále obsahuje slider pro nastavení rozlišení. Ten v daném poměru nastavuje šabloně snímku parametry `Width`, `Height`, přitom zachovává zobrazovanou velikost elementu `<canvas>` a tím efektivně mění rozlišení obrazu. Na obrázku 4.14 můžete vidět efekt změny rozlišení.

Obrázek 4.14: Ukázka změny rozlišení snímku

Slider `Compression` upravuje množství komprese, prováděné na snímku před odesláním, a podle její hodnoty volí odpovídající kompresní algoritmus. V šabloně snímku se však jedná o parametr `Quality` (kvalita). Původně tak byl pojmenován i slider, ale vzhledem k uživatelské zpětné vazbě bylo vhodnější prezentovat obrácenou hodnotu kvality jako kompresi (*komprese* = *max_hodnota_kvality* - *kvalita*), aby se uživateli nepletly dva zdánlivě podobné parametry `Quality` a `Resolution`.



Obrázek 4.15: Ukázka změny komprese snímku

Poslední slider udává takzvanou `DesiredBrightness` neboli požadovaný jas. Tento parametr udává ovladači clony požadovaný jas snímku, kterého by měla inteligentní kamera dosáhnout. Je vhodný v příliš tmavých nebo příliš přesvětlených podmínkách. Efekty tohoto slideru na snímek viz 4.16. Hodnota jasu však může být kamerou automaticky nastavena jinak, podle konkrétního nastavení ovladačů clony.

Obrázek 4.16: Ukázka změny jasu snímku

Výběr aplikačního módu a výběr zařízení

Komponenta `AppModeSelection` funguje jako menu pro přehledné zprostředkování přechodu mezi aplikačními módy. Pro tyto účely komponenta generuje několik `MainButton` komponent a vkládá je do kontejnerů `MainDialogWindow` a `MainButtonContainer`, které tlačítka rovnoměrně zarovnávají.

Komponenta `DeviceSelection` se podobá komponentě výběru aplikačního módu a navíc přidává správu zařízení. Po inicializaci načte seznam zařízení z konfiguračního souboru, zobrazuje je jako tlačítka a umožňuje jejich přidávání, úpravy i mazání přímo v dialogu. Po kliknutí na zařízení provádí ověření připojení (pomocí `ApiService.Ping()`). Pak podle výsledku zobrazuje hlášku v `PopUpPanel`. Na rozdíl od statického menu pro módy má `DeviceSelection` dva režimy zobrazení a editace, kdy se po přepnutí zobrazí vstupní pole pro úpravu názvu, adresy a ikony, a dále poskytuje zpětnou vazbu o všech provedených akcích bez nutnosti opuštění dialogu.

4.3 Implementace aplikace

Projekt v adresáři `/ibt` obsahuje implementaci aplikace. Ta sestává ze souborů, typických pro Blazor WebAssembly aplikaci. Pro přidání knihovny komponent do projektu aplikace stačí:

- Přidat v souboru `Program.cs` služby do `ServiceCollection` (viz 4.2).
- Přidat do souboru `index.html` reference na JS skripty, ikony a vygenerované styly knihovny (viz 4.17).

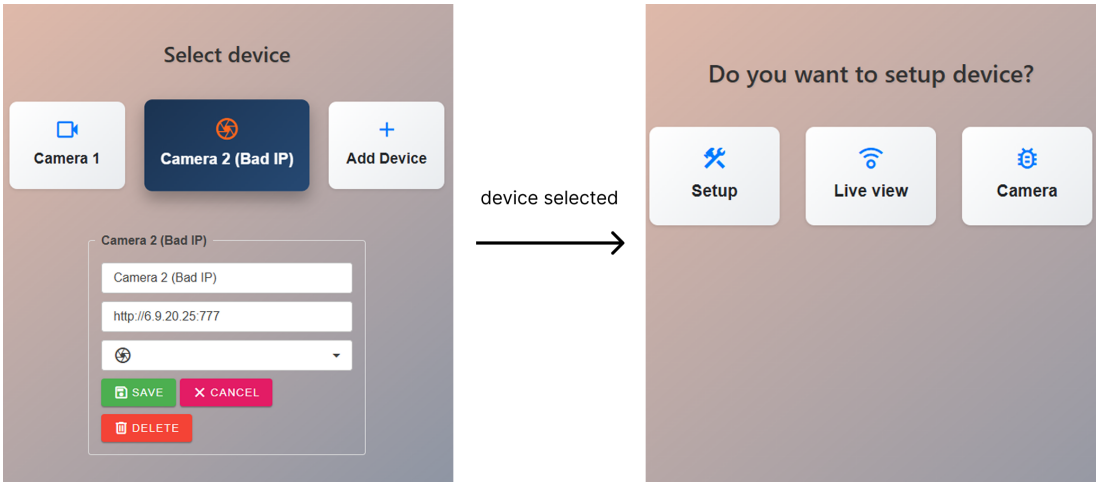
```
...
  <link
  href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined"
  rel="stylesheet»
  <link href="_content/ibtlib/ibtlib.styles.css"rel="stylesheet"/>
</head>
<body>
  <script src="_content/ibtlib/js/scripts.js»</script>
...
```

Obrázek 4.17: Přidání skriptů a stylů knihovny do aplikace

V samotném projektu aplikace jsou pak definovány stránky v adresáři `/Pages`. Stránky obsahují kombinace komponent z knihovny a jsou popsány v následujících sekcích.

