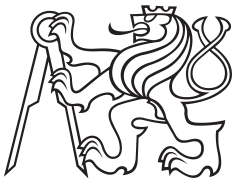
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Hybridní webová a desktopová aplikace pro správu a validaci prezentací v multiprojekčním systému

Bakalářská práce

Filip Toman

Vedoucí: Ing. Ivo Malý Ph.D.
Obor: Softwarové inženýrství a technologie
Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Toman** Jméno: **Filip** Osobní číslo: **483857**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Hybridní webová a desktopová aplikace pro správu a validaci prezentací v multiprojekčním systému

Název bakalářské práce anglicky:

Hybrid web and desktop application for management and validation of presentations in multi-projection system

Pokyny pro vypracování:

Seznamte se s projektem 5D projekce připravovaným pro Muzeum hlavního města Prahy (MMP) a s prvním prototypem jeho ovládacího systému [1]. Zaměřte se zejména na funkcionalitu dostupnou ve webové aplikaci pro řízení a správu.
Prostudujte hlavní nedostatky uživatelského rozhraní a funkcionality nalezené v rámci vyhodnocení prototypu [1].
Navrhněte novou verzi aplikace pro správu. Kromě eliminace původních nedostatků rozhraní se zaměřte také na novou funkcionalitu, jako je sjednocená správa multimediálních souborů a validace prezentací.
Výslednou aplikaci implementujte jako desktopovou aplikaci s využitím vhodné hybridní web/desktop platformy. Aplikaci otestujte pomocí uživatelských testů s alespoň 5 uživateli. Porovnejte výhody a nevýhody hybridní web/desktop aplikace vůči pouze webové verzi aplikace.

Seznam doporučené literatury:

1. O. Trojan, Řídící systém pro multiprojekci, Diplomová práce, České vysoké učení technické v Praze, 2020.
2. T. Lowdermilk, User-Centered Design, O'Reilly Media, 2013.
3. S. Kinney. Electron in action. Manning Publications Company, 2018.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Ivo Malý, Ph.D., katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **12.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **30.09.2022**

Ing. Ivo Malý, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

Poděkování

Chtěl bych poděkovat vedoucímu této práce, Ivu Malému, a také svým kolegům Vítu Říhovi a Yevgenu Ponomarenkovi za jejich rady a pomoc s její tvorbou. Speciální poděkování pak patří také členům mé rodiny za jejich dlouhodobou podporu během celého studia.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu. Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

Abstrakt

Tato práce se zabývá projektem 5D projekce připravovaným pro Muzeum hlavního města Prahy a jejím cílem je vytvořit aplikaci pro administraci a správu obsahu v rámci tohoto systému. Rozhraní je implementováno jednak ve formě webové aplikace, která je vždy snadno dostupná skrz uživatelův prohlížeč, a jednak ve formě Electron desktopové aplikace, která oproti ní nabízí přidanou funkcionalitu a díky využití hybridního přístupu také sdílí značnou část zdrojového kódu. Aplikace je implementována v jazyce TypeScript a využívá reaktivní framework Svelte pro renderování uživatelského rozhraní.

Klíčová slova: multiprojekční systém, 5D projekce, web, hybridní, Electron, Svelte, TypeScript, reaktivní, JavaScript, ECMAScript

Vedoucí: Ing. Ivo Malý Ph.D.

Abstract

This work is related to the project of 5D projection that is being developed for purposes of The City Of Prague Museum. Its goal is to create an application for administration and content management within the system. Its user interface is implemented in the form of a web application which is always easily accessible through user's browser and also in the form of a desktop application built with Electron which is able to add additional functionality and reuses a big portion of the source code thanks to its hybrid approach. The application itself is implemented in TypeScript and uses Svelte as its framework for rendering user interfaces.

Keywords: multiprojection system, 5D projection, web, hybrid, Electron, Svelte, TypeScript, reactive, JavaScript, ECMAScript

Title translation: Hybrid web and desktop application for management and validation of presentations in multi-projection system

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

1.1 Představení projektu

Cílem tohoto projektu je vytvořit webovou a desktopovou aplikaci, která bude sloužit jako administrační a ovládací rozhraní pro multi-projekční systém vytvořený pro účely Muzea hlavního města Prahy [1].

