FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	ETCS - Lektorské PC - Uživatelské rozhraní
Student:	Jiří Sternwald
Vedoucí:	Ing. Jiří Chludil
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Počítačová grafika
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

ETCS (European Train Control System) je jednotný celoevropský vlakový zabezpečovací systém. Cílem této práce je pro simulátor ETCS navrhnout a implementovat prototyp uživatelského rozhraní modulu Lektorské PC.

1. Proveďte rešerši a následně analyzujte současný stav projektu Simulátor ETCS.
2. Na základě konzultací se zadavateli z Fakulty dopravní definujte vhodné funkční a nefunkční požadavky webové aplikace Lektorské PC – zaměřte se především na požadavky na uživatelské rozhraní.
3. Navrhněte uživatelské rozhraní včetně grafické podoby s důrazem na použitelnost.
4. Vytvořte low-fidelity prototyp uživatelského rozhraní.
5. Prototyp podrobte heuristickému testování použitelnosti a uživatelským testům.
6. Na základě výsledků testování implementujte prototyp uživatelského rozhraní webové aplikace Lektorské PC.

Elektronicky schválil/a Ing. Radek Richtr, Ph.D. dne 21. prosince 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

ETCS - Lektorské PC - Uživatelské rozhraní

Jiří Sternwald

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Chludil

11. května 2023

Poděkování

V první řadě bych rád poděkoval vedoucímu této práce, panu Ing. Jiřímu Chludilovi, za jeho čas, cenné rady a přátelský přístup. Dále bych chtěl poděkovat panu doc. Ing. Martinu Lesovi, Ph.D. za zasvěcení do problematiky vlakových zabezpečovačů a možnost zúčastnit se tohoto projektu. Zároveň bych chtěl poděkovat Michalu Mandovi za pomoc s testováním v laboratoři a všem ostatním účastníkům projektu za jejich podporu. Na závěr chci poděkovat své rodině a přátelům, bez jejichž podpory by tato práce nemohla vznikout.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2023 . . . . . .

Odkaz na tuto práci

Sternwald, Jiří. *ETCS - Lektorské PC - Uživatelské rozhraní.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Abstrakt

Práce se zaměřuje na návrh a testování grafického uživatelského rozhraní modulu Lektorské PC pro projekt vlakového simulátoru se zabezpečovacím systémem ETCS. Na začátku jsou popsány základní pojmy a metody návrhu uživatelského rozhraní. V další části se práce zabývá analýzou vlakového zabezpečovacího systému ETCS a současného stavu projektu. Poté se zaměřuje na tvorbu prototypu a následně testováním jeho použitelnosti. Testování bylo provedeno metodou Nielsenovy heuristické evaluace a uživatelským testováním v laboratoři. Výsledkem práce je realizovaný prototyp webové aplikace, která umožní vývojářům i lektorům ovládat a monitorovat stav simulace.

Klíčová slova ETCS, European Train Control System, RBC, Radio Block Centre, GUI, grafické uživatelské rozhraní, React, TypeScript, webová aplikace, frontend

Abstract

The thesis focuses on designing and testing of a graphical user interface of the Lecturer PC module for the train simulator project with the ETCS safety system. At the beginning, the basic concepts and methods for designing user interfaces are introduced. The next part of the thesis focuses on analyzing the ETCS train safety system and the project's current state. Then it focuses on creating a prototype and testing its usability. Testing was carried out using the Nielsen heuristic evaluation method and user testing in a laboratory. The result of this thesis is an implemented prototype of a web application that will allow developers and lecturers to control and monitor the state of the simulation.

Keywords ETCS, European Train Control System, RBC, Radio Block Centre, GUI, graphical user interface, React, TypeScript, web application, frontend

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V současné době vzniká ve spolupráci Fakulty dopravní a Fakulty informačních technologií projekt Simulátor ETCS, který má usnadnit přechod strojvedoucích na nový vlakový zabezpečovací systém. V této práci se zabývám návrhem a implementací prototypu uživatelského rozhraní webové aplikace Lektorské PC, určené pro konfiguraci, ovladání a monitorování stavu simulace. Práce tím navazuje na moduly, které jsme s týmem navrhli v rámci předmětů *Softwarový týmový projekt 1* a *Softwarový týmový projekt 2*. Zároveň se přímo doplňuje s bakalářskou prací Matěje Gorgola [7], která se věnuje implementaci modulu RBC a backendové části webové aplikace Lektorské PC.

V teoretické části uvádím základní pojmy a metody tvorby uživatelského rozhraní. V části analytické vysvětluji, co je to systém ETCS a k čemu má sloužit jeho simulátor. Dále se zabývám analýzou již existujících řešení simulátorů ETCS, analýzou současného stavu našeho projektu Simulátor ETCS a formálními požadavky získaných od zadavatele z Fakulty dopravní.

V další kapitole se na základě zjištěných požadavků věnuji návrhu uživatelského rozhraní a tvorbě low-fidelity prototypu. Vytvořený low-fidelity prototyp dále podrobuji testování heuristickou analýzou a uživatelskému testování použitelnosti v laboratoři. V poslední kapitole se zabývám realizací funkčního prototypu webové aplikace Lektorské PC. Nejprve uvádím použité technologie, dále rozebírám stav implementace prototypu a nakonec uvádím uživatelskou příručku pro jeho zprovoznění.

Kapitola 1

Teorie

Tato část bakalářské práce se zabývá znalostmi a pojmy potřebnými pro návrh a implementaci grafického uživatelského rozhraní.

Nejprve uvádím základní pojmy a jejich definice z oboru tvorby uživatelského rozhraní, dále se zabývám metodami získávání potřebných informací o uživateli a jeho oboru činnosti. Na konci této kapitoly popisuji metodu testování uživatelského rozhraní pomocí heuristické analýzy.

1.1 Základní pojmy a definice

Některé z uvedených pojmů a metod v této části bakalářské práce mají velice širokou škálu užití, a proto se budu v této práci zabývat jen užšími definicemi užitečnými pro návrh grafického uživatelského rozhraní webové aplikace.

- **Human-computer interaction**
  (HCI, česky: interakce člověk-počítač) je výzkumná oblast, která se zabývá přenášením informací mezi člověkem a počítačem.

- **User interface**
  (UI, česky: uživatelské rozhraní) definuje místo, kde dochází k interakcím mezi člověkem a počítačem. Hlavním smyslem designu uživatelského rozhraní je vytvořit uživateli co nejsnažší a nejefektivnejší způsob k dosažení jeho cíle.

- **User experience**
  (UX, česky: uživatelská zkušenost/přívětivost) popisuje celkovou zkušenost uživatele s používáním nějakého produktu, v tomto případě uživatelského rozhraní - jak snadné a přívětivé bylo pro uživatele dosáhnout v uživatelském rozhraní svého cíle.

- **User-centered design**
  (UCD, česky: uživatelsky orientovaný design) je přístup, kterým by se

měli řídit vývojáři uživatelských rozhraní. Umisťuje uživatele do středu pozornosti a zaměřuje se na schopnost uživatele se v rozhraní dobře orientovat.

- **Usability**
  (česky: použitelnost) je atribut kvality, který posuzuje, jak snadno se uživatelské rozhraní používá. Slovem „použitelnost“ se také označují metody pro vytvoření snadno použitelného uživatelského rozhraní.

  Podle Jakoba Nielsena je použitelnost definována následujícími vlastnostmi [9]:

  - **Naučitelnost**
    Jak snadné je pro uživatele dosáhnout svého cíle při prvním kontaktu s designem uživatelského rozhraní.
  - **Efektivita**
    Jak rychle dokaže uživatel dosáhnout svých cílů, jakmile se naučí uživatelské rozhraní používat.
  - **Zapamatovatelnost**
    Jak dlouho uživateli trvá, než začne být znovu efektivní, pokud uživatelské rozhraní dlouho nepoužíval.
  - **Chybovost**
    Jak často uživatelé chybují, jak závažné tyto chyby jsou a jak snadno je mohou napravit.
  - **Příjemnost**
    Jak příjemný (subjektivně) celkový dojem z designu uživatelského rozhraní je.

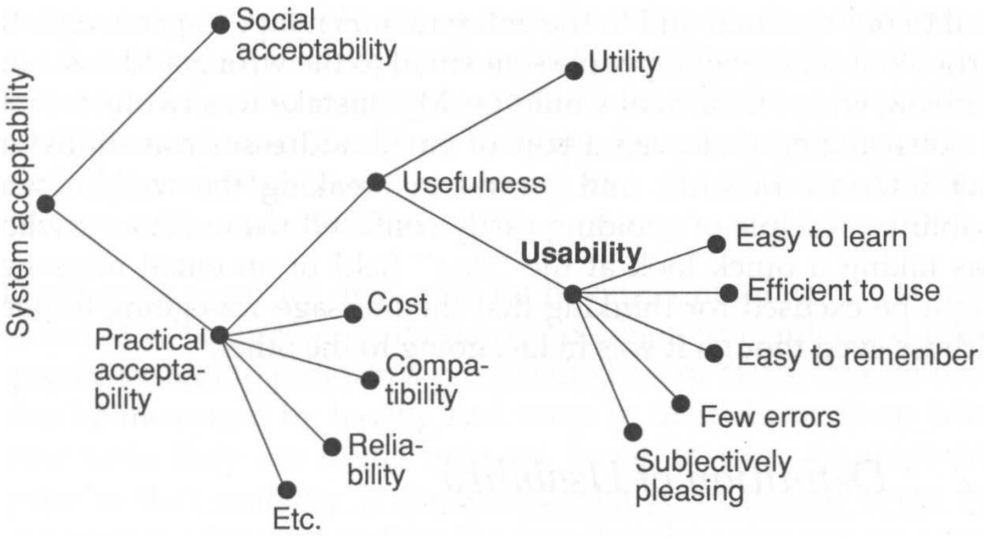
  Použitelnost je jednou z hlavních složek celkové přijatelnosti systému, viz obrázek 1.1.

1.2 Metody zisku informací

Zisk informací a jejich analýza by měla být první částí každého uživatelsky orientovaného designu. Jedná se o soubor metod pro softwarové inženýry k získání požadovaných vlastností a podporovaných činností systému.

Z metod zisku informací v oblasti tvorby uživatelského rozhraní uvádím tři hlavní zástupce: kontextovou analýzu, doménovou analýzu a informační analýzu. Uvedení zástupci nemají přímo definované hranice a informace získané z nich se budou značně překrývat. Smyslem není informace přesně rozdělit do tří kategorií, ale získat všechny potřebné informace o budoucích uživatelích a jejich potřebách.

Na návrh přijatelného systému má vliv spoustu věcí, uvedu zde několik hlavních příkladů [2]:



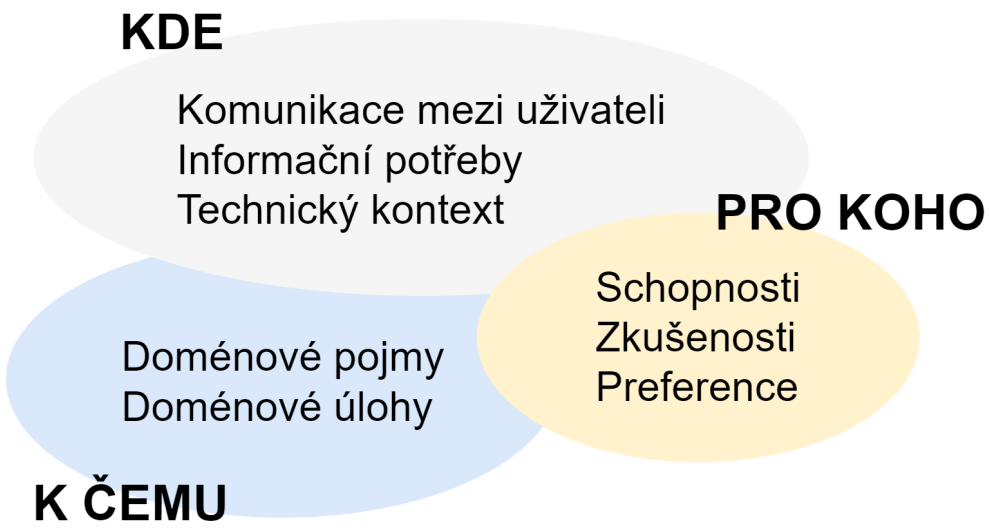
Obrázek 1.1: Jakob Nielsen, Usability Engineering: Model atributů přijatelnosti systému [1]

- **Vlastnosti uživatele**
  Jaké má znalosti, schopnosti, zkušenosti a preference.
- **Obor činnosti**
  Terminologie odvětví a činností, které se v dané oblasti provozují.
- **Prostředí**
  - **Fyzické prostředí**
    Zahrnuje, na jakém zařízení a v jakých podmínkách uživatel pracuje. Potřebuje uživatel například ovládat zařízení pomocí dotykové obrazovky?
  - **Informační potřeby**
    Co uživatel potřebuje vědět a jak mají být případné otázky formulovány.

1.2.1 Kontextová analýza

> *„Každá část kontextové analýzy má svou informačně analytickou část a část práce s uživateli. Bez první fáze nebývá možné se s uživateli domluvit.“* [2]

Kontextová analýza zahrnuje pozorování uživatele v jeho přirozeném prostředí a jeho běžné činnosti. Účelem této analýzy je pochopit fyzický a sociální kontext, ve kterém uživatel bude systém používat.



Obrázek 1.2: Rozdělení kontextové analýzy z předmětu *Tvorba uživatelského rozhraní* [2]

Kontextovou analýzu je možné rozdělit do tří kategorií: pro koho, kde a k čemu, jak je znázorněno na obrázku 1.2. V praxi se ale velmi často „pro koho“ a „k čemu“ překrývají. Informace, pro koho daný artefakt je, se obvykle získávají uživatelským průzkumem. K čemu má artefakt sloužit se zabývá doménová analýza. [2] [9]

1.2.2 Doménová analýza

> *„Doménová analýza vznikla původně v softwarovém inženýrství. Základní myšlenkou bylo, že analýzy pro různé aplikace v témž oboru není dobré dělat odděleně.“* [2]

Doménou je nějaký soubor vědomostí nebo obor lidské činnosti. Například v našem případě je doménou železniční doprava a vlakové zabezpečovací systémy. Doména nemusí být přesně vymezená. Není důležité, jaký pojem do ní patří a jaký ne. Podstatné je, jestli jej uživatelé používají.

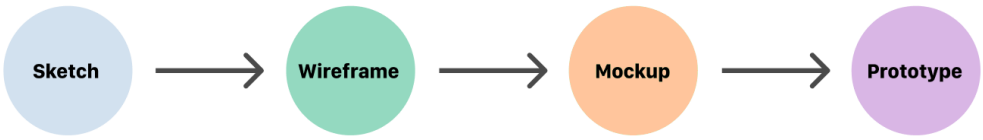
Doménová analýza tedy zahrnuje studování daného odvětví lidské činnosti a prostředí, ve kterém bude výsledný produkt využíván. Je důležité, aby náš vzniklý systém mluvil takovým jazykem, který je pro uživatele jasný a dobře srozumitelný. Cílem doménové analýzy je zjištění používané slovní zásoby a vztahů mezi pojmy. K získání těchto informací se kombinuje informační analýza a práce s uživateli. [2]

1.2.3 Informační analýza

Informační analýza se zaměřuje na studium literatury a standardů z daného odvětví lidské činnosti. Cílem informační analýzy je získat ke zkoumané doméně všechny potřebné pojmy z literatury a pochopit souvislosti mezi nimi. K pochopení získaných informací se informační analýza doplňuje s analýzou doménovou a prací s uživateli. [2]

1.3 Prototypování

Termín „prototyp“ se využívá v mnoha různých kontextech. V kontextu návrhu uživatelského rozhraní se jedná o simulaci vzhledu a vybraných funkcionalit hotového produktu. Díky prototypu tedy můžeme otestovat různé designy dříve, než se vůbec pustíme do implementace. Tvorba věrného prototypu může být časově velmi náročná. Obrázek 1.3 znázorňuje, jaké fáze návrhu by měly předcházet tvorbě prvního prototypu.