Výběr aplikačního módu a zařízení

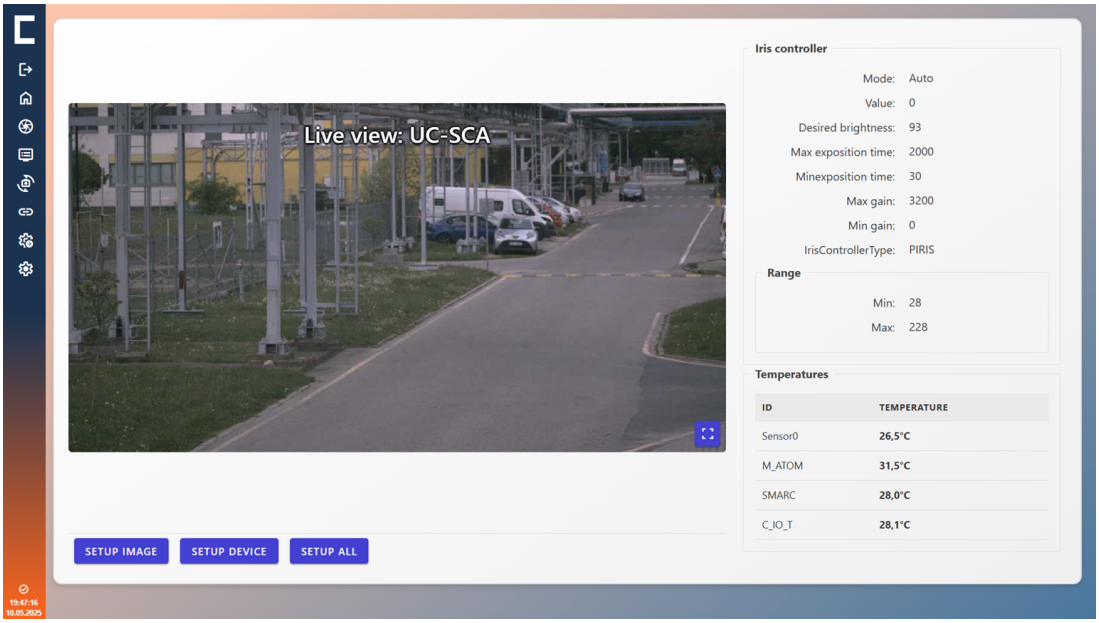
Stránka pro výběr aplikačního módu a zařízení se nachází na URL `/`, případně `/manage-devices` a podle příznaku `deviceIsSelected` přepíná mezi výběrem zařízení a výběrem aplikačního módu. Tyto komponenty jsou obaleny v `PageContent` komponentě se skrytým navigačním panelem a to z důvodu, aby byl uživatel veden, aby se nejdříve připojil ke správnému zařízení.



Obrázek 4.18: Stránka: Výběr zařízení a aplikačního módu

Domovská stránka

Stránka na routě `/live-view` obsahuje jednu komponentu `LiveView` obalenou v `PageContent` s viditelným navigačním panelem, včetně informací nacházejících se ve spodní části bočního panelu (viz obrázek 4.19).