Tento multi-projekční systém spočívá v synchronizovaném přehrávání multimediálních souborů (video, obrázky) na několika obrazovkách zároveň (viz Obrázek 1.1). Aby toto bylo možné, tak se ze samostatných multimediálních souborů nejprve vytváří takzvané DisplayTracky, které určují, jaké soubory budou v danou chvíli přehrávány, na kterých obrazovkách. Slouží tak jako základní jednotka, která může být přehrávána v rámci multi-projekčního systému. Tyto vytvořené DisplayTracky se pak dále seskupují do delších sekvencí, které se nazývají DisplaySety. DisplaySety fungují jako seznam DisplayTracků, který určuje, jak se jednotlivé DisplayTracky přehrávají po sobě [1].

Můj administrační systém musí nabídnout kompletní prostředí pro tvorbu tohoto multimediálního obsahu od nahrávání souborů, přes jejich kompozici do jednotlivých DisplayTracků, až po tvorbu hotových DisplaySetů, které budou moci být spuštěny na multi-projekčním systému.

Mimo to také aplikace nabídne uživatelské rozhraní pro samotné přehrávání v rámci multiprojekčního systému a to ve 2 různých variantách. Jednak ve formě klasického ovládání v reálném čase, kdy uživatel dostane přístup k rozhraní přehrávače a zde bude moci vybrat určitý DisplaySet a v rámci něj poté přetáčet, pauzovat, zastavovat a také monitorovat stav přehrávání na jednotlivých obrazovkách. Tato varianta je určená převážně pro osobu průvodce, který tak bude moci vizuální obsah na obrazovkách vždy přizpůsobit

Obrázek 1.1: Vizualizace multi-projekčního systému [1]

svému výkladu nebo otázkám publika.

Druhou variantou je pak nastavení takzvaného Autonomního módu. V tomto módu je možné přednastavit sekvenci několika DisplaySetů a tato sekvence se poté bude cyklicky přehrávat na multiprojekčním systému, pokud zrovna nebude probíhat žádná aktivní prohlídka. Toto nastavení tak slouží k více všeobecné a pasivní vizualizaci a poslouží vlastně podobně jako klasické šetřiče obrazovky, které se ukazují při neaktivitě zařízení.

Celkově pak aplikace nabídne tři různé uživatelské role. Roli „Tvůrce“, která umožní přístup k veškerým zmiňovaným schopnostem systému, roli „Průvodce“, která je limitovaná pouze na funkcionalitu přehrávače, ale neumožní nahrávání souborů a další správu multimediálního obsahu a nakonec roli „Admin“, která kromě všech zmiňovaných funkcí také umožní správu uživatelských účtů (přidávání nových uživatelů, změna uživatelských rolí, resetování hesla, atd.).

1.2 Existující řešení

Ačkoliv stávající administrační aplikace nabízela základní požadovanou funkcionalitu, tak se potýkala především s nepřehledným uživatelským rozhraním, ve kterém se uživatelé obtížně orientovali [1]. Snahou této nové aplikace tak

je sjednotit designový jazyk aplikace do více konzistentní podoby a vytvořit rozhraní, které bude lépe indikovat hierarchii jednotlivých sekcí.

Zatímco barevná paleta původní aplikace často nedbala na barevnou konzistenci a obsahovala prvky různých barev, tak nová aplikace se striktně drží trojbarevné palety. Dominantní barvou je zde modrá, která slouží ke zvýraznění důležitých interaktivních prvků a ta je kombinovaná s několika odstíny šedé. Nakonec je použita ještě červená barva, která je všeobecně známa jako barva nebezpečí, a je zde tak využita ke zvýraznění potenciálně destruktivních akcí. Původní design také trpěl nadbytkem psaných informací, který také ztěžoval rychlé porozumění prvkům na stránce. Nová aplikace tak v reakci na to často využívá ikonografické prvky, které by měli na první pohled fungovat jako vizuální indikátor funkcionality.

Posledním markantním rozdílem je, že zatímco původní administrační rozhraní bylo dostupné pouze ve formě webové stránky, tak tento nový návrh bude dostupný i jako desktopová aplikace. To zaručí nejen rychlejší načítání, ale pro mnohé uživatele také může být koncept klasické desktopové aplikace snadno uchopitelnější než webová stránka a budou mít do administrace vždy snadný přístup skrz ikonu na ploše.