Obrázek 1.3: Skica, drátěný model, maketa, prototyp [3]

1.3.1 Věrnost prototypu

Prototypy nutně nemusí vypadat stejně jako hotový produkt. Věrnost (fidelity) se může lišit v oblastech vizuálního designu, obsahu i interaktivity. Právě podle podobnosti prototypu s hotovým produktem rozdělujeme věrnost prototypu na low-fidelity a high-fidelity. [10]

Low-fidelity prototyp je rychlý a jednoduchý způsob převedení high-level design konceptu do snadno testovatelné podoby. Mezi hlavní výhody low-fidelity prototypu oproti high-fidelity prototypu patří například [11]:

- **Tvorba low-fidelity prototypu zabere méně času**
  Tvorba prototypu zabere výrazně méně času. Není potřeba, aby většina položek bylo klikatelných a funkcionalit přesvědčivě reprezentovaných. Díky tomu je k dispozici větší množství času pro designování více stránek a obsahu.

- **Je snažší dělat změny v designu v průběhu testování**
  Prototyp není natolik komplikovaný, aby byl problém na základě testů

cokoliv změnit a testování opakovat. Případně je možné cokoliv změnit i v průběhu jednoho testu.

- **Menší tlak na uživatele**
  Uživatel nepozná, jak dlouho trvala tvorba prototypu. Pokud prototyp nepůsobí kompletní, je pravděpodobnější, že se uživatel upřímně vyjádří, co se mu na designu nelíbí.

1.4 Metody testování

V této bakalářské práci vyžívám dvě metody testování uživatelského rozhraní. Jedna je založena na Nielsenových heuristických kritériích a druhá se zaměřuje na testování na skutečných uživatelích podle pevně definovaných scénářů. Zde, v teoretické části této práce, uvádím metodu testování pomocí heuristické evaluace. Druhou metodou testování uživatelského rozhraní se dále zabývám v kapitole 5.

1.4.1 Heuristická evaluace

Metoda heuristické evaluace k testování uživatelského rozhraní využívá Nielsenovu heuristiku. Jedná se o deset základních vlastností, které by měl splňovat každý interaktivní design. Nejedná se však o návod, jak takový design vytvořit. Každý systém má jiné potřeby a je možné tomu pravidla pro účely testování heuristickou evaluací přizpůsobit (případně zjednodušit). Zde uvádím základních deset heuristik podle Jakoba Nielsena [12]:

1. Viditelnost stavu systému
Design by měl uživatele informovat o svém stavu vhodnou zpětnou vazbou a to v rozumném časovém horizontu.

2. Shoda mezi systémem a reálným světem
Systém by měl mluvit řečí uživatele. Informace by měly být podávány v přirozeném a logickém pořadí.

3. Kontrola uživatele a svoboda
Uživatel často provede akci omylem. Je potřeba přehledně vyznačit „únikový východ“, aby měl uživatel možnost vrátit akci zpět, aniž by musel projít zdlouhavým procesem.

4. Konzistence a standardy
Uživatel by neměl být nucen přemýšlet nad významem různých slov, situací a akcí. Každá funkcionalita by měla být jasně definována a měly by být dodržovány průmyslové standardy.

5. Prevence chyb
Chyby v systému jsou v některých situacích nevyhnutelné. Uživatel by o

nich měl být přehledně a včas informován. Nejlépe by ale mělo být, pokud možno, chybám zamezeno vzniknout. Toho je možné dosáhnout buď chybě vzdornými podmínkami nebo požadováním potvrzení od uživatele u akcí, kde se chyby nacházejí nejčastěji.

6. Přehlednost nad zapamatovatelností
Množství informací, které by si měl uživatel při používání systému pamatovat, by mělo být minimální. A to například při přechodu z jedné části systému na druhou. Informace potřebné k používání systému by měly být kdykoliv přehledně dostupné.

7. Flexibilita a efektivita používání
V systému by měly být zkratky pro pokročilejší uživatele, které umožní urychlit často opakované akce.

8. Estetika a minimální design
Rozhraní by nemělo obsahovat irelevantní ani často nedůležité informace. Každá další zbytečná informace snižuje celkovou přehlednost rozhraní.

9. Pomoc uživatelům s diagnostikou a opravou chyb
Chybové hlášky by měly být psané v jazyce používaném uživatelem, nikoliv v kódu. U každé chyby by mělo být navržené její řešení, pokud to situace dovoluje.

10. Nápověda a dokumentace
V ideálním případě by uživatel neměl potřebovat dodatečné nápovědy k používání systému. Pokud to ale není možné jinak, k systému by měla být dodána dokumentace.

V praxi je potřeba sestavit na základě požadovaných vlastností systém hodnocení. Obvykle provádí testování heuristickou evaluací tři nezávislí experti, kteří prototyp pomocí zadaných heuristik ohodnotí a na základě jejich výstupu se udělají závěry. Tuto metodu využívám k otestování low-fidelity prototypu v kapitole 5.

Kapitola 2

Analýza

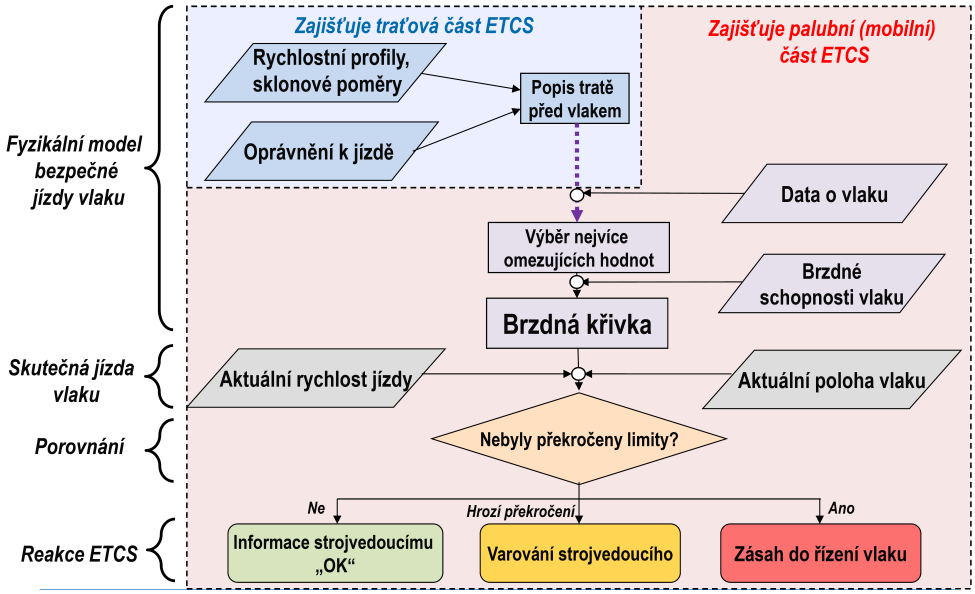
V této kapitole představuji moderní vlakový zabezpečovací systém ETCS a jeho hlavní části. Je zde také uveden slovník základních pojmů potřebných pro pochopení dalších částí této práce. Dále představuji projekt Simulátor ETCS a provádím analýzu již existujících řešení. Poté provádím analýzu současného stavu našeho projektu Simulátor ETCS a jeho modulu RBC, nezbytného pro funkčnost modulu Lektorské PC. Pro modul RBC definuji požadované funkcionality a stručně popisuji stav jejich dokončení.

V poslední části této kapitoly uvádím modul Lektorské PC. Nejprve vysvětluji jeho hlavní účel, analyzuji jeho stav a popisuji použitou architekturu. Nakonec definuji funkční a nefunkční požadavky uživatelského rozhraní modulu Lektorské PC, kterým se dále zabývám v návrhové části této práce.

2.1 Systém ETCS

V současné době se v celé Evropské unii i v České republice zavádí systém ETCS. Jedná se o jednotný celoevropský zabezpečovací systém, který bude zajišťovat lepší bezpečnost železnic a bezproblémové přejíždění vlaků mezi státy. Toto zařízení bude nahrazovat spoustu (mnohdy) zastaralých systémů, které spolu doposud vzájemně nespolupracovaly. Díky tomu se výrazně zvýší bezpečnost na nových, ale i na stávajích tratích a dojde ke zvýšení rychlostních limitů. V České republice se nyní systém instaluje na hlavní tratě a vozidla dopravců se vybavují mobilní částí. Ostrý provoz se plánuje na začátek roku 2025. [4] [13] [14]

Vzhledem k povaze požadovaných funkcí je zabezpečovací systém umístěný částečně na trati a částečně na palubě vlaků. Je tedy možné moduly dělit na traťovou a mobilní část. Na obrázku 2.1 je znázorněno blokové schéma systému ETCS, včetně rozdělení na traťovou a mobilní část.



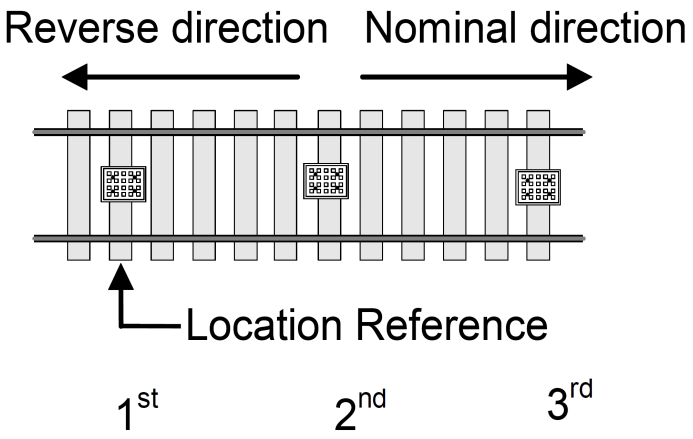
Obrázek 2.1: Blokové schéma systému ETCS [4]

2.1.1 Slovník pojmů ETCS

ETCS (European Train Control System)	Evropský vlakový zabezpečovací systém, který zajišťuje bezpečný pohyb železničních vozidel.
MA (Movement authority)	Povolení od traťové části zabezpečovače pro vlak ujet určitou vzdálenost v rámci omezení infrastruktury.
Eurobalise (Balise)	Eurobalíza (nebo též balíza) je prvek trati umístěný na kolejích. Jedná se o základní prostředek přenosu informací z traťové části na mobilní. Viz obrázek 2.2 a 2.3.
BG (Balise group)	Balízová skupina, která se používá jako reference k určení lokace (a směru jízdy) mezi traťovou a mobilní částí. Viz obrázek 2.2.
LRBG (Last Relevant Balise Group)	Poslední relevantní balízová skupina. Tedy poslední balízová skupina, kterou vlak přejel.
SP (Speed profile)	Profil rychlostních omezení v daném úseku trati.
GP (Gradient profile)	Výškový profil v daném úseku trati.

2.1.2 Části ETCS

V této podkapitole vyberu hlavní zástupce modulů obou částí zabezpečovače a stručně popíšu jejich význam. Vybral jsem hlavně moduly relevantní pro



Obrázek 2.2: Reference pozice a směru vlaku pomocí skupiny tří balíz na kolejích ze subsetu 026-3 [5]

vývoj projektu Simulátor ETCS. Rozmístění mobilní (palubní) a traťové části je vidět na obrázku 2.3.

Moduly nacházející se v traťové části zabezpečovače:

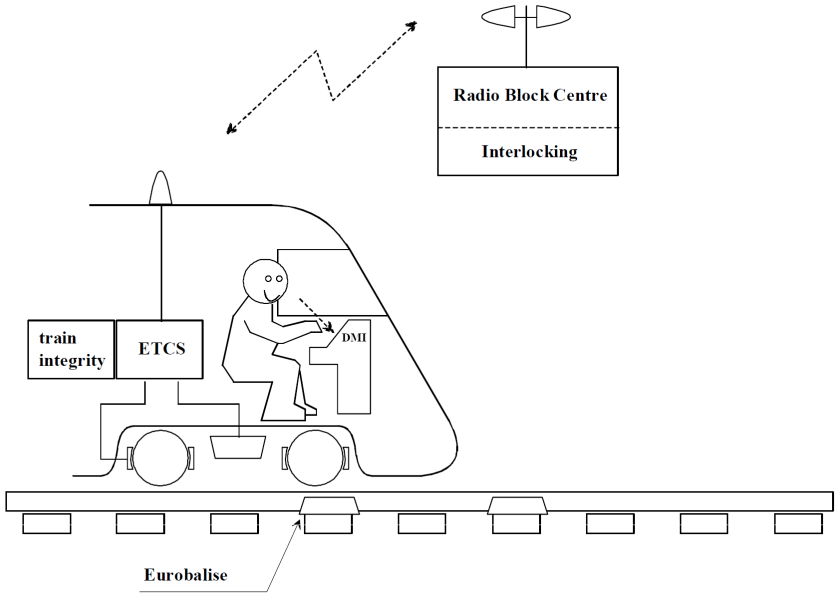
- **RBC** (Radio Block Centre)
  Hlavním cílem RBC je komunikovat s mobilní částí ETCS, monitorovat dění na trati a na základě těchto informací vlakům udělovat oprávnění k jízdě. [5]

Moduly nacházející se v mobilní části zabezpečovače:

- **EVC** (European Vital Computer)
  je hlavní palubní počítač. Jeho účelem je, mimo jiné, zpracování přijatých zpráv z traťové části ETCS, hlášení svého stavu RBC a výpočet brzdných křivek. [5]

- **DMI** (Driver Machine Interface)
  je zobrazovací jednotka, obvykle tvořena dotykovou obrazovkou. Jejím hlavním účelem je informovat strojvedoucího o povolené rychlosti, režimu jízdy a o povolení k jízdě. [5]

- **JRU** (Juridical Recording Unit)
  je záznamová jednotka sloužící k ukládání všech informací z provozu. [5]



Obrázek 2.3: Znázornění komunikace mezi mobilní a traťovou částí zabezpečovače ze subsetu 026-2 [5]

2.2 Projekt Simulátor ETCS

Pro zajištění optimálního přechodu na tento zabezpečovací systém je potřeba vyškolit strojvedoucí v používání systému ETCS na palubě vlaku prostřednictvím simulátoru. Projekt je zaměřen na softwarový návrh komponent simulátoru na PC platformě s reálnými ovládacími prvky, sloužící pro výcvik strojvedoucích k ovládání nového vlakového zabezpečovače ETCS. [13]

Na vývoji projektu spolupracují Fakulta dopravní a Fakulta informačních technoligií ČVUT v Praze. Vývoj simulátoru je rozdělen na několik částí, které odpovídají modulům ETCS a fyzickým prvkům simulátoru. Mezi vyvíjené moduly ETCS patří: RBC, DMI, EVC a JRU. Vývoj modulu EVC se ještě dělí na vývoj simulace chování palubního počítače samotného a jeho interní komponenty k výpočtu brzdných křivek. Dále se vyvíjí modul Vizualizace, který se zaměřuje na generování krajiny a 3D vizualizace jízdy vlaku z pohledu strojvedoucího.