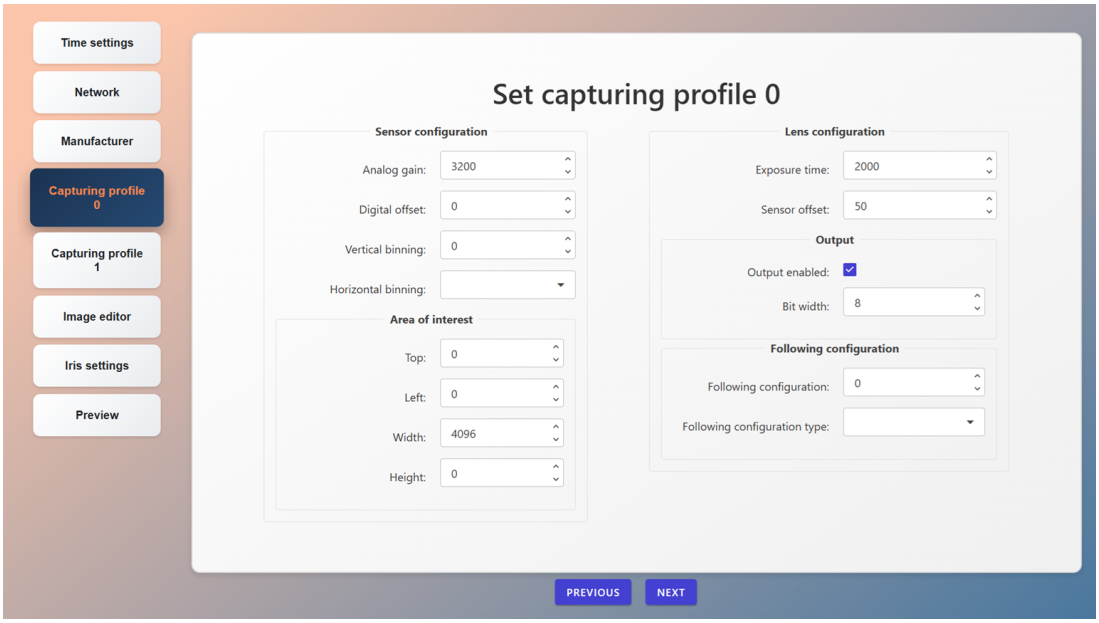
Obrázek 4.19: Stránka: Domovská stránka

Stránky nastavovacích rutin

Pro demonstrační aplikaci byly vytvořeny dvě stránky zajišťující provádění rutinních nastavení parametrů zařízení: stránka prvotního nastavení a stránka nastavení obrazu (na routách `/initial-setup` a `/image-setup`). V průběhu vývoje se ukázalo, že pro nasta-

vení některých základních parametrů zařízení bude účinnější, když všechny tyto parametry budou viditelné pouze na jedné stránce.

Protože účel stránky prvotního nastavení je potřeba provést v jednom celku, nezobrazuji v ní navigační panel. Tím je uživatel veden k dokončení nastavení všech potřebných parametrů. Stránka sestává ze zanořených komponent `PageContent`, `InitialSetupWindow` a `InitialSetupItem`. Dále v jednotlivých `InitialSetupItem` (krocích konfigurace) jsou podle potřeby zanořeny komponenty `Fieldset`, `EndpointDataEditor` a `ItemDataEditor`, zajišťující nastavení položek formulářů.



Obrázek 4.20: Stránka prvotního nastavení zařízení

Dále pak komponenty `CapturingProfile` a `LiveImageEditor`, které obsluhují nastavení zachytávacích profilů a parametrů obrazu. Na obrázcích 4.20 a 4.21 můžeme vidět, jak se komponenty na stránce chovají v normálním a mobilním zobrazení.

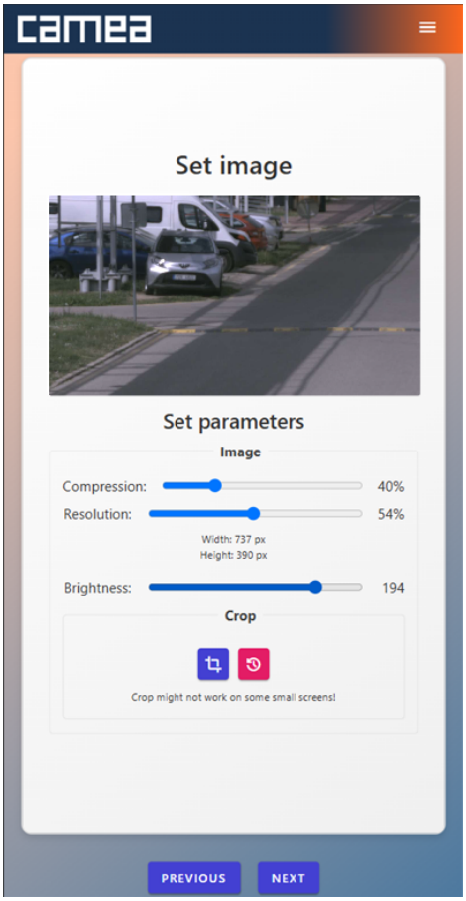
Kamera

Stránka kamery demonstruje společné použití komponent `PageContent`, `LiveImageEditor` a `Card` a slouží k nastavení parametrů souvisejících výhradně s obrazem (viz 4.2.4). Nachází se na routě `/camera` a na rozdíl od stránky prvotního nastavení, umožňuje zobrazení navigačního panelu. Nejedná se totiž o parametry, jejichž nastavení musí být provedeno v určitém pořadí.

Ikony

Jako zdroj ikon byla použita sada Material Icons[12] od společnosti Google, kvůli velkému množství nabízených ikon a proto, že je kompatibilní s Radzen komponentami. Ikony byly vloženy do hlavičky `index.html` souboru v projektu aplikace.