Kapitola 2

Návrh

2.1 Funkční požadavky

Tento seznam popisuje základní požadavky, které musí aplikace splňovat, aby byla použitelná v praxi a mohlo dojít k jejímu nasazení do produkce:

- Uživatelské účty
  - Aplikace umožní přihlášení a odhlášení z aplikace
  - Aplikace umožní změnu uživatelského hesla
  - Aplikace nabídne možnost „Zůstat přihlášen“ pro dlouhodobé přihlášení
  - Aplikace zobrazí možnosti v závislosti na uživatelské roli
- Administrace
  - Aplikace umožní resetování hesla vybraného uživatele přes jeho zaregistrovanou emailovou adresu
  - Aplikace umožní přidávání nových uživatelských účtů
  - Aplikace umožní mazání uživatelských účtů
  - Aplikace umožní změnu uživatelské role pro vybraného uživatele
- Přístupnost
  - Aplikace bude fungovat všech klasických desktopových až tabletových rozlišeních obrazovky
  - Aplikace nabídne přepínání mezi českou a anglickou lokalizací
  - Aplikace bude kompatibilní s nejpoužívanějšími moderními prohlížeči (Chrome, Firefox, Opera, Edge)

- Přehrávání
  - Aplikace umožní výběr DisplaySetu ke spuštění
  - Aplikace umožní ovládání aktuálně přehrávaného DisplaySet (pauza, zastavení, přeskočení na další / předchozí DisplayTrack)
  - Aplikace umožní restartování připojených zařízení v případě technických problémů
  - Přehrávač umožní přepnutí do autonomního módu
  - Přehrávač bude zobrazovat stav jednotlivých zařízení a stav přehrávání na nich
- Tvorba
  - Soubory
    - Aplikace umožní nahrávání nových multimediálních souborů
    - Aplikace umožní procházení, vyhledávání a řazení stávajících souborů
    - Aplikace umožní přejmenování souborů
    - Aplikace umožní mazání existujících souborů
    - Aplikace při mazání souboru upozorní uživatele, které DisplayTracky a DisplaySety jsou na tomto souboru závislé
  - DisplayTracky
    - Aplikace umožní tvorbu nových DisplayTracků
    - Aplikace umožní procházení, vyhledávání a řazení stávajících DisplayTracků
    - Aplikace umožní nahrání nových souborů přímo při tvorbě DisplayTracku
    - Aplikace umožní přejmenování a libovolnou úpravu DisplayTracků
    - Aplikace umožní mazání existujích DisplayTracků
    - Aplikace při mazání DisplayTracku upozorní uživatele, které DisplaySety jsou na tomto DisplayTracku závislé
  - DisplaySety
    - Aplikace umožní tvorbu nových DisplaySetů
    - Aplikace umožní procházení, vyhledávání a řazení stávajících DisplaySetů
    - Aplikace umožní přejmenování a libovolnou úpravu DisplaySetů
    - Aplikace umožní mazání existujích DisplayTracků
  - Autonomní mód
    - Aplikace umožní vytvoření sekvence DisplaySetů pro autonomní mód
    - Aplikace umožní libovolnou úpravu sekvence autonomního módu

2.2 Případy užití

Tato část popisuje hlavní procesy, které jsou používané při práci s aplikací:

2.2.1 Vytvoření uživatelského účtu

Aktér: Administrátor

Vstupní podmínky:

- Uživatel přihlášený k aplikaci

Hlavní scénář:

1. Uživatel najede na ikonu Menu
2. Systém zobrazí menu možností
3. Uživatel vybere možnost Accounts
4. Systém zobrazí tabulku uživatelských účtů
5. Uživatel do vstupních polí na posledním řádku tabulky uživatelů zadá požadované uživatelské jméno, e-mail a roli nového uživatele
6. Uživatel klikne na tlačítko Create Account
7. Systém přidá nového uživatele do tabulky

Alternativní scénáře:

Neplatné údaje

7. Systém zobrazí modál popisující chybu v zadaných údajích
8. Uživatel zavře modál

Návrat do bodu 4

2.2.2 Nahrání souboru

Aktér: Administrátor / Tvůrce

Vstupní podmínky:

- Uživatel přihlášený k aplikaci

Hlavní scénář:

1. Uživatel najede na ikonu Menu
2. Systém zobrazí menu možností
3. Uživatel vybere možnost Create
4. Systém zobrazí Create stránku (ve výchozím stavu zobrazená správa autonomního módu)
5. Uživatel vybere možnost Manage Files z postranního panelu
6. Systém zobrazí prohlížeč souborů
7. Uživatel klikne na tlačítko Upload File
8. Systém zobrazí dialog pro výběr souboru
9. Uživatel vybere soubor a klikne na tlačítko Otevřít v dialogu
10. Systém zobrazí progress bar po dobu uploadu
11. Systém přidá nový soubor do prohlížeče souborů

Alternativní scénáře:

Drag and Drop

7. Uživatel přetáhne soubor do prohlížeče souborů

Návrat do bodu 10

Neplatný soubor

10. Systém zobrazí modál popisující chybu při nahrávání souboru (např. duplicitní jméno, překročení maximální velikosti, nepodporovaný formát, atd.)
11. Uživatel zavře modál