2.3 Přehled současných řešení

Tato kapitola se zaměřuje na analýzu již existujících řešení simulace vlakových zabezpečovačů. V současné době bohužel neexistuje mnoho funkčních řešení

simulace ETCS vhodných pro výcvik budoucích strojvedoucích. Uvádím zde několik příkladů:

- **Dovetail Games:** Train Simulator 2022
- **Aerosoft:** ZUSI 3
- **Clearsy:** Simulateur opérationnel ERTMS/ETCS

Základní verze hry od firmy Dovetail Games zabezpečovací systém ETCS neobsahuje. Pro jeho používání je potřeba doinstalovat komunitní modifikaci. Příklad modelu vlaku pro *Train Simulator 2022* s podporou ETCS od Dominika Chaloupky je dostupný ke stažení zde: `https://k-trains.com/produkt/vectron-rada-193/`.

V následující sekci se zabývám uvedeným řešením od firmy Clearsy, které považuji za nejkompletnější.

2.3.1 Clearsy

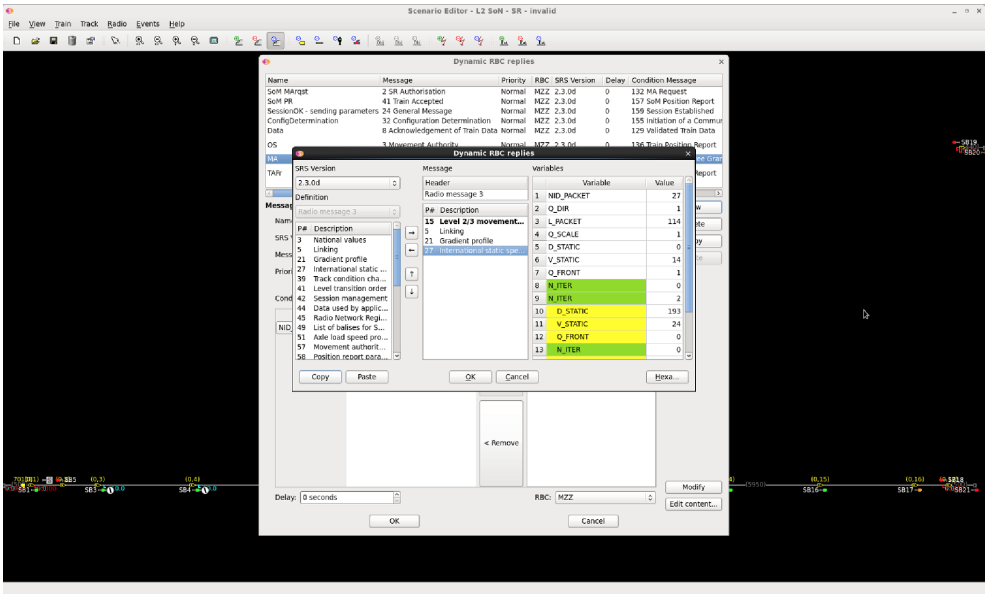
Simulateur opérationnel ERTMS/ETCS od francouzské firmy Clearsy je v dnešní době nejpodrobnější a nejvěrnější simulátor ETCS. Program nabízí velmi detailní editory jak scénářů, tak i jednotlivých zpráv, které se mezi mobilní a traťovou částí ETCS posílají. Ukázka editoru dynamických zpráv od RBC viz obrázek 2.4.

Program používá poměrně malé ikonky bez jasných popisků a kvůli velkému množství nastavení tak ztrácí na přehlednosti. Po spuštění simulace je součástí uživatelského rozhraní jak display strojvedoucího, DMI, tak ovládací prvky strojvedoucího. V horní části nalezneme 2D vizualizaci jízdy. Viz obrázek 2.5.

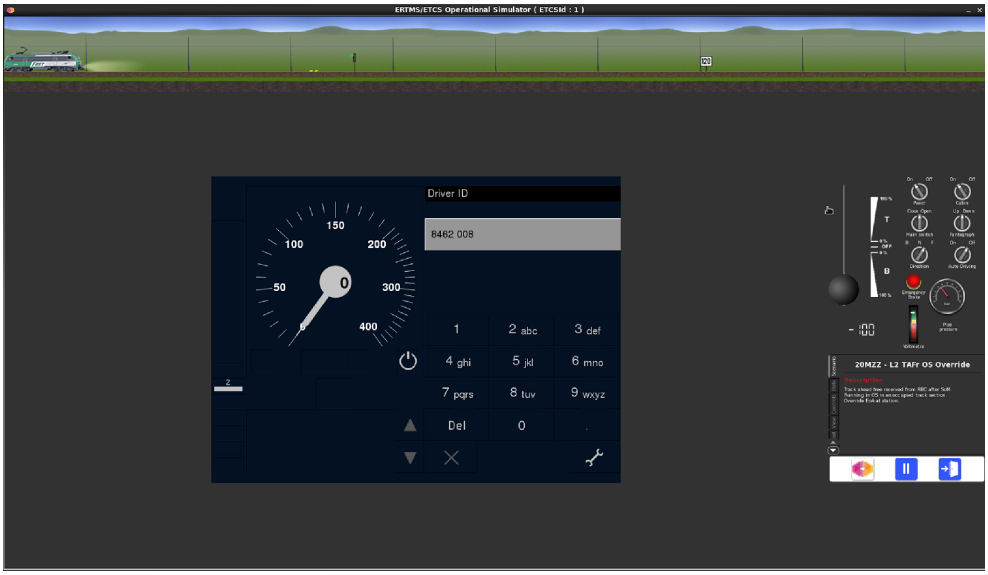
Kvůli nemožnosti spojení s fyzickým pohyblivým simulátorem a absenci 3D vizualizace z pohledu strojvedoucího tento program není vhodným kandidátem pro školení strojvedoucích. Nicméně nám program velmi pomohl k pochopení základních principů vlakových zabezpečovačů a zároveň se pro nás stal nástrojem pro demonstraci požadovaných funkcionalit jednotlivých modulů. Uvedeným nedostatkům uživatelského rozhraní simulátoru se pokouším předejít použitím testovacích metod a heuristik uvedených v kapitole 5.

2.4 Analýza současného stavu projektu

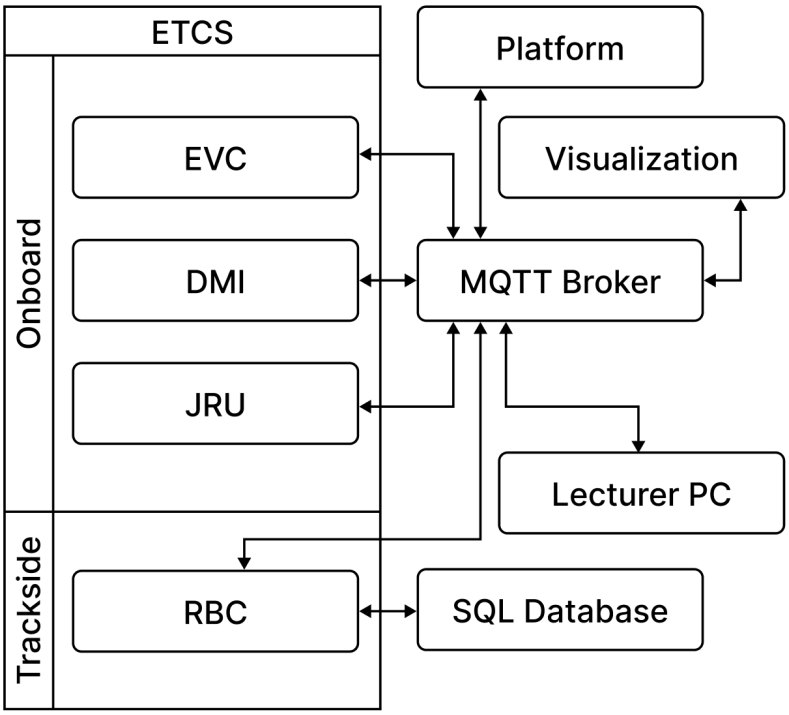
Každý simulovaný modul ETCS běží na vlastním virtuálním počítači. Komunikace mezi moduly probíhá pomocí síťového protokolu MQTT. Síťový protokol umožňuje zveřejňovat a odebírat zprávy na zařízení zvané MQTT broker. Pro implementaci síťového protokolu MQTT v jazyce C++ je v projektu využita open source knihovna Mosquitto [15]. Samostatný běh každého modulu a jejich vzájemná komunikace pomocí síťového protokolu umožňuje



Obrázek 2.4: ERTMS/ETCS Operational Simulator, ukázka editoru zpráv RBC [6]



Obrázek 2.5: ERTMS/ETCS Operational Simulator, ukázka spuštěné simulace [6]

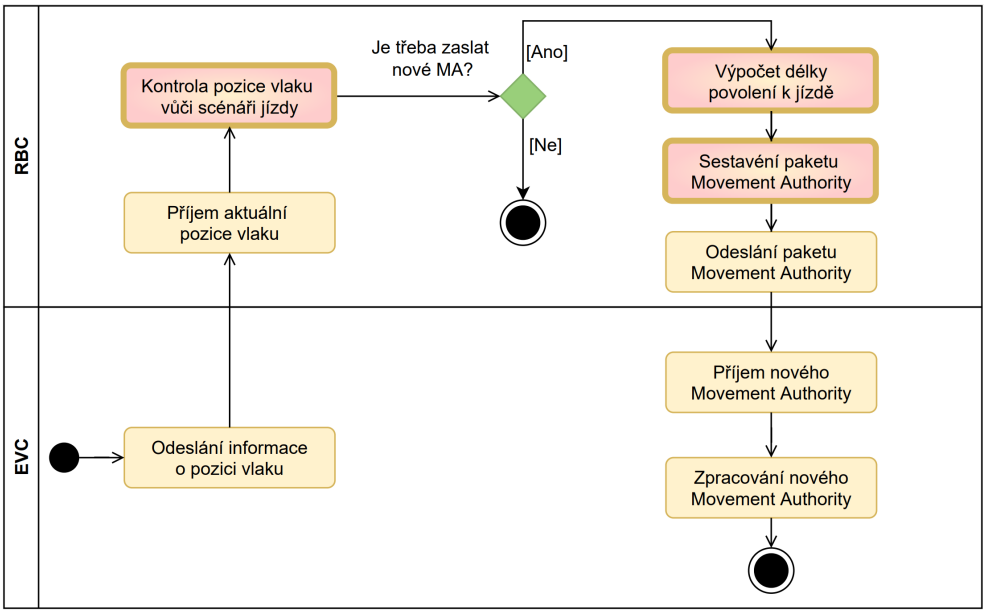


Obrázek 2.6: Diagram komunikace modulů Simulátoru ETCS

věrně simulovat reálnou předlohu systému ETCS. Zprávy posílané na MQTT broker simulují pakety posílané mezi traťovou a mobilní částí ETCS. Podobu paketů jsme navrhovali na základě subsetů dostupných na stránkách Evropské železniční agentury (European Railway Agency, zkráceně ERA) [5]. Schéma komunikace vyvíjených modulů simulátoru viz diagram 2.6.

Projekt se začal vyvíjet během předmětu *Softwarový týmový projekt 1* a následně jsme s vývojem navázali v rámci předmětu *Softwarový týmový projekt 2*. Vzhledem ke komplexnosti a časové náročnosti projektu nebyl na konci těchto předmětů žádný z modulů zcela funkční.

Pro hlavní cíle modulu Lektorské PC, jehož uživatelským rozhraním se zabývá tato práce, je nejdůležitějším modulem RBC. Modul RBC by totiž měl mít veškeré informace o trati i vlacích a díky tomu ovlivňovat průběh jízdy. V praxi se Movement authority vydává na základě informací o volnosti následujícího úseku tratě. Vzhledem k tomu, že v naší aktuální verzi simulace jezdí pouze jeden vlak, modulu RBC by nic nebránilo vydat oprávnění k jízdě na celou trať. K tomuto účelu byl navržen koncept scénáře, který simuluje blokování úseků tratě a jejich postupné uvolňování během jízdy. Scénář je tedy konfigurační soubor, který obsahuje délky volných úseků a v jaký moment se



Obrázek 2.7: Diagram aktivity RBC [7]

má uvolnit další. Takovýto konfigurační soubor je potřeba dodávat modulu RBC a tím řídit průběh celé simulace. Modulům RBC a Lektoské PC se dále věnuji v následujících sekcích této práce.

2.5 Analýza modulu RBC

Pro účely této bakalářské práce pouze stručně popisuji, co bylo potřeba na modulu RBC doimplementovat, aby dávalo smysl s modulem komunikovat pomocí Lektorského PC a měli jsme první funkční spustitelné demo celého simulátoru. Na dokončení modulu RBC se mnou spolupracoval Matěj Gorgol, který se jeho návrhu a implementaci podrobněji věnuje ve své bakalářské práci *ETCS - RBC I* [7].

Na konci předmětu *Softwarový týmový projekt 2* byl modul RBC schopný se připojit k databázi MySQL [16], zpracovat data o trati a přijmout informace o poloze vlaku. Dále uměl vygenerovat některé pakety a sdílet je s modulem EVC prostřednicvím síťového protokolu MQTT. Viz diagram 2.7 [7].

Na modulu chybělo doimplementovat:

- Kontrola pozice vlaku vůči scénáři jízdy [7]
- Dynamické uvolňování volných úseků na základě scénáře
- Výpočet délky povolení k jízdě (Movement Authority) [7]

- Sestavení paketů:
  - Movement Authority
  - Gradient Profile
  - Speed Profile
  - Position Report Parameters (paket 58)
- Logika odesílání zpráv na základě zpráv přijatých z modulu EVC

2.6 Formální požadavky modulu RBC

V této části se budu zabývat formálními požadavky na modul RBC. Tyto požadavky jsme získali spolu s Matějem Gorgolem na schůzkách se zadavateli z Fakulty dopravní. Matěj Gorgol se podrobněji zabývá návrhem a implemetací své části modulu RBC a Lektorského PC v bakalářské práci *ETCS - RBC I* [7].

2.6.1 Funkční požadavky RBC

Funkční požadavky udávají, které funkčnosti bude systém obsahovat. Systém RBC tedy bude splňovat následující:

F01 Komunikace s EVC
Systém bude obsahovat modul komunikace, který bude odesílat a přijímat pakety definované v subsetu 026-7, pomocí síťového protokolu MQTT.

F02 Management vnitřních struktur trati
Systém bude obsahovat databázový modul, který bude umožňovat načítání informací o infrastruktuře trati z databáze. Data budou obsahovat například pozice balízových skupin, výškové gradienty a rychlostní omezení.

F03 Komunikace s lektorským PC
Systém bude obsahovat modul komunikace s lektorským PC, který bude umět odesílat informace o aktuální poloze vlaku a přijímat a zpracovávat příkazy, které přímo ovlivňují povolení o vjezdu. Pro komunikaci bude využívat síťový protokol MQTT.

F04 Načítání informací z konfiguračních souborů
Systém bude umět načítat aktuální informace o dostupnosti trati z konfiguračních souborů. Na základě polohy vlaku a informací v konfiguračních souborech bude systém informován o novém dostupném úseku. Tato funkcionalita simuluje postupné uvolňování trati, tak jak k němu dochází v reálném prostředí. Konfigurační soubory jsou ve formátu JSON.

F05 Vydávání povolení o vjezdu (Movement Authority)
Systém bude schopný na základě informací o dostupnosti trati a o pozici vlaku počítat vzdálenosti a generovat povolení o vjezdu. Každý paket s povolením o vjezdu bude obsahovat informace o výškovém gradientu a rychlostních omezeních platných na danou vzdálenost, tak jak je definováno v subsetech 026-3 a 026-7.

F06 Validace dat
Systém bude ověřovat a zpracovávat všechny přijaté a odchozí pakety, na základě jejich definice v subsetu 026-7. Při zjištění chyby v paketu vypíše chybovou hlášku do konzole a v případě odchozího paketu se pokusí o nové sestavení a následné odeslaní paketu.