[12]Ikony jsou dostupné z https://fonts.google.com/icons



Obrázek 4.21: Stránka nastavení obrazu v mobilním zobrazení

Barvy

Jelikož demonstrační aplikace bude testována na chytré kameře společnosti CAMEA, také web měl barevně odpovídat hlavnímu barevnému schématu společnosti[13]. Zvoleny tedy byly hlavní kontrastní barvy modrá (`#1b3351`) a oranžová (`#fd661e`) viz dva levé spodní čtverce na obrázku 4.22. Barevná paleta byla následně obohacena o neutrální bílou (`#ffffff`) a tmavě šedou barvu (`#212529`). Tři pravé čtverce pak dále určují barvy interaktivních prvků aplikace a slouží k vizuálnímu nastínění účelu daných prvků, například neutrální modrá (`#4340D2`), zelená pro ukládání (`#4CAF50`) a červená pro mazání, výstrahu (`#F44336`).

Pozadí

Aby pozadí reprezentovalo design společnosti, byly pro základ barev použity hlavní kontrastní barvy (modrá a oranžová). Vzhledem k tomu, že pozadí by nemělo být vizuálně přitažlivější než hlavní obsah aplikace, byla barvám mírně snížena saturace a zvýšen jas. Pro samotné pozadí pak byl použit lineární gradient těchto barev. Pozadí je implementováno ve stylech komponenty `PageContent`. Výsledné barevné pozadí můžete vidět v horní části obrázku 4.22.

[13]Oficiální barevná paleta společnosti CAMEA je patrná na jejich webových stránkách https://www.camea.cz/cz/.

Obrázek 4.22: Základní barevné schéma

Možnosti rozšíření barevné palety

Základní paleta byla v případě potřeby rozšířena o světlejší nebo tmavší odstíny hlavních barev za účelem jemnější typografie, rozlišení stavu komponent (např. aktivní vs. neaktivní) nebo vizuální hierarchie prvků v rámci rozhraní. V aplikaci tedy byly samozřejmě použity i další barvy, gradienty a varianty barev ze schématu, jejichž účel již ale není pro cíle práce významný. Kontrast mezi textem a pozadím byl otestován pomocí nástroje pro ověření WCAG kontrastu a splňuje doporučené hodnoty pro čitelnost na většině zařízení[14].

[14] Testování kontrastu bylo provedeno pomocí nástroje dostupného na http://a11yrocks.com/colorPalette/

Kapitola 5

Testování

Testování probíhalo dvěma způsoby: testování během vývoje a uživatelské testování. Výsledky testování motivovaly několik změn návrhu a implementaci některých nových komponent.

5.1 Testování během vývoje

Část testování probíhala průběžně při testování prototypů komponent a stránek v aplikaci Figma. Závěry tohoto testování se projevily už na samotném výsledku práce. Na základě zpětné vazby byl změněn přístup ke zobrazování a nastavování komprese obrazu (více rozvedeno v sekci 4.2.4). Dále byly na doporučení některých dobrovolníků implementovány komponenty `InitialSetupWindow` a `InitialSetupItem` pro rutiny a jejich krokování.

Vhodným zdrojem zpětné vazby jsou komentáře uživatelů s hlubším porozuměním dané problematiky. V pozdější části vývoje byla aplikace předvedena zaměstnanci společnosti CAMEA, Filipu Kadlčkovi, který měl k aplikaci přínosnou zpětnou vazbu. Líbila se mu spodní část navigačního panelu s informacemi o stavu zařízení a systémovém čase. Dále ocenil nové vizuální zpracování aplikace obohacené o firemní barevnou paletu. Dalším přínosným poznatkem bylo poukázání na skutečnost, že někteří zkušení uživatelé upřednostňují více ovládacích prvků naráz. Z toho důvodu bylo upraveno zadávání parametrů zachytávacích profilů v rutinách (/initial-setup a /image-setup). Ty byly dříve rozděleny do více kroků, které byly pro zkušeného uživatele zbytečně zdlouhavé. Po úpravě se celé nastavení jednoho snímacího profilu provádí v jednom kroku.

5.2 Uživatelské testování

Jedním ze způsobů, jak otestovat GUI aplikace, je uživatelské testování. Jelikož se aplikace zaměřuje na pokročilejší práci s parametry kamery, není úplně vhodné spoléhat se na zpětnou vazbu uživatelů, kteří s danou problematikou nemají žádné zkušenosti. Na druhou stranu zpětná vazba od uživatelů s byť jen menším povědomím o problematice může být přínosná. V takovém případě je vhodné neznalé uživatele poučit o základních principech testované disciplíny. Pro tyto účely byl sestaven stručný návod s vysvětlivkami některých prvků aplikace spolu s teorií potřebnou ke správnému pochopení problematiky nastavování kamer.

Pro účely testování byl sestaven formulář, zaměřený na ergonomičnost a intuivitu aplikace. Formulář byl spolu s aplikací poskytnut dobrovolníkům a zaměstnancům společnosti CAMEA.

Závěry průzkumu pomocí formuláře byly následující. Většina respondentů projevila spokojenost se vzhledem a ergonomií GUI. Výběr a editace zařízení přišly uživatelům snadné a pohyb jednotlivými kroky nastavení rutin spíše snadný, avšak jeden uživatel by preferoval nějakou formu dvojího potvrzení před odesláním hodnoty. Editace a ořez snímků přišly uživatelům přirozené a intuitivní.

5.3 Omezení a možná rozšíření

Během implementace jsem narazil na jedno omezení, které je spojeno s vlastnostmi API. Jelikož podstatou kamer ve firemním API není přehrávání videa s vysokou obnovovací frekvencí (maximální obnovovací frekvence mně přístupné kamery byla uzamčena na 5fps), API nemá implementován žádný protokol podporující spojení a streaming videa. Získávání snímků kamery probíhá pomocí neustálého dotazování na API kamery. Vzhledem k tomu, že data jednoho snímku mohou při maximálním rozlišení (4096x2172) s nulovou kompresí nabrat až 200 KB[1], jedná se o nezanedbatelnou zátěž. Toto se nejvíce projevuje při vykreslování takových snímků, kdy `<canvas>` přirozeně nestíhá tuto operaci stihnout do 0.2 sekundy, kdy má přijít nový snímek. Z tohoto důvodu bylo maximální rozlišení kamery zmenšeno (1365x724).

Aby bylo možné zvýšit rozlišení a ponechat i vyšší obnovovací frekvenci, doporučuji API rozšířit o možnost využití streamovacího protokolu, například MJPEG nebo RTSP, který by umožnil efektivnější přenos obrazových dat bez nutnosti neustálého dotazování.

[1] Velikost obsahu přenášeného JSONu byla zjištěna z měření pomocí aplikace Wireshark viz https://www.wireshark.org/.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat knihovnu komponent, určených pro nastavování a správu inteligentních dopravních systémů, nasadit komponenty do demonstrační aplikace pro správu a monitorování vybraného produktu ITS.

Nejdříve jsem prostudoval problematiku týkající se grafických rozhraní pro nastavování a správu ITS spolu s výzvami spojenými s vývojem ergonomických webových aplikací. Na základě této analýzy jsem stanovil požadavky na nové grafické rozhraní, specifické pro ITS produkty. Podle stanovených požadavků jsem navrhl jednotlivé komponenty knihovny a stránky aplikace, které jsem následně implementoval ve frameworku Blazor.

Volba frameworku Blazor ve variantě Blazor WebAssembly umožnila, že výsledná aplikace je spouštěna na straně klienta, kde je provedena většina náročných výpočtů, souvisejících například se zpracováním obrazu. Aplikaci je dále možno spouštět na prohlížečích Apple Safari, Google Chrome (a prohlížeče založené na Chromiu), Microsoft Edge a Mozilla Firefox. Tím byl zajištěn předpoklad pro podporu zařízení, běžících na různých platformách. Dále byla zvolena Long-Term Supported verze Blazoru .NET 6, která díky zpětné kompatibilitě umožňuje přechod na novější LTS verze .NET 8 a nadcházející .NET 10, čímž byl zajištěn předpoklad pro dlouhou životnost knihovny.

Podařilo se mi podle návrhu implementovat knihovnu komponent a z ní aplikaci pro nastavování kamery UC-SCA společnosti CAMEA. Aplikace dokáže zobrazovat a manipulovat živý obraz kamery. Aplikace umí zobrazit a nastavovat parametry kamery, týkající se nastavení clonových ovladačů, snímacích profilů a základních informací o zařízení, jako například způsoby připojování do lokální sítě nebo nastavení NTP. Nabízí rutiny, ve kterých může uživatel v předem dané sekvenci kroků ergonomicky nastavit velké množství parametrů. Řádově aplikace pracuje s více než 100 parametry kamery.

Vzhledem ke znovupoužitelnosti knihovny je možné aplikaci rozvíjet více způsoby. Jedním z rozšíření, které bych rád přidal, je podpora uživatelských rolí, které již knihovna částečně podporuje. Dalším logickým krokem by bylo umožnit v aplikaci roli administrátora, který by uživatelům přiřazoval různé stupně oprávnění k přístupu ke zdrojům zařízení.