F07 Přijímání příkazů z lektorského PC
Systém bude schopen upravovat předem definovaný scénář pomocí příkazů, které obdrží z lektorského PC.

F08 Zobrazení aktuálního stavu systému
Systém bude vypisovat stručné informace o stavu vygenerovaných dat a přijatých a odeslaných paketech do příkazové řádky. Zde se rovněž budou vypisovat chybové hlášky z funkčních požadavků F9 a F10.

F09 Řešení neúspěšného připojení k EVC
Systém bude uživatele informovat o neúspěšném pokusu připojení k EVC. V případě neúspěšného pokusu o připojení se bude s rostoucím časovým odstupem pokoušet o znovupřipojení k EVC. Po dvacátém neúspěšném pokusu o připojení systém vypíše chybové hlášení a ukončí se.

F10 Řešení neúspěšného připojení k databázi
Systém bude uživatele informovat o neúspěšném pokusu připojení k databázi. V případě neúspěšného pokusu o připojení se bude s rostoucím časovým odstupem pokoušet o znovupřipojení k databázi. Po dvacátém neúspěšném pokusu o připojení systém vypíše chybové hlášení a ukončí se.

F11 Samostatný běh
Systém bude po spuštění schopen samostatně obsluhovat vlaky.

2.6.2 Nefunkční požadavky RBC

Nefunkční požadavky určují omezení kladená na systém a mají zásadní dopad na návrh architektury. Pro systém RBC budou platit následující.

N1 Jazyk implementace
Systém bude implementován v jazyce C++ verze 14.

N2 Databáze popisu infrastruktury
Databáze bude využívat software MySQL verze 1.1.9.

N3 Kompatibilita systému
Systém poběží na vzdáleném virtuálním počítači s operačním systémem Debian 10.

N4 Verze subsetů
Implementace bude podporovat více verzí subsetů ETCS, konkrétně verze: 2.3.0d, 3.4.0, 3.6.0.

N5 Formát scénářů jízdy
Scénáře jízdy budou definované pomocí datového formátu JSON. Pro práci s daným formátem se bude využívat knihovna Nlohmann-json3 verze 3.5.0.

N6 Komunikace s moduly
Ke komunikaci s ostatními moduly ETCS bude využit síťový protokol MQTT a knihovna Mosquitto verze 1.5.7.

2.7 Stav modulu RBC

Modul RBC nyní umí zpracovávat dodané konfigurace (scénáře), kontrolovat vůči nim pozici vlaku a sestavovat Movement authority, včetně informací o výškovém a rychlostním profilu. Modul je připravený na zapojení do demo simulace ETCS. Na implementaci jsem spolupracoval s Matějem Gorgolem, který implementoval kontrolování scénáře a výpočet vzdáleností Movement Authority. Já jsem implementoval sestavení paketů a systém uvolňování volných úseků ze scénáře. Pro testování byl vytvořen mock modulu EVC, který periodicky odesílal polohu vlaku a výstup RBC byl kontrolován vůči očekávanému výstupu. Podrobnějšímu testovaní se věnovala Ing. Kateřina Kasalická ve své diplomové práci [17]. Po ukončení naší činnosti na modulu RBC po nás vývoj převzaly další týmy studentů na předmětu *Softwarový týmový projekt*. Jejich úkolem je připravit modul RBC na komunikaci s nově vznikajícím modulem Lektorské PC.

2.8 Analýza modulu Lektorské PC

V této části se zabývám analýzou již vznikající části modulu Lektorské PC. Nejprve vysvětluji jeho hlavní účel. Dále se zaměřuji na aktuální stav funkčnosti a možnostmi dalšího rozšíření. Na konci této sekce se věnuji funkčním a nefunkčním požadavkům uživatelského rozhraní pro tento modul.

2.8.1 Význam modulu

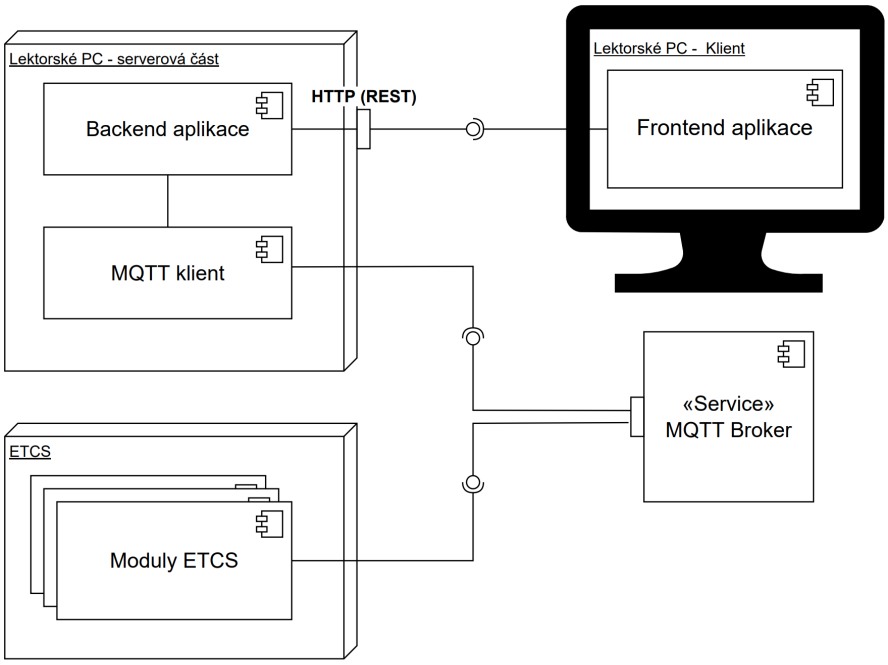
V průběhu předmětů *Softwarový týmový projekt 1* a *Softwarový týmový projekt 2* jsem se podílel na vývoji jednoho z nezávisle vyvíjených modulů Simulátoru ETCS. Jedná se o již zmiňované RBC, EVC, DMI a JRU. Celý průběh simulace je závislý na perfektní součinnosti všech jeho modulů. Ovšem tím, že každý modul je samostatný program ovládaný z příkazové řádky na vlastním virtuálním serveru, je jejich spouštění a následná koordinace náročnou operací. Naše pravidelné testování simulace vyžadovalo pro bezproblémové spuštění zástupce všech modulů, kteří programu předali správné počáteční konfigurace a modul spustili (případně přenastavili a restartovali). Tento proces značně ztěžoval jak testování, tak debugování simulátoru.

Smyslem modulu Lektorké PC je především usnadnění uživateli (nebo případnému vývojáři) spuštění simulátoru, konfigurace modulů a monitorování průběhu jízdy. Sloužit by tedy měl jak vývojářům modulů, tak budoucímu lektorovi bez znalosti implementačních detailů a síťového protokolu, pro snadné a plynulé spuštění. Každý uživatel může mít na systém jiné požadavky, které bude potřeba v návrhu uživatelského rozhraní adresovat. Běžný uživatel (lektor) potřebuje simulátor samostatně spustit, zvolit konfigurace, scénáře a dále monitorovat průběh jízdy. Případný vývojář potřebuje přístup k logům, stavu jednotlivých modulů (například při debugování výpadku spojení) a nástroje pro úpravu konfigurací modulů.

2.8.2 Architektura modulu

Modul Lektorské PC je rozdělen na dvě části: serverovou část a klientskou část. Částí běžící na vzdáleném serveru se zabývá ve své bakalářské práci Matěj Gorgol *ETCS - RBC I* [7]. Serverová část část je napsána hlavně v C++. K volbě tohoto jazyka přispěl fakt, že se jedná o jazyk vyučovaný na Fakultě informačních technologií. Tedy se jedná o jazyk, se kterým by měli mít zkušenosti studenti, kteří budou v budoucnu doplňovat další funkcionality. Architektura modulu je inspirována návrhovým modelem MVC (Model-View-Controller) a ovládací část má na starosti rozhraní REST API. S klientskou částí tedy bude komunikovat pomocí HTTP REST protokolu.

Modul Lektorské PC se zbytkem simulátoru komunikuje pomocí protokolu MQTT, stejně jako ostatní moduly. Viz diagram na obrázku 2.8. V době psání této bakalářské práce ještě není backend modulu Lektorské PC hotov. V současném stavu modul neumí číst zprávy z MQTT brokeru a v jeho implementaci pokračují studenti v rámci předmětu *Softwarový týmový projekt 1.*



Obrázek 2.8: Diagram architektury aplikace [7]

2.9 Formální požadavky modulu Lektorské PC

Po analýze aktuálního stavu Simulátoru ETCS a schůzkách se zadavateli z Fakulty dopravní byly definovány následující funkční a nefunkční požadavky ohledně uživatelského rozhraní modulu Lektorské PC.

Zde uvedené požadavky se částečně překrývají s požadavky v bakalářské práci Matěje Gorgola [7], který se zabývá jejich implementací na serverové části modulu.

2.9.1 Funkční požadavky Lektorského PC

U funkčních požadavků byla provedena prioritizace s ohledem na míru užitečnosti a časové možnosti.

Minimální požadavky

F01 Ovládání stavu simulace
Uživatel bude mít možnost v rozhraní aplikace simulaci kdykoliv spustit, pozastavit, ukončit nebo restartovat.

F02 Zobrazení stavu modulů
Webová aplikace bude schopná v reálném čase zobrazovat stav všech

modulů a v případě ztráty spojení s kterýmkoliv z nich tuto událost nahlásí uživateli.

F03 Zobrazení času simulace
V průběhu simulace se bude na hlavní stránce zobrazovat čas uběhlý od startu.

F04 Zobrazení aktuální rychlosti
V průběhu simulace se bude na hlavní stránce zobrazovat aktuální rychlost lokomotivy.

F05 Zobrazení průběhu scénáře v procentech
V průběhu simulace se bude na hlavní stránce zobrazovat ujetá vzdálenost trati v procentech.

F06 Zobrazení poslední přejeté balízové skupiny
V průběhu simulace se bude na hlavní stránce zobrazovat ID poslední přejeté balízové skupiny a vzdálenosti ujeté od ní.

Optimální požadavky

F07 Volba trasy
Uživatel si před startem simulace bude moct vybrat jednu z předem definovaných tras. Trasa je uložená v SQL databázi propojené s modulem RBC. Obsahuje informace o rozmístění prvků po trati. Dále udává rychlostní omezení a gradientní profily.

F08 Volba scénáře
Uživatel si před startem simulace bude moct vybrat jeden z předem definovaných scénářů. Scénář obsahuje informace o uvolňování úseků tratě v závislosti na času a poloze vlaku.

F09 Počáteční konfigurace modulů a národních hodnot
Při spuštění webové aplikace bude možné zvolit počáteční konfiguraci modulů ETCS včetně národních hodnot.

F10 Nastavení verze subsetů
Uživatel bude moct před spuštěním simulace nastavit používanou verzi subsetů ETCS. Na výběr bude mít verze 2.3.0d, 3.4.0 a 3.6.0.

F11 Vizualizace stavu jízdy
Na hlavní stránce bude v průběhu jízdy v reálném čase zobrazena poloha lokomotivy, balízových skupin a návěstidel. V případě aktivace nouzového brždění bude o této události uživatel informován vykřičníkem nad ikonou lokomotivy.

F12 Zobrazení stavu návěstidel
Součástí vizualizace jízdy bude aktuální stav návěstidel.

F13 Ovládání stavu návěstidel
Uživatel bude mít možnost zasahovat do stavu návěstidel.

Další rozšíření

F14 Editor počátečních konfigurací a národních hodnot
Uživatel bude mít možnost vytvořit vlastní a upravit existující počáteční konfigurace modulů.

F15 Editor scénářů
Uživatel bude mít možnost nadefinovat vlastní scénáře nebo upravit existující. Takto definované scénáře bude možné uložit ve formě konfiguračních souborů.

F16 Zobrazení logů modulů
Uživatel bude mít možnost zobrazit výpisy jednotlivých modulů získaných z MQTT brokeru a záznamy získané modulem JRU.

F17 Zobrazení logů Lektorského PC
Uživatel bude mít možnost zobrazit výpisy informací a chyb Lektorského PC pro účely debugování.

F18 Nastavení jazyka uživatelského rozhraní
Uživatelské rozhraní webové aplikace bude dostupné v českém a anglickém jazyce, s možností rychlého přepnutí.

F19 Vizualizace stavu plošiny
V uživatelském rozhraní bude v průběhu jízdy v reálném čase vizualizována poloha plošiny simulátoru lokomotivy.

F20 Vykreslování brzdných křivek
V uživatelském rozhraní se budou v reálném čase vykreslovat brzdné křivky lokomotivy.

2.9.2 Nefunkční požadavky Lektorského PC

Nefunkční požadavky určují omezení kladená na systém a mají zásadní dopad na návrh uživatelského rozhraní. Pro modul Lektorské PC budou platit následující.

N1 Podpora prohlížečů
Aplikace bude podporovat poslední verze prohlížečů založených na webovém prohlížeči Chromium a jejich mobilních verzí.

N2 Použité technologie
Frontendová část aplikace bude implementována v jazyce Typescript s

využitím frameworku React. Při vývoji budou dodržovány jeho standardy. Na backendovou část modulu Lektorské PC bude zaměřena bakalářská práce Matěje Gorgola s názvem *ETCS - RBC I* [7].

N3 Fungování při zrátě spojení
V případě ztráty spojení s jedním nebo více moduly bude aplikace nadále fungovat a bude možné se znovu pokusit o navázání spojení a případné restartování simulace.

N4 Responzivita
Aplikace bude plně responzivní a bude podporovat jak desktopové, tak mobilní verze prohlížečů. Při vývoji aplikace bude kladen důraz na použitelnost na tabletech a jiných dotykových zařízeních.

N5 Použitelnost
Při tvorbě uživatelského rozhraní bude kladen důraz na ovládání, srozumitelnost a přehlednost pro uživatele z oboru vlakové dopravy.

Kapitola 3

Návrh uživatelského rozhraní

V této části se budu věnovat vydefinování případů užití, analýze budoucích užitelů a jejich rozdělení do dvou hlavních skupin. Ke konci této kapitoly se zaměřím na návrh uživatelského rozhraní na základě zjištěných uživatelských požadavků a jejich případů užití.

Technologie evropského vlakového zabezpečovače (ETCS) je velice rozsáhlá, což je dáno především velkým množstvím variability chování a parametrizace systému. Je proto potřeba před zahájením návrhu uživatelského rozhraní dobře poznat jeho budoucí uživatele a identifikovat všechny možné očekáváné interakce uživatele se systémem.

3.1 Případy užití

Před zahájením návrhu prototypu je potřeba správně definovat všechny případy užití. Případem užití se rozumí, co uživatel od aplikace očekává. Vydefinoval jsem osm hlavních případů užití, ke kterým jsem přiřadil jejich příslušné aktéry v diagramu. Na konci této sekce jsem provedl kontrolu pokrytí všech funkčních požadavků pomocí tabulky.

3.1.1 Diagramy

Smyslem diagramu případů užití (use case diagramu) je ukázat různé způsoby, kterými uživatel může interagovat se systémem. Takový diagram by neměl zacházet do příliš mnoha detailů, ale měl by zachycovat vztahy mezi různými uživateli, systémem a případy užití. Jeden případ užití (use case) může obsahovat více formálních požadavků. Díky tomu dosahuje lepší přehlednosti pro představu, jak může uživatel systém používat.

Pro jednoduchost jsem případy užití rozdělil na dva diagramy: Jízda vlaku a Počáteční konfigurace. Případy užití spadající do prvního diagramu popisují všechny možné interakce uživatele se systémem v době běhu simulace. Před-

	Případy užití							
Požadavky	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
F01	+							
F02		+						
F03		+						
F04		+						
F05		+						
F06		+						
F07					+			
F08					+			
F09						+		
F10								+
F11		+						
F12		+						
F13	+							
F14						+		
F15					+			
F16			+					
F17				+				
F18							+	
F19		+						
F20		+						

Tabulka 3.1: Porovnání případů užití s formálními požadavky

pokládá se tedy, že je systém již adekvátně nakonfigurovaný a strojvedoucí je usazený v simulátoru. Viz obrázek 3.1.

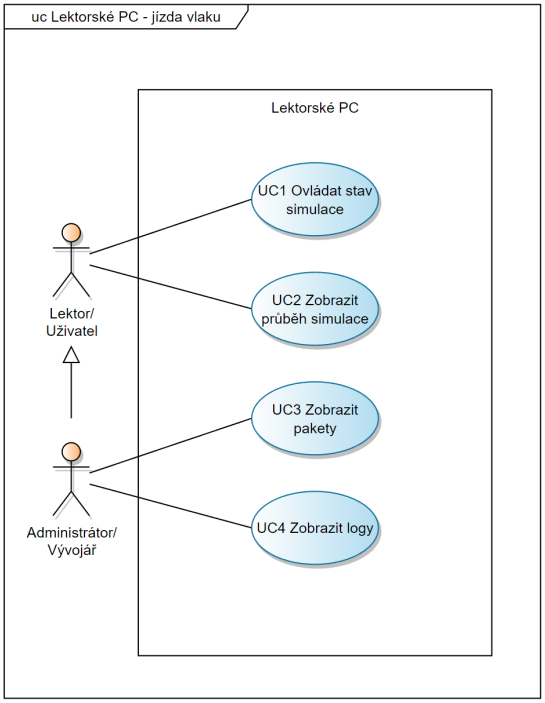
Případy užití spadající do druhého diagramu popisují všechny možné interakce uživatele se systémem v době, kdy simulátor není spuštěný. Zahrnuje tedy všechny formální požadavky, které uživateli umožňují nastavit rozhraní a konfigurovat systém pro příští jízdu. Viz obrázek 3.2.

3.1.2 Kontrola požadavků

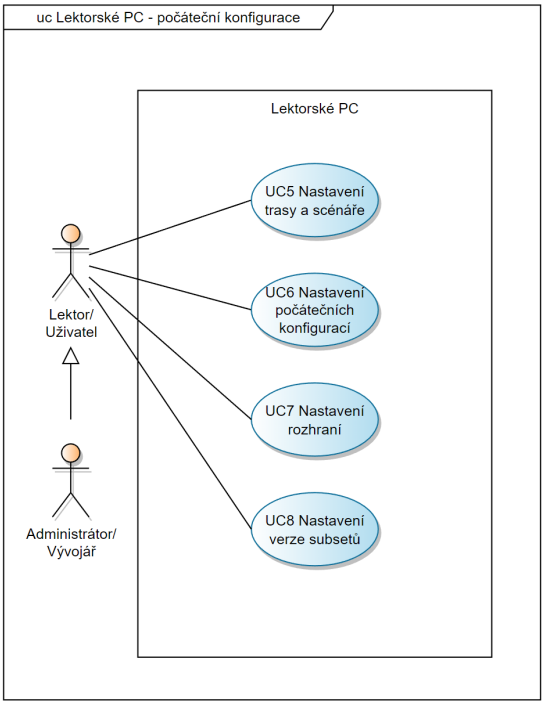
Pro kontrolu, že jsou splněny všechny formální požadavky na uživatelské rozhraní Lektorského PC jsem vypracoval tabulku. Viz tabulka Table 3.1.

3.2 Průzkum uživatelů

Cílem průzkumu uživatelů je zjistit vše o uživatelích, co je potřeba pro použití vyvíjené aplikace a dále tyto informace předat vývojářům takovou formou, aby byli schopni si uživatele a jejich potřeby snadno představit. Důležité je zejména se zamyslet nad tím, jestli je možné uživatele rozdělit do skupin.



Obrázek 3.1: Use case diagram jízdy



Obrázek 3.2: Use case diagram konfigurace

V kontextu simulátoru ETCS dává smysl uživatele rozdělit do dvou skupin podle případů užití: lektora a vývojáře simulátoru.

- **Lektor**
  Má na starost řízení celé simulace. Do toho spadá například výběr scénáře, tratě a následné spuštění všech modulů. Měl by být schopný snadno monitorovat průběh jízdy: znát aktuální rychlost, poslední relevantní balízovou skupinu i procentuální plnění scénáře. Zajímá ho totiž především, jak si zkoušený strojvedoucí vede.

- **Vývojář simulátoru**
  Zajímá jej chod simulátoru a všech jeho modulů. Chce tedy vidět přehledný výpis všech informačních logů, chybových výpisů a znát podrobné informace o běhu simulace, včetně předávaných paketů. Vývojář simulátoru by měl být rychle a přehledně informovaný o každém selhání, jako je například selhání komunikace s konkrétním modulem.

3.2.1 Persony

Persony jsou jedním z možných výstupů průzkumu uživatelů. Tato prezentace výsledků průzkumu je určena především pro kolegy ve vývojářském týmu, návrháře uživatelského rozhraní a jako zpětná vazba pro zákazníka - co jsme se dozvěděli. Fotografie lidí použité pro persony jsem vygeneroval pomocí online nástroje *This Person Does Not Exist* [8].

Karel

Muž ve věku 20–25 let, student oboru softwarové inženýrství, Fakulty informačních technologií ČVUT. Spolupracuje na vývoji a testování simulátoru evropského zabezpečovacího systému železnic. Je součástí týmu, který se zaměřuje na vývoj modulu RBC. Jeho specializací je programování v C++ a komunikace s ostatními moduly pomocí síťového protokolu MQTT. Soustředí se na předávání vlaku informací o trati. Má dobrou znalost implementačních detailů Simulátoru ETCS a jeho modulů.

Obrázek 3.3: Persona Karel [8]

Pro účely debugování potřebuje přehledně zobrazit informace o stavu všech modulů a zaznamenané logy. V rámci simulace potřebuje sledovat interakce

vlaku s infrastrukturou, zprávy o změnách levelu ETCS, změny režimu ETCS, zprávy mezi RBC a EVC a chybová hlášení balízových skupin.

Magdaléna

Žena ve věku 30–35 let, doktorandka Fakulty dopravní, ČVUT. Zabývá se moderními způsoby zabezpečení jízdy železničních vozidel a má velmi dobrou znalost fungování systému ETCS. Nezná ale implementační detaily Simulátoru ETCS a jeho modulů.

Obrázek 3.4: Persona Magdaléna [8]

Chtěla by zaučovat studenty a budoucí strojvedoucí do ovládání nového zabezpečovacího systému, k čemuž potřebuje snadný přístup ke konfiguraci počátečních podmínek, zvolení trati, scénáře a spuštění celého simulátoru.

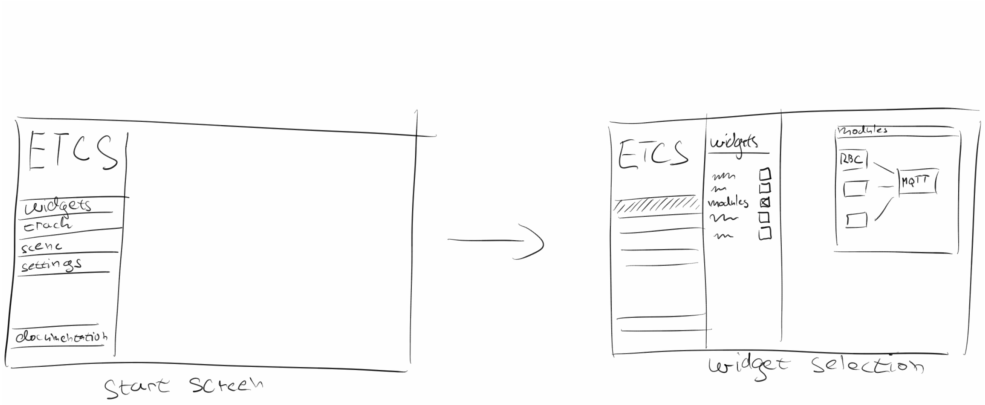
Nezajímají ji debugovací nástroje a záznamy z komunikace modulů. Ocení však přehled o průběhu simulace a vizualizace stavu plošiny.

3.3 Náčrt

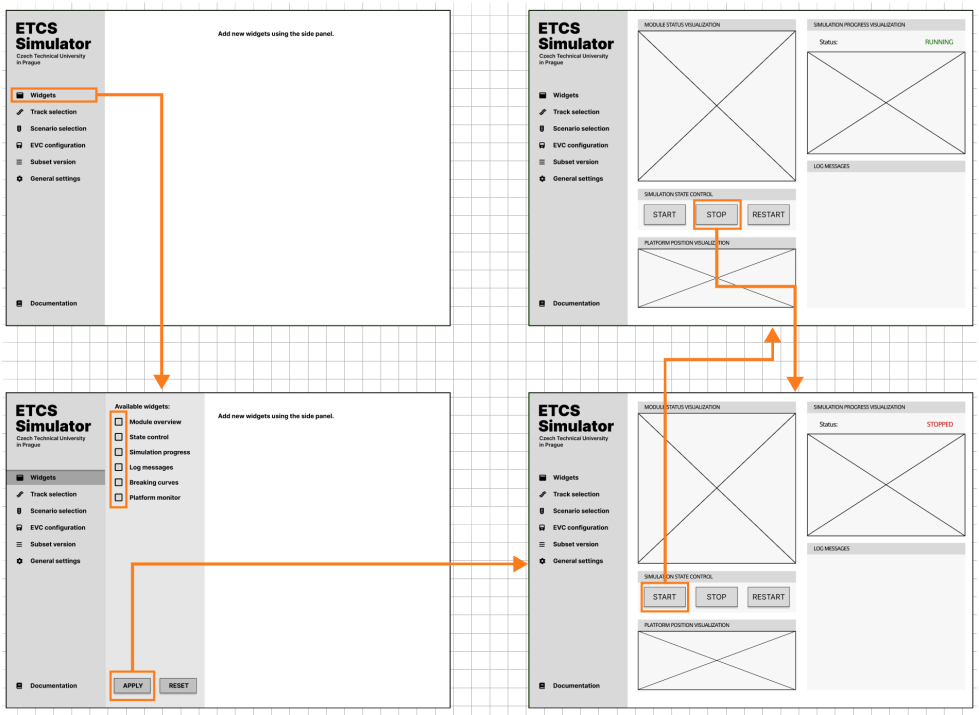
Návrh uživatelského rozhraní jsem začal náčrtem, který velmi obecně zachycuje základní obrazovku s menu v podobě postranního panelu a okno s přehledem aktivních modulů. Viz obrázek 3.5.

3.4 Drátěný model

Drátěný model (angl. wireframe) je výrazně přesnější než skica. Byl vyvinut pomocí webového nástroje Figma [18]. Zachycuje umístění nabídek, tlačítek a pomocí šipek vizualizuje přechody mezi nimi. Viz obrázek 3.6.



Obrázek 3.5: Skica uživatelského rozhraní Lektorského PC



Obrázek 3.6: Drátěný model uživatelského rozhraní Lektorského PC

Kapitola 4

Návrh prototypu

V této kapitole se věnuji návrhu prvního testovatelného prototypu uživatelského rozhraní Lektorského PC. V první části kapitoly vysvětluji, jak jsem interpretoval požadavky zadavatelů z Fakulty dopravní a jak jsem se je rozhodl zakomponovat do designu výsledné aplikace. V další části této kapitoly popisuji mnou vytvořený low-fidelity prototyp. Na konci ukazuji jednotlivé komponenty navrženého designu a uvádím z něj ukázky.

4.1 Interpretace požadavků

Při návrhu uživatelského rozhraní na základě funkčních požadavků je potřeba se zabývat smyslem celého systému, cíli jeho uživatelů a jakým způsobem se bude v budoucnu rozšiřovat.

Pro uvedené účely jsem určil tři hlavní body návrhu uživatelského rozhraní aplikace:

- **Použitelnost**
  Aplikace by měla být použitelná na různých typech zařízení, především na PC a na tabletu. Je tedy potřeba dbát důraz na dobrou ovladatelnost s pomocí myši i dotykových zařízení.

- **Jednoduchost**
  Aplikace by neměla uživatele nijak zdržovat. Pokud byl již systém jednou nastavený, uživatelské rozhraní by mělo uživateli umožnit celý systém spustit během pár kliků.

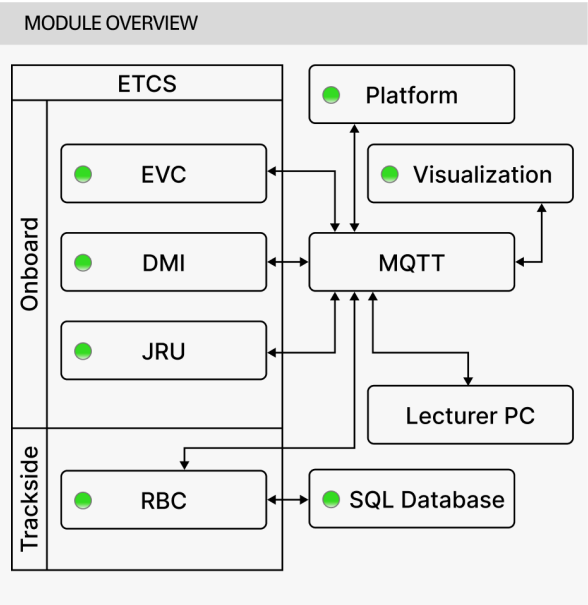
- **Rozšiřitelnost**
  Vzhledem k tomu, že na projektu Simulátor ETCS stále pracují studenti ČVUT, je potřeba brát v potaz přidávání dalších funkcionalit v budoucnu. Uživatelské rozhraní by tedy mělo klást důraz na rozšiřitelnost.

Na základě těchto bodů jsem navrhl systém widgetů. Jeden widget může reprezentovat jednu často opakovanou akci nebo monitorovací/logovací nástroj. Každý uživatel si z nabídky může podle svých potřeb vybrat pouze ty widgety, které potřebuje.

4.2 Widgety

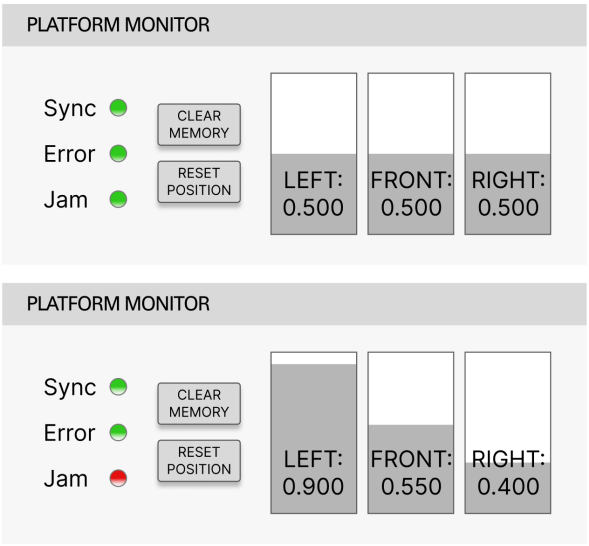
Použití widgetů tedy značně zpřehledňuje celý systém. Zároveň otevírá okno pro snadnou rozšiřitelnost systému v budoucnu. Studenti budou moci navrhovat a implementovat vlastní widgety na základě aktuálních potřeb simulátoru. Zároveň se přidáním nových funkcionalit pomocí widgetů nenaruší konzistence uživatelského rozhraní. Uvádím zde ukázku dvou navržených widgetů.