Práce mi dala především hlubší náhled na práci se vzdálenými zařízeními s omezeným výpočetním výkonem. Dále jsem se dozvěděl mnohé o dopravních kamerách a způsobech, jakými operují. Osobně mě práce na knihovně i aplikaci bavila a výsledek práce hodnotím jako zdařilý.

Literatura

[1] *About Figma* [online]. Navštíveno dne 2025-04-05. Dostupné z: https://www.figma.com/about/.

[2] *HikCentral Professional* [online]. Navštíveno dne 2023-12-27. Dostupné z: https://www.hikvision.com/cz/products/software/HikCentral-Professional-series/hikcentral-professional/.

[3] *Intelligent transport systems — Framework for cooperative telematics applications for regulated commercial freight vehicles (TARV) — Part 9: Remote digital tachograph monitoring* [online]. Navštíveno dne 2023-12-26. Dostupné z: https://www.iso.org/obp/ui/en/#iso:std:iso:tr:17465:-1:ed-1:v1:en.

[4] *Intelligent transportation systems: Transforming modern mobility* [online]. Navštíveno dne 2024-11-11. Dostupné z: https://www.iso.org/transport/its-intelligent-transportation-systems.

[5] *JSON - Introduction* [online]. Navštíveno dne 2025-04-19. Dostupné z: https://www.w3schools.com/js/js_json_intro.asp.

[6] *.NET Support Policy* [online]. Navštíveno dne 2025-04-05. Dostupné z: https://dotnet.microsoft.com/en-us/platform/support/policy.

[7] *React A JavaScript library for building user interfaces* [online]. Navštíveno dne 2025-04-16. Dostupné z: https://legacy.reactjs.org/.

[8] *Smart Cities and Internet of Things: Building a Connected World of Tomorrow* [online]. Navštíveno dne 2025-03-22. Dostupné z: https://www.imei.info/cs/news/smart-cities-and-internet-things-building-connected-world-tomorrow/.

[9] *WebAssembly Core Specification* [online]. Navštíveno dne 2025-04-16. Dostupné z: https://www.w3.org/TR/wasm-core-1/.

[10] *WebAssembly Core Specification - Binary format* [online]. Navštíveno dne 2025-04-16. Dostupné z: https://www.w3.org/TR/wasm-core-1/#binary-format.

[11] *What is Angular?* [online]. Navštíveno dne 2025-04-16. Dostupné z: https://angular.dev/overview.

[12] *Colour Psychology* [online]. 2018. Navštíveno dne 2023-12-27. Dostupné z: www.contourheating.co.uk.

[13] *Build fast, responsive sites with Bootstrap* [online]. 2023. Navštíveno dne 2023-04-05. Dostupné z: https://getbootstrap.com/docs/5.3/about/overview/.

[14] Anderson, R. a kol. *ASP.NET Core Blazor* [online]. Navštíveno dne 2025-03-22. Dostupné z: https://learn.microsoft.com/cs-cz/aspnet/core/blazor/?view=aspnetcore-6.0.

[15] Anderson, R. a kol. *ASP.NET Core Razor component lifecycle* [online]. Navštíveno dne 2025-04-16. Dostupné z: https://learn.microsoft.com/cs-cz/aspnet/core/blazor/components/lifecycle?view=aspnetcore-6.0.

[16] Anderson, R. a kol. *Podporované platformy ASP.NET Core Blazor* [online]. Navštíveno dne 2025-04-18. Dostupné z: https://learn.microsoft.com/cs-cz/aspnet/core/blazor/supported-platforms.

[17] Clark, L. *What makes WebAssembly fast?* [online]. Navštíveno dne 2025-04-16. Dostupné z: https://hacks.mozilla.org/2017/02/what-makes-webassembly-fast/.

[18] Fassati, T. *Česko-anglický slovník praktické globální vizuální komunikace*. ČVUT, 2021. 402 s. ISBN 978-80-01-06864-9.

[19] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

[20] Hartinger, D. *První webová aplikace v ASP.NET* [online]. Navštíveno dne 2025-03-22. Dostupné z: https://www.itnetwork.cz/csharp/webforms/asp-dot-net-tutorial-prvni-webova-aplikace.

[21] Horáková, H. *Strategický marketing*. Praha: Grada Publishing, 2003. ISBN 80-247-0447-1.

[22] Janovský, D. *CSS Kaskádové styly* [online]. Navštíveno dne 2024-11-12. Dostupné z: https://www.jakpsatweb.cz/css/.

[23] Janovský, D. *HTML příručka* [online]. Navštíveno dne 2024-11-12. Dostupné z: https://www.jakpsatweb.cz/html/.

[24] Katz, Y. *What is Handlebars?* [online]. Navštíveno dne 2025-03-22. Dostupné z: https://handlebarsjs.com/guide/.