- **Přehled stavu modulů**
  První widget slouží k přehledu o stavu jednotlivých modulů. Zelená dioda značí, že je modul online a vývojář tedy ví, že je možné s ním komunikovat. V případě výpadku změní dioda u patřičného modulu barvu na červenou. Viz obrázek 4.1.



Obrázek 4.1: Module overview widget

- **Ovládání pohyblivé platformy**
  Tento widget slouží k monitorování a ovládání stavu pohyblivé části simulátoru. Viz obrázek 4.2.



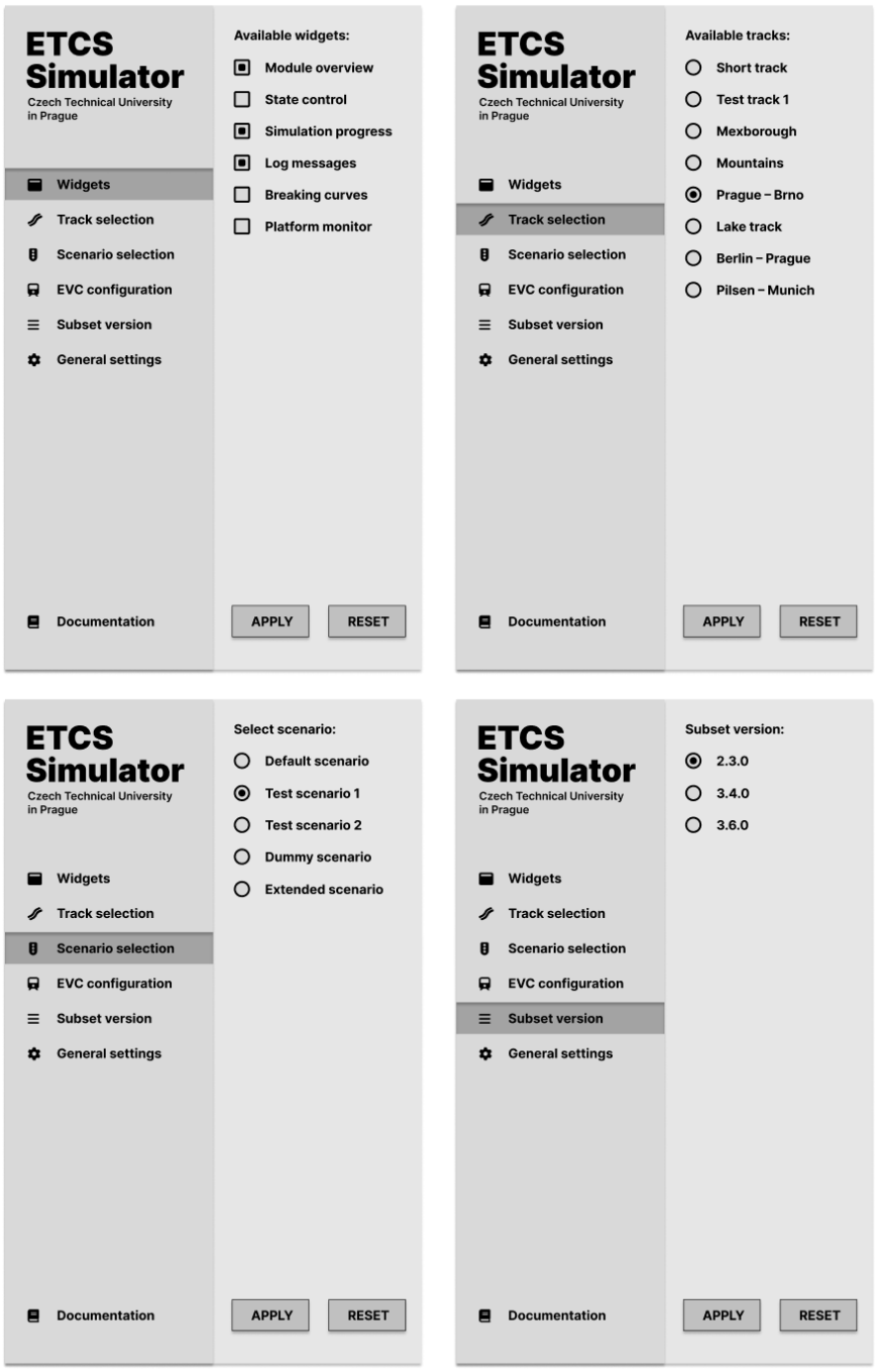
Obrázek 4.2: Platform monitor widget

4.3 Low-fidelity prototyp

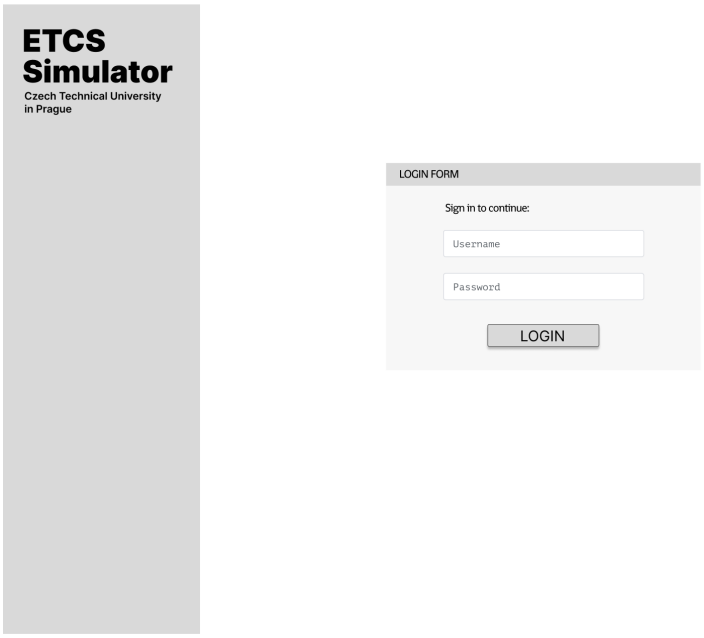
Na základě náčrtů a drátového modelu jsem začal vytvářet první klikatelný prototyp uživatelského rozhraní. Ke tvorbě prototypu jsem využil již zmíněného prototypovacího nástroje Figma [18]. Úvodní obrazovka prototypu Lektorského PC obsahuje prázdnou plochu a levou postranní nabídku. Postranní menu (viz obrázek 4.3) slouží ke konfiguraci simulace, například výběr tratě nebo scénáře a k přidání nových widgetů, určených převážně k monitorování a řízení běhu. Výsledný low-fidelity prototyp je možné vyzkoušet ve webové aplikaci Figma pod odkazem v příloze. Vzhledem k tomu, že se jedná o low-fidelity prototyp, je potřeba se pevně držet scénáře uvedeného v sekci 5.2.

4.3.1 Ukázka low-fidelity prototypu

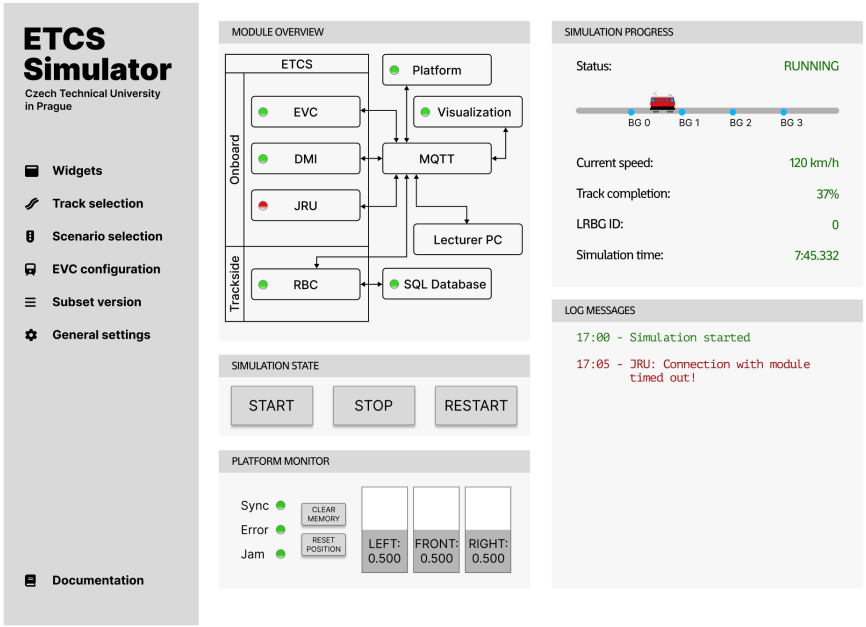
Ukázka přihlašovací obrazovky low-fidelity prototypu viz obrázek 4.4, ukázka pracovní plochy zaplněné widgety viz obrázek 4.5.



Obrázek 4.3: Ukázka postranního menu low-fidelity prototypu



Obrázek 4.4: Ukázka přihlašovací obrazovky low-fidelity prototypu



Obrázek 4.5: Ukázka low-fidelity prototypu s aktivními widgety

Kapitola 5

Testování

Pro účely tohoto projektu jsem pro testování využil dvě metody. Testování heuristickou evaluací pomocí Nielsenových pravidel a uživatelské testování použitelnosti provedené na cílové skupině uživatelů v laboratoři. K testování byl využit low-fidelity prototyp. V praxi testování uživatelských rozhraní na low-fidelity prototypech šetří velké množství času a financí. Více o výhodách low-fidelity prototypu je uvedeno v sekci 1.3.1. Na základě výsledků testování uživatelského rozhraní implementuji webovou aplikaci Lektorské PC v kapitole 6.

5.1 Heuristické testování

V praxi by měli heuristické testování provádět alespoň tři nezávislí odborníci, aby se dosáhlo co nejlepších výsledků. Pro účely této bakalářské práce jsem se ale o kognitivní průchod pomocí heuristické metody pokusil sám. Další nedostatky návrhu odhalí uživatelské testování uvedené v sekci 5.2. K testování využívám zjednodušená Nielsenova hodnotící kritéria uvedená v sekci 1.4.1.

5.1.1 Systém hodnocení

Každá z Nielsenových heuristik je rozdělena na dvě podotázky, kterým hodnotící přidělí procentuální hodnocení (100 % - vlastnost systému je splněna, 0 % - jedná se o závažný problém s použitelností). Podotázky, vycházející z nielsenových pravidel, jsou inspirované otázkami z předmětu *Tvorba uživatelského rozhraní* [2]. Hodnocení konkrétní heuristiky se získá zprůměrováním hodnocení podotázek. V následující části je uveden výpis otázek, včetně jejich procentuálního hodnocení a zdůvodnění. Celkové výsledky heuristického testování, včetně tabulky, jsou shrnuty v sekci 5.1.2.

- **Viditelnost stavu systému**
  - Ví uživatel vždy, kde se v systému nachází?
    Hodnocení: 100 %
    Jedná se o single-page aplikaci, tedy uživatel se nemůže ztratit v hlubší struktuře systému. V případě menu je vždy zvýrazněná aktuálně otevřená položka.
  - Ví uživatel vždy, v jakém je aplikace stavu?
    Hodnocení: 100 %
    Uživatel je o stavu simulátoru a jeho komponent informován prostřednictvím konzole a monitorovacích widgetů.
- **Shoda mezi systémem a reálným světem**
  - Jsou názvy, popisky a používané termíny v souladu s terminologií cílové skupiny?
    Hodnocení: 100 %
    Terminologie je daná zadavatelem dodanou literaturou a technickými specifikacemi Evropské železniční agentury.
  - Jsou prvky uživatelského rozhraní logicky uspořádané?
    Hodnocení: 100 %
    Uspořádání je dané rozestavěním widgetů. Všechny ovládací prvky jedné kategorie tak uživatel nalezne ve stejné oblasti.
- **Kontrola uživatele a svoboda**
  - Může uživatel každou akci vrátit zpět?
    Hodnocení: 100 %
    Uživatel v systému nemůže provádět žádné nevratné akce. Uživatel má svobodu nastavení simulace kdykoliv změnit pomocí vždy přítomného postranního menu.
  - Je vrácení akce snadné? Obejde se bez nutnosti dlouhé sekvence akcí?
    Hodnocení: 100 %
    Pokud uživatel špatně nakonfiguroval simulaci, má možnost ji kdykoliv ukončit a z postranního menu zvolit jiné nastavení. Není potřeba vracet se několik kroků zpět. Stačí změnit konkrétní hodnotu a simulaci znovu spustit.
- **Konzistence a standardy**
  - Je uživateli jasný význam všech termínů, situací a akcí?
    Hodnocení: 70 %
    Systém převážně mluví jazykem uživatele. Mezi nové termíny, které se uživatel musí naučit patří „widget“.

  - Jsou dodrženy průmyslové standardy a je uspořádání logické pro cílovou skupinu? Jsou akce a jejich význam konzistentní s očekáváním cílové skupiny?
    Hodnocení: 100 %
    Vyžadovované akce pro spuštění simulátoru následují běžné konvence průmyslu. Spuštění simulátoru předchází nastavení počátečních konfigurací a výběr trasy s požadovaným scénářem.

- **Prevence chyb**

  - Jsou součástí rozhraní prvky, které usnadňují práci a předcházejí chybám?
    Hodnocení: 100 %
    Systém předchází chybám například tím, že nabízí uživateli pouze scénáře kompatibilní se zvolenou trasou. Simulátor není možné spustit bez zvolení všech potřebných konfigurací.
  - Podporuje systém akce „vrátit zpět“ nebo případně „resetovat nastavení“?
    Hodnocení: 100 %
    Vzhledem k tomu, že se jedná o single-page aplikaci, tlačítko „vrátit zpět“ není přítomno. Ze všech akcí se ale dá vrátit zpět do předchozího stavu jedním klikem a to i z menu (zavřením menu nebo potvrzením nastavení) nebo z probíhající simulace (zastavením simulace). Každé nastavení se dá resetovat zpět na defaultně nastavené hodnoty.

- **Přehlednost nad zapamatovatelností**

  - Jsou všechny potřebné informace vidět, aniž by si je uživatel musel pamatovat?
    Hodnocení: 50 %
    Informace o vybrané trase, scénáři a počáteční konfiguraci jsou k nalezení v konzoli nebo po otevření příslušného menu. Považuji to za nedostatek, uživatel by měl vidět jaká data simulátoru předává, aniž by otevíral jednotlivá menu nebo scrolloval v konzoli. Uživatel si v aktuální verzi prototypu musí před finálním potvrzením pamatovat, jaká nastavení již zvolil.
  - Jsou informace dostupné i po přechodu do jiné části systému, pokud jsou potřeba?
    Hodnocení: 100 %
    Všechny příchozí informace i provedené akce se ukládají do logovací konzole, kde zůstávají k dohledání i po přechodu do jiného stavu systému.

- **Flexibilita a efektivita používání**

  - Je používání aplikace efektivní? Nevyžadují jednoduché akce příliš mnoho kroků?
    Hodnocení: 100 %
    Všechny podporované akce vyžadují minimum kroků. Uživatel pokaždé upravuje pouze nezbytná nastavení pro spuštění simulace.
  - Umožňuje systém pokročilejšímu uživateli urychlit často opakované akce?
    Hodnocení: 100 %
    Pokročilý uživatel si může nastavit vlastní widgety, čímž se mu značně usnadní práce. Zároveň si systém pamatuje konfigurace nastavené z minula a tak nemusí pro spuštětní stejného scénáře vše znovu nastavovat.

- **Estetika a minimální design**
  - Neobsahuje uživatelské rozhraní irelevantní informace?
    Hodnocení: 100 %
    Uživatel má plnou kontrolu nad tím, jaké informace mu systém zobrazuje.
  - Je uživatelské rozhraní přehledné?
    Hodnocení: 100 %
    Důležité ovládací prvky jsou výrazné a rozložení monitorovaných komponent je inspirované fyzickou předlohou. Menu s nastavením je přítomno na stejném místě ve všech stavech simulátoru. Nástroje k monitorování stavu a průběhu simulace jsou rozdělené mezi jednotlivé widgety.