[25] Learned, E. P., Andrews, K. R., Christensen, C. R. a Guth, W. D. *Business Policy: Text and Cases*. Homewood/Ill: Irwin, 1965.

[26] McConnell, J. *WebAssembly support now shipping in all major browsers* [online]. Navštíveno dne 2025-04-16. Dostupné z: https://blog.mozilla.org/en/mozilla/webassembly-in-browsers/.

[27] Moll, C. *Mobile Web Design*. Cameron Moll, 2008. ISBN 978-0615185910.

[28] Nešpor, Z. R. *Metoda srovnávací* [online]. 2023. Navštíveno dne 2023-12-27. Dostupné z: https://encyklopedie.soc.cas.cz/w/Metoda_srovnÃąvacÃŋ.

[29] Sankar, G. *Top Camera Features that Empower Smart Traffic Management Systems* [online]. Navštíveno dne 2024-11-11. Dostupné z: https://www.edge-ai-vision.com/2024/07/top-camera-features-that-empower-smart-traffic-management-systems.

[30] Sriskandaraja, D. *WebAssembly (WASM): The Next Big Thing in Web Development?* [online]. Navštíveno dne 2025-04-16. Dostupné z: https://medium.com/%40sridinu03/webassembly-wasm-the-next-big-thing-in-web-development-be43a5a410f3.

[31] Tattile. *Pathfinder Device Discovery & Management Software* [online]. Navštíveno dne 2024-01-25. Dostupné z: https://www.tattile.com/.

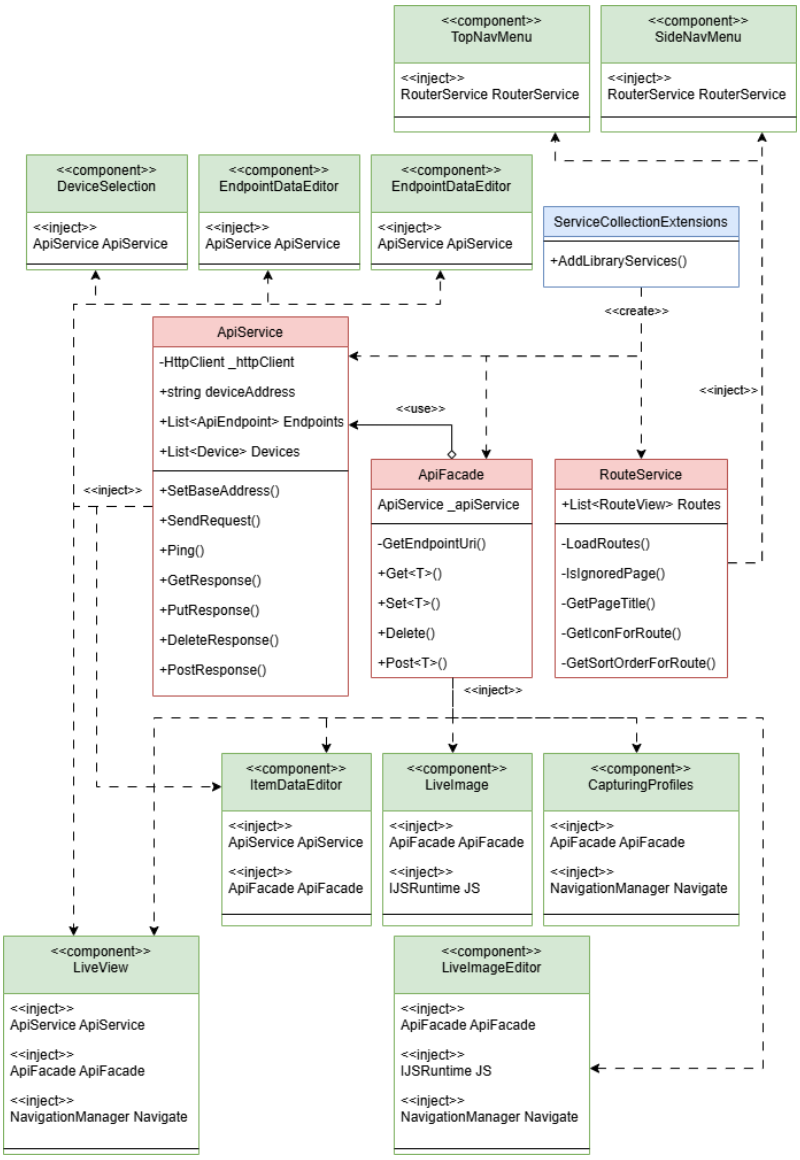
[32] Thirunaukkarasu, R. *Microsoft Blazor vs. Angular vs. React – Why Should You Choose Blazor for Web UI Development?* [online]. Navštíveno dne 2025-04-05. Dostupné z: https://kanini.com/blog/blazor-vs-react-vs-angular/.