- **Pomoc živatelům s diagnostikou a opravou chyb**
  - Jsou chybové hlášky psány jazykem uživatele?
    Hodnocení: 80 %
    Většina chybových hlášek, s výjimkou síťových výpadků, jsou v souladu se zavedenou terminologií. Běžný uživatel by nemusel rozumět výstupům výpadku komunikace s některými moduly. Z těchto důvodů je ale výstup z logovací konzole doplněn grafickým znázorněním aktivity modulů pomocí diod (zelená značí, že modul komunikuje a vše je v pořádku.)
  - Je u chybových hlášek navržené jejich řešení?
    Hodnocení: 40 %
    Pokud například zvolený scénář nebyl platný, uživatel je požádán o výběr jiného formou výstupu v logovací konzoli. K některým problémům není uživateli dostupný návod a je vyžadován zásah zkušenějšího uživatele nebo vývojáře.

- **Nápověda a dokumentace**

– Je uživatel schopný se v systému orientovat bez nápověd a dokumentace?
Hodnocení: 80 %
Uživatelské rozhraní je navržené, aby bylo pro cílovou skupinu uživatelů intuitivní. Předpokládá se ale, že uživatel rozumí konceptu widgetů.

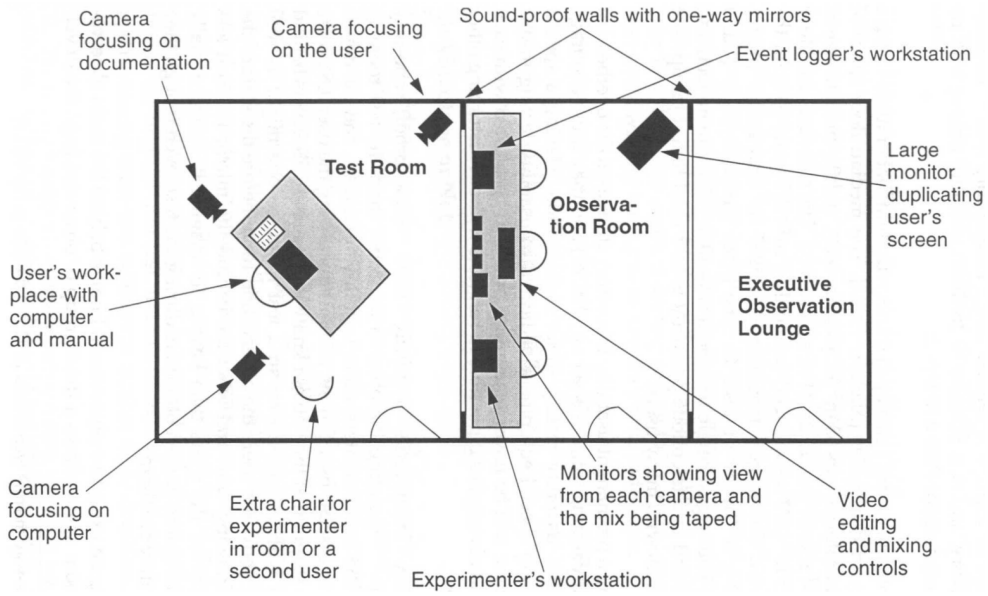
– Je případná nápověda či dokumentace uživateli dostupná?
Hodnocení: 20 %
Uživatelské rozhraní obsahuje několik nápověd. Například je uživatel upozorněn, pokud nemá zobrazené žádné widgety. V této verzi prototypu není uživateli dostupná dokumentace.

5.1.2 Výsledky heuristické evaluace

Testování heuristickou analýzou odhalilo některé nedostatky prototypu. Menu uživatelského rozhraní by mělo být doplněno o přehled zvolených nastavení, aby si je uživatel nemusel do spuštění simulace pamatovat, zároveň aby byl informován, která nastavení je potřeba pro spuštění simulace doplnit. Pro nezkušeného uživatele by nemusely být dostačující chybové výpisy v konzoli a tak by bylo vhodné doplnit přehledný systém oznamování chyb, včetně návodu k jejich řešení. Dokumentace není součástí prototypu, ale uživatelské rozhraní obsahuje nápovědy pro pomoc s orientací. Jestli jsou nápovědy dostatečné ukážou uživatelské testy. Shrnuté výsledky testování heuristickou evaluací pomocí Nielsenových kritérií viz tabulka 5.1.

Tabulka 5.1: Evaluace použitelnosti systému

Heuristika	*Hodnocení*
Viditelnost stavu systému	100 %
Shoda mezi systémem a reálným světem	100 %
Kontrola uživatele a svoboda	100 %
Konzistence a standardy	85 %
Přehlednost nad zapamatovatelností	100 %
Prevence chyb	75 %
Flexibilita a efektivita používání	100 %
Estetika a minimální design	100 %
Pomoc živatelům s diagnostikou a opravou chyb	60 %
Nápověda a dokumentace	50 %
Celkem	87 %



Obrázek 5.1: Plánek typické laboratoře pro testování použitelnosti [1]

5.2 Uživatelské testování

Testování proběhlo v UsabilityLabu (laboratoř pro testování uživatelského rozhraní) na Fakultě informačních technologií, ČVUT. Cílem bylo otestovat lidi z obou cílových skupin. Na obrázku 3.1 je vidět plánek typické laboratoře pro testování použitelnosti převzatý z knihy „Usability Engineering“ [1].

Ve fakultní laboratoři jsou umístěny celkem tři kamery. První kamera zabírá obličej testera. Druhá je umístěna nad pracovištěm s počítačem a zabírá práci uživatele s klávesnicí a myší. Třetí kamera je širší pohled na celou místnost, která zabírá testera i instruktora. Součástí záznamu je i obrazovka uživatele. Záběr všech kamer v laboratoři před zahájením testování viz foto 5.2 (fotografie z průběhu testování zde neuvádím, protože bylo účastníkům slíbeno, že záznam nebude zveřejněn).

Scénář byl nastaven tak, aby odpovídal běžné činnosti budoucího uživatele simulátoru a zároveň prototyp postupně simuloval chyby různého původu. Tím nutil testery řešit podobné problémy, se kterými se cílové skupiny lidí budou setkávat během používání a testování hotového produktu. Chyby byly navrženy, aby otestovali interakce uživatele s každou komponentou uživatelského rozhraní. Jedná se o chyby tří druhů:

- **Přerušení komunikace**
  Během vývoje a testování simulátoru může v některých případech nastat problém s komunikací s jedním či více moduly. V praxi by bylo potřeba zjistit původ problému a nekomunikující modul opravit. Pro potřeby

testování uživatelského rozhraní postačí, když uživatel identifikuje nekomunikující modul a simulaci restartuje.

- **Neplatná data v paketu**
  V praxi taková situace nastane, pokud uživatel (například ve scénáři) vyplnil údaje, které se neshodují s tratí, tak jak je definovaná v SQL databázi. Validace paketu v cílovém modulu takovou chybu odhalí a nahlásí ji uživateli v logu.

- **Hardwarový problém**
  Jedná se o problémy s fyzickou částí simulátoru, tedy například s pohyblivou plošinou. Testuje se, jestli je uživatel schopný detekovat zaseknutí polohy pohyblivé platformy.

Zde je uvedena ukázka použitého testovacího scénáře. Jeho podrobnější verze, včetně výsledků je, v příloze:

Testovací scénář

Odhadovaný čas

10 min.

Účel testování

Testujeme uživatelské rozhraní webové aplikace určené primárně pro lektory nového vlakového zabezpečovacího systému. Uživatelské rozraní slouží pro konfiguraci, ovládání stavu simulace a vizualizaci příchozích paketů. Je tedy žádoucí, aby uživatelské rozhraní rovněž sloužilo i jako nástroj pro debugování vývojářům simulátoru.

Potřebujeme otestovat schopnost uživatele se v rozhraní orientovat a provádět základní operace. Mezi ty patří: ovládání stavu simulace a řešení případných problémů, které se v průběhu mohou vyskytnout.

Počáteční bod

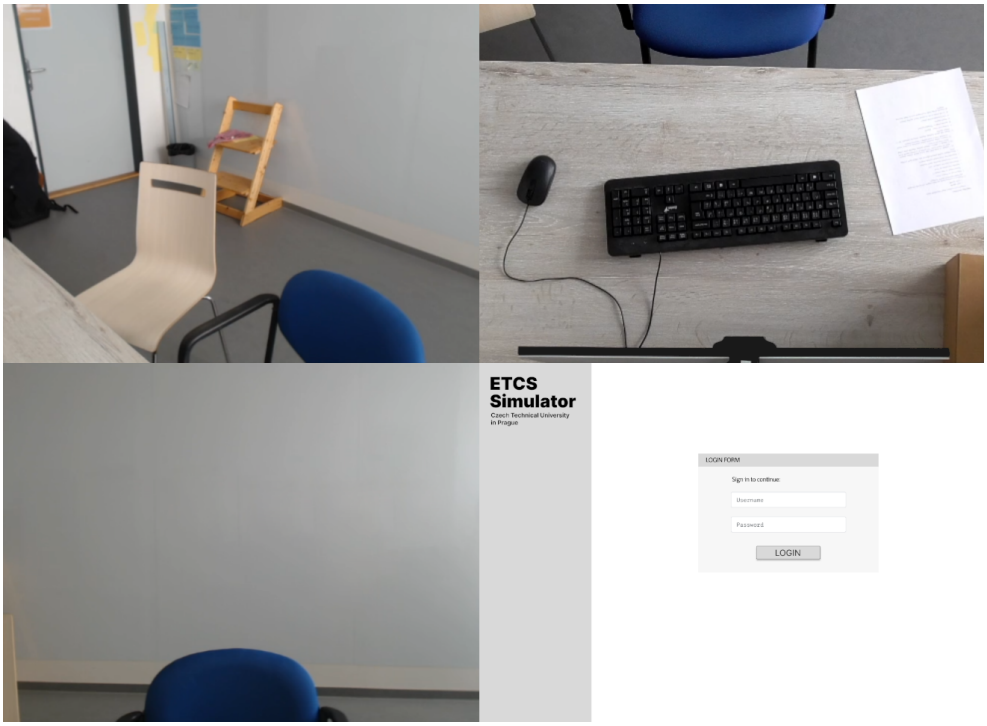
Otevřená přihlašovací stránka prototypu webové aplikace.

Koncový bod

Zastavená simulace po úspěšně projeté trase a vyčištěná pracovní plocha (na ploše se již nevyskytuje žádný widget).

Instrukce pro testera

1. přihlaste se do systému pomocí následujících údajů:
   - login: **admin**
   - heslo: **lpc123**
2. Přidejte si na pracovní plochu relevantní widgety (pro potřeby této simulace budete potřebovat všechny dostupné)
3. Nastavte trať na „Prague - Brno“
4. Nastavte scénář na „Test scenario 1“
5. Nastavte verzi subsetu na „3.6.0“
6. Zkontrolujte správnost nastavení v logu
7. Pokud je vše správně nastaveno, spusťte simulaci
8. Kontrolujte stav všech modulů a průběh simulace
9. Pokud některý z modulů přestane odpovídat nebo nahlásí chybu, problém vyřešte:
   a) V případě, že modul RBC hlásí chybný scénář, simulaci zastavte, zvolte jiný scénář (například „Test scenario 2“) a simulaci spusťte znovu
   b) V případě, že některý z modulů přestane odpovídat nebo nahlásí chybu, restartujte simulaci
   c) Pokud se zasekne pohyblivá plošina (platform), je potřeba vyčistit její paměť, resetovat pozici plošiny a restartovat simulaci
10. Jakmile vlak přejede balízovou skupinu s kódovým označením „BG 2“, simulaci zastavte
11. Nastavte trať „Pilsen - Munich“
12. Nastavte scénář na „Extended scenario“
13. Spusťte simulaci
14. Opět kontrolujte průběh simulace
15. Až vlak dojede do cíle a bezpečně zastaví, zastavte simulaci
16. Pracovní plochu uveďte do původního stavu (resetujte nastavení widgetů)



Obrázek 5.2: Záznam z laboratoře pro testování uživatelského rozhraní (UsabilityLab), Fakulta informačních technologií, ČVUT

5.2.1 Výsledky uživatelského testování

Uživatelské testy ukázaly, že se cílová skupina uživatelů v rozhraní dokáže velmi dobře orientovat. Nikdo z přítomných testerů neměl problém s hledáním ovládacích prvků ani plněním úkolů. Získal jsem však díky tomu lepší přehled, co by se dalo dále vylepšovat. Pro testery bylo neočekávané, že tlačítka stop a restart nevyžadovala další potvrzení, aby se zabránilo nechtěným zastavením/restartováním simulace ze strany lektora. Zároveň jsem byl upozorněn, že pro některé testery nebylo zcela intuitivní kontrolovat zvolené nastavení v konzoli. V příští verzi prototypu by tedy bylo vhodné zvolené nastavení zobrazovat přímo v menu, aby jej uživatel nemusel dohledávat/pamatovat si.

Testovací fáze low-fidelity prototypu je ukončena. V následující kapitole se zabývám implementací prototypu webové aplikace Lektorské PC na základě všech informací získaných v předchozích částech této práce.

Kapitola 6

Implementace

V této kapitole se zabývám implementací klientské části modulu Lektorské PC. Jedná se o webovou aplikaci s uživatelským rozhraním založeným na poznatcích získaných v předchozích kapitolách této práce. Nejprve uvádím hlavní použité technologie, dále popisuji realizovaný prototyp, možnosti budoucího vývoje a kapitolu zakončuji instalační příručkou.

Vzhledem k tomu, že se jedná o webovou aplikaci, byl k implementaci použit zejména programovací jazyk TypeScript s využitím knihovny React. Psát webové aplikace je možné v libovolném textovém editoru. Pro tuto práci byl k vývoji využit open source editor Visual Studio Code od společnosti Microsoft.

6.1 Technologický stack

V této sekci popíšu technologie použité pro vývoj klientské části modulu Lektorské PC. Výběr technologií je ovlivněn především technologiemi využitých pro vývoj serverové části a osobními zkušenostmi autora.

TypeScript

TypeScript [19] je open-source staticky typovaný programovací jazyk, postavený na jazyku JavaScript. Přidává nový syntax a poskytuje JavaScriptu typové rozhraní. Obsahuje veškeré funkcionality JavaScriptu a zároveň jej rozšiřuje o další. Je podporován širokou škálou JavaScriptových knihoven a podporuje i používání externích knihoven, které nebyly napsány v TypeScriptu. Typescript není samostatně spustitelný, a tak je potřeba jej před použitím webovým prohlížečem zkompilovat do Javascriptu.

Mezi jeho hlavní výhody oproti JavaScriptu patří:

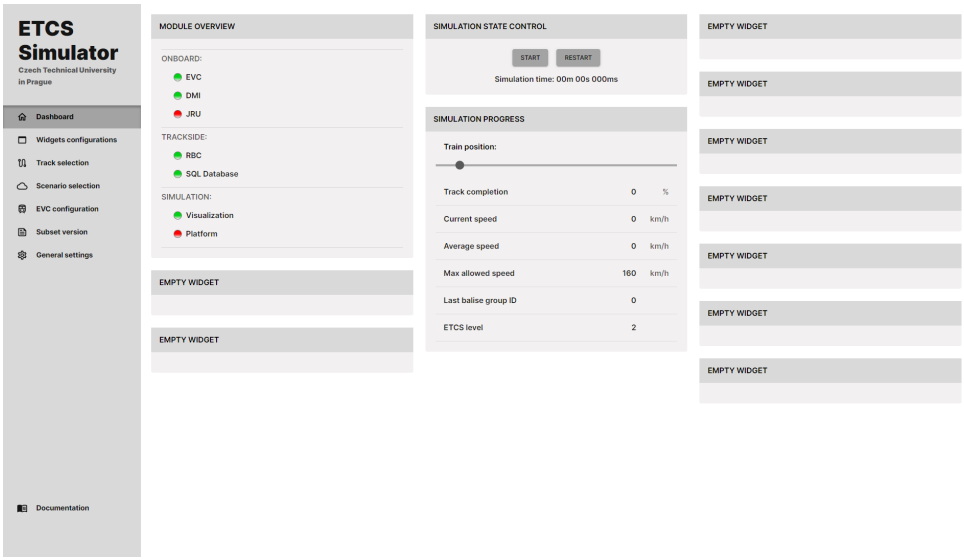
- **Statické typování**
  Umožňuje vývojáři pevně nastavit datové typy všech proměnných, parametrů funkcí a návratových hodnot. Díky tomu se snáze odhalují typové chyby při kompilaci a zlepšuje se celková čitelnost a kvalita kódu.
- **Vylepšená podpora vývojových prostředí**
  Statické typování umožňuje vývojovému prostředí (zkratka IDE) nabízet lepší nápovědy, automatické doplňování a refaktorování.
- **Podpora objektově orientovaného programování**
  Typescript je rozšířený o základní principy objektově orientovaného programování, jako jsou třídy a dědičnost. To může vést k lepší organizaci a rozšiřitelnosti kódu.

React

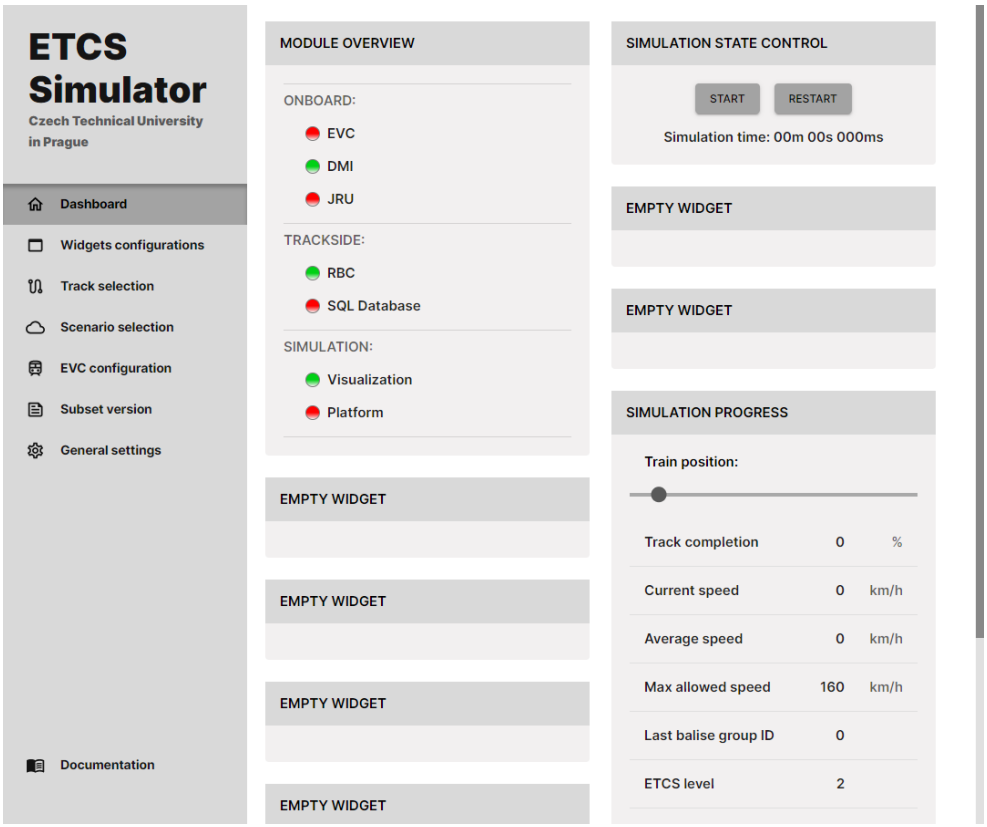
React [20] je JavaScriptová open-source knihovna pro vývoj uživatelských rozhraní webových aplikací. Aplikace napsané v Reactu se skládají z jednotlivých komponent. Každá komponenta reprezentuje část uživatelského rozhraní s vlastním vzhledem a funkcionalitami. Ve zdrojovém kódu aplikace je komponenta definována JavaScriptovou funkcí, jejíž návratovou hodnotou je HTML kód, napsaný pomocí JSX. JavaScript XML (zkráceně: JSX) je rozšířením syntaxu JavaScriptu, které umožňuje psát bloky HTML kódu uvnitř JavaScriptu, případně TypeScriptu. Hlavní výhodou používaní Reactu pro vytváření uživatelských rozhraní je jeho ekosystém a komunita. Díky tomu mají vývojáři k dispozici mnoho nástrojů a knihoven, které usnadňují a zrychlují vývoj. Aktivní komunita kolem Reactu zajišťuje, že je knihovna stále aktualizována a drží krok s nejnovějšími technologiemi.

6.2 Realizace webové aplikace

Při realizaci jsem vycházel z low-fidelity prototypu uvedného v sekci 4.3 a informací získaných z jeho testování v kapitole 5. V rámci této bakalářské práce se mi podařilo implementovat veškeré minimální funkční požadavky a některé optimální, uvedené v sekci 2.9. Uživatelské rozhraní webové aplikace umožňuje uživateli: ovládat stav simulace, monitorovat stav modulů, monitorovat průběh simulace. Dále jsem z kategorie optimálních funkčních požadavků implementoval výběr trasy, scénáře, nastavení verze subsetů a počáteční konfiguraci modulů. Uživatelské rozhraní je zároveň schopné se přizpůsobit velikosti používaného zařízení. Počet sloupců na obrazovce s widgety se dynamicky mění podle velikosti okna. Na obrázcích 6.1 a 6.2 je vidět přizpůsobení velikosti widgetů a počet sloupců v závislosti na velikosti okna (pro účely demonstrace rozložení jsou na pracovní ploše naimportovány i prázdné widgety).



Obrázek 6.1: Screenshot prototypu webové aplikace zobrazené na zařízení s vysokým rozlišením



Obrázek 6.2: Screenshot prototypu webové aplikace na přenosném zařízení

Vzhledem k tomu, že v době psaní této bakalářské práce nebyly moduly připraveny ke komunikaci s Lektorským PC, uživatelské rozhraní pro účely testování jejich běh simuluje. Další fází vývoje bude napojení prototypu na zbytek Simulátoru ETCS, až budou ostatní moduly připraveny ke komunikaci.

6.3 Instalační příručka

V této části popisuji, jak projekt zprovoznit na operačních systémech Windows 10/11 a Ubuntu 20.04. Celý projekt se nachází na fakultním GitLabu a odkaz k jeho stažení je v příloze.

1. Nejprve je potřeba nainstalovat běhového prostředí (anglicky: runtime environment) *Node.js* a správce balíčků *npm* (Node Package Manager) z následujícího odkazu:

   ```
   https://nodejs.org/en/download
   ```

2. Po instalaci *Node.js* přejděte pomocí systémové příkazové řádky do složky projektu a nainstalujte všechny potřebné balíčky následujícím příkazem:

   ```
   npm install
   ```

3. Pro otestování webové aplikace je nejprve potřeba spustit vývojový server pomocí následujícího příkazu:

   ```
   npm run dev
   ```

4. Jakmile se server spustí, vypíše zprávu s odkazem pro přístup k aplikaci. Adresa je obvykle následující:

   ```
   http://localhost:3000
   ```

6.4 Uživatelská příručka

Vzhledem k tomu, že se jedná o prototyp uživatelského rozhraní a většina funkcionalit je závislá na vývoji dalších modulů simulátoru, je pro používání aplikace doporučeno se držet definovaných scénářů uvedených v sekci 5.2. S výjimkou importu widgetů, které jsou již naimportované při prvním spuštění aplikace.

6.5 Programátorská příručka

Aplikace je strukturovaná podle funkcionalit. Zdrojový kód se nachází ve složce `src/`. Vstupním bodem aplikace je soubor `src/main.tsx`. V něm se naimportují základní knihovny pro renderování React komponent, knihovna router pro navigaci aplikací na straně klienta a soubor CSS obsahující globální styl aplikace včetně zvoleného fontu. Metoda `render()` zahájí rendering komponenty `App` definované v souboru `src/App.tsx`.

Složka `src/scenes/` obsahuje všechny stránky aplikace. Hlavní část programu se nachází ve složce `src/scenes/dashboard/`, která obsahuje úvodní obrazovku aplikace a složku se všemi naimportovanými widgety. Složka `src/widgets/` navíc obsahuje soubor `widget_template.tsx`, který vývojaři umožňuje do uživatelského rozhraní Lektorského PC doplnit další funkctionality. V souboru stačí vyplnit požadované části do vyznačených bloků, soubor přejmenovat a naimportovat v souboru `src/scenes/dashboard/index.tsx`. Složka `scenes/` navíc obsahuje podsložku `global/`. Ta definuje prvky uživatelského rozhraní nebo funkcionality, které se vztahují na všechny stránky obsažené ve stejné složce. Jedná se například o navigační menu v levé části uživatelského rozhraní, které je definované v souboru `src/scenes/global/Sidebar.tsx`. Celkový vzhled aplikace včetně tmavého a světlého motivu je definovaný v souboru `src/theme.tsx`.

Závěr

Práce se zaměřovala na návrh, tvorbu a testování uživatelského rozhraní webové aplikace Lektorské PC k projektu Simulátor ETCS.

Na začátku práce jsem popsal základní pojmy a techniky tvorby uživatelského rozhraní. V další části jsem uvedl systém vlakového zabezpečovače ETCS, popsal jeho hlavní části a vysvětlil význam jeho simulátoru. Dále jsem provedl analýzu současného stavu projektu Simulátor ETCS a věnoval se dokončení modulu RBC, potřebného pro správnou funkčnost Lektorského PC. Následně jsem analyzoval serverovou část Lektorského PC, kterou se zabýval Matěj Gorgol ve své bakalářské práci. V praktické části jsem se zabýval návrhem uživatelského rozhraní a tvorbě low-fidelity prototypu. Low-fidelity prototyp jsem podrobil testování heuristickou analýzou a uživatelskému testování použitelnosti, provedeného v laboratoři pro testování uživatelských rozhraní na Fakultě informačních technoligií, ČVUT. Na základě výsledků testování jsem položil základy prototypu webové aplikace Lektorské PC. Na závěr jsem k implementovanému prototypu uvedl příručku, která by měla usnadnit doplňování dalších funkcionalit.

Výsledkem této práce je navržený a otestovaný design uživatelského rozhraní, který byl dále uplatněn při implementaci. Prototyp klientské části Lektorského PC, v podobě webové aplikace, je funkční a připraven vstoupit do další fáze vývoje.

Literatura

[1] Nielsen, J.: Usability Engineering. San Francisco: Morgan Kaufmann, 1993, ISBN: 1-12-518406-9.

[2] Schmidt, J.: Tvorba uživatelského rozhraní (BI-TUR.21) [online]. ČVUT v Praze, Fakulta informačních technologií. Stránka předmětu: `https://courses.fit.cvut.cz/BI-TUR/` (dostupné po přihlášení do sítě ČVUT), [cit. 2023-03-01].

[3] Babich, N.: Sketch, Wireframe, Mockup, and Prototype: Why, When and How [online]. Dostupné z: `https://uxplanet.org/sketch-wireframe-mockup-and-prototype-why-when-and-how-29a25b3157c4`, [cit. 2022-06-10].

[4] Správa železnic: Co je ETCS [online]. Dostupné z: `https://www.spravazeleznic.cz/stavby-zakazky/modernizace/etcs/co-je-etcs`, [cit. 2023-03-29].

[5] The European Union Agency for Railways (ERA): Set of specifications 3 (ETCS B3 R2 GSM-R B1) [online]. Dostupné z: `https://www.era.europa.eu/era-folder/set-specifications-3-etcs-b3-r2-gsm-r-b1`, [cit. 2022-11-12].

[6] Clearsy: ERTMS/ETCS OPERATIONAL SIMULATOR [online]. Dostupné z: `https://www.clearsy.com/en/offers/ertms-etcs-operational-simulator/`, [cit. 2023-04-12].

[7] Gorgol, M.: ETCS - RBC I. Bakalářská práce, ČVUT v Praze, Fakulta informačních technologií, 2022.

[8] This person does not exist [online].
Dostupné z: `https://www.thispersondoesnotexist.com/`, [cit. 2022-08-28].

[9] Nielsen, J.: Usability 101: Introduction to Usability [online]. Dostupné z: https://www.nngroup.com/articles/usability-101-introduction-to-usability/, [cit. 2023-03-01].

[10] Babich, N.: Prototyping 101: The difference between low-fidelity and high-fidelity prototypes and when to use each [online]. *Adobe Blog*, listopad 2017. Dostupné z: https://blog.adobe.com/en/publish/2017/11/29/prototyping-difference-low-fidelity-high-fidelity-prototypes-use

[11] Kara, P.: UX Prototypes: Low Fidelity vs. High Fidelity [online]. Dostupné z: https://www.nngroup.com/articles/ux-prototype-hi-lo-fidelity/, [cit. 2023-03-01].

[12] Nielsen, J.: 10 Usability Heuristics for User Interface Design [online]. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/, [cit. 2023-03-01], duben 1994.

[13] Dopravní sál Fakulty dopravní: Projekt Simulátor ETCS [online]. Původní plakát (dostupný po přihlášení do sítě ČVUT): https://moodle-vyuka.cvut.cz/pluginfile.php/493926/mod_page/content/31/plakat.pdf, [cit. 2023-04-15].

[14] Ministerstvo dopravy: Národní implementační plán [online]. Dostupné z: https://zdopravy.cz/wp-content/uploads/2021/08/etcsplan.pdf, [cit. 2023-04-03].

[15] Eclipse Foundation: Eclipse Mosquitto: An open source MQTT broker [online]. Dostupné z: https://mosquitto.org/, [cit. 2023-04-12].

[16] Oracle: MySQL: The world's most popular open source database [online]. Dostupné z: https://dev.mysql.com/, [cit. 2022-06-10].

[17] Kasalická, K.: Rozšíření frameworku Cpputest a jeho využití k testování ETCS simulátoru. Diplomová práce, ČVUT v Praze, Fakulta informačních technologií, 2022.

[18] Figma: the collaborative interface design tool [online]. Dostupné z: https://www.figma.com/, [cit. 2023-03-30].

[19] Microsoft: TypeScript: JavaScript with syntax for types [online]. Dostupné z: https://www.typescriptlang.org/, [cit. 2023-04-12].

[20] Meta Platforms: React: The library for web and native user interfaces [online]. Dostupné z: https://react.dev/, [cit. 2023-04-12].

Příloha **A**

Seznam použitých zkratek

API Application programming interface

BG Balise group

ČVUT České vysoké učení technické v Praze

DMI Driver Machine Interface

ETCS European Train Control System

EVC European Vital Computer

GP Gradient profile

GUI Graphical user interface

JSON JavaScript Object Notation

JRU Juridical Recording Unit

LRBG Last relevant balise group

MA Movement authority

RBC Radio Block Centre

REST Representational State Transfer

SP Speed profile

XML Extensible markup language

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `prototypes` ..... odkazy na zdrojové kódy prototypů
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `user-testing.pdf` ..... výsledky testování ve formátu PDF