[33] Veroukis, M. *Guide to Blazor JavaScript Interop* [online]. Navštíveno dne 2025-03-22. Dostupné z: https://imaginet.com/2021/guide-blazor-javascript-interop/.

[34] Špačková, Z. *Vizualizace uživatelského rozhraní: diplomová práce.* 2018. 91 s. Diplomová práce. Masarykova univerzita, Fakulta pedagogická, Katedra výtvarné výchovy.

Příloha A

Diagram tříd služeb a komponent



Obrázek A.1: Diagram tříd služeb a komponent knihovny

Příloha B

Formulář

B.1 Úvod

Vyplnění formuláře nezabere víc než 5 minut. Formulář je určen k otestování použitelnosti a funkčnosti demonstrační aplikace pro správu inteligentních kamer. Vaše odpovědi mi pomohou odhalit případné problémy v uživatelském rozhraní, zpřehlednit průchod aplikací a vylepšit celkový uživatelský zážitek. Děkuji za váš čas a cennou zpětnou vazbu!

B.2 Otázky

1. **Jsem:**
   - Uživatel amatér (žádné předchozí zkušenosti s nastavováním kamer UC-SCA)
   - Zkušený uživatel (mám předchozí zkušenosti s nastavováním kamer UC-SCA)
2. **Jak snadné bylo najít a vybrat zařízení v seznamu?**
   - 1 – velmi těžké
   - 2
   - 3
   - 4
   - 5 – velmi snadné
3. **Bylo pro vás intuitivní přidat nové zařízení?**
   - Ano
   - Ne
   - Jiné: _____
4. **Objevily se během připojování k zařízení nějaké chybové hlášky, které nebyly srozumitelné?**
   - Ano
   - Ne
   - Jiné: _____

5. **Jak jasné bylo, kam vede každé ze tří tlačítek (Setup, Live view, Camera)?**
   - 1 – vůbec nejasné
   - 2
   - 3
   - 4
   - 5 – zcela jasné
6. **Byl průchod jednotlivými kroky Nastavení obrazu / Síť / Výrobce / ... přehledný?**
   - Ano
   - Ne
   - Jiné: _____
7. **Bylo snadné se pohybovat mezi kroky pomocí tlačítek „Next“ a „Previous“?**
   - 1 – velmi těžké
   - 2
   - 3
   - 4
   - 5 – velmi snadné
8. **Jak hodnotíte přehlednost jednotlivých vstupních polí (číslo, checkbox, dropdown, datum)?**
   - 1 – velmi zmatečné
   - 2
   - 3
   - 4
   - 5 – velmi intuitivní
9. **Líbilo se vám, že můžete upravit hodnotu přímo ve stránce bez dalšího potvrzovacího dialogu?**
   - Ano
   - Ne
   - Jiné: _____

10. Jak na vás působila funkce pro ořez obrazu?

- 1 – velmi zmatečná
- 2
- 3
- 4
- 5 – velmi intuitivní

11. Líbila se vám možnost přejít do celoobrazovkového režimu?

- Ano
- Ne
- Jiné: _____

12. Jak byste zhodnotili celkový design a barevné schéma aplikace?

- 1 – velmi nepřehledné / rušivé
- 2
- 3
- 4
- 5 – velmi přehledné, estetické

13. Setkali jste se s jakýmikoliv chybami nebo nečekaným chováním?

- Obsah odpovědi: _____

B.3 Odpovědi

Otázka	Odpověď	Poznámka
Jak snadné bylo najít a vybrat zařízení v seznamu?	Průměr: 4.8	Všichni respondenti hodnotili velmi pozitivně.
Bylo pro vás intuitivní přidat nové zařízení?	Ano (5x)	
Objevily se chybové hlášky, které nebyly srozumitelné?	Ne (5x)	
Jak jasné bylo, kam vede každé ze tří tlačítek?	Průměr: 4.2	
Byl průchod jednotlivými kroky přehledný?	Ano (5x)	
Bylo snadné se pohybovat pomocí „Next/Previous“?	Průměr: 4.8	
Jak hodnotíte přehlednost vstupních polí?	Průměr: 4.6	
Líbilo se vám upravovat hodnoty přímo bez potvrzování?	Ano (4x), Ne (1x)	Jeden respondent navrhl dialog pro potvrzení.
Jak na vás působila funkce pro ořez obrazu?	Průměr: 4.6	
Líbila se možnost přejít do full-screen režimu?	Ano (5x)	
Jak hodnotíte design a barevné schéma?	Průměr: 4.8	Pozitivní komentáře o barvách a kalendáři.
Setkali jste se s chybami nebo nečekaným chováním?	Ne (4x), částečně (1x)	Jeden komentář o nejasnosti mezi „kamera“ a „live view“.
Typ uživatele	Amatér (5x)	