Návrat do bodu 6

2.2.3 Vytvoření DisplayTracku

Aktér: Administrátor / Tvůrce

Vstupní podmínky:

- Uživatel přihlášený k aplikaci

Hlavní scénář:

1. Uživatel najede na ikonu Menu
2. Systém zobrazí menu možností
3. Uživatel vybere možnost Create
4. Systém zobrazí Create stránku (ve výchozím stavu zobrazená správa autonomního módu)
5. Uživatel klikne na Manage DisplayTracks v postranním panelu
6. Systém zobrazí submenu pro správu DisplayTracků
7. Uživatel klikne na tlačítko Create New DisplayTrack
8. Systém zobrazí rozhraní pro tvorbu DisplayTracku
9. Uživatel klikne na ikonu tužky vedle názvu DisplayTracku
10. Systém místo názvu zobrazí editovatelné textové pole
11. Uživatel vyplní název DisplayTracku
12. Uživatel zvolí cílový počet klientů pro daný DisplayTrack
13. Systém zobrazí požadovaný počet klientů
14. Uživatel klikne na ikonu tužky v panelu daného klienta
15. Systém zobrazí overaly pro výběr souboru
16. Uživatel vybere požadovaný soubor
17. Systém skryje overlay a v panelu daného klienta se zobrazí název vybraného souboru
18. Uživatel klikne na tlačítko Save
19. Systém zašedne tlačítko Save a nově vytvořený DisplayTrack se objeví v listu v postranním menu

Alternativní scénáře:

Potřebný soubor ještě není nahraný

16. Uživatel přetáhne požadovaný soubor do prohlížeče souborů nebo ho vybere po stisknutí tlačítka Upload File
17. Systém zobrazí progress bar po dobu uploadu
18. Systém přidá nový soubor do prohlížeče souborů

Návrat do bodu 16

Ne všichni klienti mají určený soubor

19. Systém zobrazí modál popisující chybu
20. Uživatel zavře modál

Návrat do bodu 14

Uživatel chce smazat klienta

18. Uživatel klikne na tlačítko křížku u vybraného klienta
19. Systém odebere klienta ze seznamu a sníží počítadlo klientů

Návrat do bodu 18

2.2.4 Vytvoření DisplaySetu

Aktér: Administrátor / Tvůrce

Vstupní podmínky:

- Uživatel přihlášený k aplikaci

Hlavní scénář:

1. Uživatel najede na ikonu Menu
2. Systém zobrazí menu možností
3. Uživatel vybere možnost Create
4. Systém zobrazí Create stránku (ve výchozím stavu zobrazená správa autonomního módu)
5. Uživatel klikne na Manage DisplaySets v postranním panelu

6. Systém zobrazí submenu pro správu DisplaySetů
7. Uživatel klikne na tlačítko Create New DisplaySet
8. Systém zobrazí rozhraní pro tvorbu DisplaySetů
9. UUživatel klikne na ikonu tužky vedle názvu DisplaySetu
10. Systém místo názvu zobrazí editovatelné textové pole
11. Uživatel vyplní název DisplaySetu
12. Uživatel klikne na tlačítko Add New DisplayTrack
13. Systém zobrazí pop-up seznam existujících DisplayTracků
14. Uživatel vybere požadovaný DisplayTrack
15. Systém přidá DisplayTrack jako novou položku do časové osy
16. Uživatel klikne na tlačítko Save
17. Systém zašedne tlačítko Save a nově vytvořený DisplaySet se objeví v listu v postranním menu

Alternativní scénáře:

Uživatel chce změnit vybraný DisplayTrack na časové ose

16. Uživatel klikne na ikonu tužky vedle DisplayTracku, který chce změnit
17. Systém zobrazí pop-up seznam existujících DisplayTracků
18. Uživatel vybere požadovaný DisplayTrack
19. Systém změní DisplayTrack v daném bodě na časové ose

Návrat do bodu 16

Ne všichni klienti mají určený soubor

16. Uživatel klikne na ikonu křížku vedle DisplayTracku, který chce odstranit
17. Systém zobrazí modál pro potvrzení akce
18. Uživatel akci potvrdí
19. Systém odebere daný bod z časové linky

Návrat do bodu 16

2.2.5 Změna hesla

Aktér: Administrátor / Tvůrce / Průvodce

Vstupní podmínky:

- Uživatel přihlášený k aplikaci

Hlavní scénář:

1. Uživatel najede na ikonu Menu
2. Systém zobrazí menu možností
3. Uživatel vybere možnost Settings
4. Systém zobrazí formulář pro změnu hesla
5. Uživatel zadá své staré heslo a dvakrát zadá nové heslo
6. Uživatel klikne na tlačítko Save
7. Systém uloží změnu a zašedne tlačítko Save

Alternativní scénáře:

Chyba při vyplnění formuláře

6. Systém zobrazí modál popisující chybu (např. neshodující se hesla, špatně zadané staré heslo, atd.)
7. Uživatel modál zavře a opraví chybu ve formuláři

Návrat do bodu 5

2.2.6 Spuštění DisplaySetu

Aktér: Administrátor / Tvůrce / Průvodce

Vstupní podmínky:

- Uživatel přihlášený k aplikaci
- Uživatel je na úvodní stránce aplikace

Hlavní scénář:

1. Uživatel klikne na tlačítko Autonomous Mode ON a tím vypne autonomní mód aplikace a přebere kontrolu nad zobrazovaným obsahem obsahem
2. Systém odšedne ovládání přehrávače
3. Uživatel klikne na tlačítko Change DisplaySet
4. Systém zobrazí menu pro výběr DisplaySetu, který bude načtený do přehrávače
5. Uživatel klikne na DisplaySet, který chce spustit
6. Systém skryje menu a aktualizuje informace v přehrávači podle vybraného DisplaySetu
7. Uživatel klikne na ikonu šipky pod časovou osou přehrávače
8. Systém změní ikonu na ikonu pauzy a spustí vybraný DisplaySet na připojených obrazovkách

2.2.7 Restartování klientského zařízen

Aktér: Administrátor / Tvůrce / Průvodce

Vstupní podmínky:

- Uživatel přihlášený k aplikaci
- Uživatel je na úvodní stránce aplikace

Hlavní scénář:

1. Uživatel v seznamu Active Devices najde zařízení, které chce restartovat, a klikne na tlačítko Restart vedle něj
2. Systém zobrazí popup pro potvrzení volby
3. Uživatel potvrdí volbu
4. Systém odešle požadavek na restartování vybraného stroje

Alternativní scénáře:

Zrušení volby

3. Uživatel klikne na tlačítko Zrušit

Návrat do bodu 1

2.2.8 Nastavení autonomního módu

Aktér: Administrátor / Tvůrce

Vstupní podmínky:

- Uživatel přihlášený k aplikaci

Hlavní scénář:

1. Uživatel najede na ikonu Menu
2. Systém zobrazí menu možností
3. Uživatel vybere možnost Create
4. Systém zobrazí Create stránku (ve výchozím stavu zobrazená správa autonomního módu)
5. Uživatel klikne na tlačítko Save
6. Systém zašedne tlačítko Save a uloží informace o nové sekvenci autonomního módu

Alternativní scénáře:

Přidání DisplaySetu

5. Uživatel klikne na tlačítko Add New DisplaySet
6. Systém zobrazí pop-up seznam existujících DisplaySetů
7. Uživatel vybere DisplaySet, který chce přidat do sekvence
8. Systém přidá vybraný DisplaySet na konec časové linky autonomního módu

Návrat do bodu 5

Odebrání DisplaySetu

5. Uživatel klikne na křížek vedle DisplaySetu, který chce odebrat ze sekvence
6. Systém odebere DisplaySet z časové linky autonomního módu

Návrat do bodu 5

Změna DisplaySetu

5. Uživatel klikne na tlačítko tužky vedle DisplaySetu, který chce změnit
6. Systém zobrazí pop-up seznam existujících DisplaySetů
7. Uživatel vybere DisplaySet, který za který chce vyměnit položku v časové lince
8. Systém změní informace o upravené položce

Návrat do bodu 5

Kapitola 3

Uživatelské rozhraní

3.1 Login

Login stránka (viz Obrázek 3.1) je ve z hlediska obsahu velmi minimalistická. Jednoduchý přihlašovací formulář se nachází přímo ve středu obrazovky a tím na sebe instantně upoutává pohled uživatele. Tento formulář obsahuje textová pole pro zadání uživatelského jména a hesla, velké tlačítko pro odeslání požadavku na přihlášení a menší zaškrtávací pole, díky kterému může daný uživatel zůstat přihlášen dlouhodobě a jeho session [5] nevyprší. Vizuál této stránky je pak podpořený menším logem v horní části a animovanou vlnou ve spodní části, která mi pomůže představit uživateli modrou barvu jako primární barvu celé aplikace.

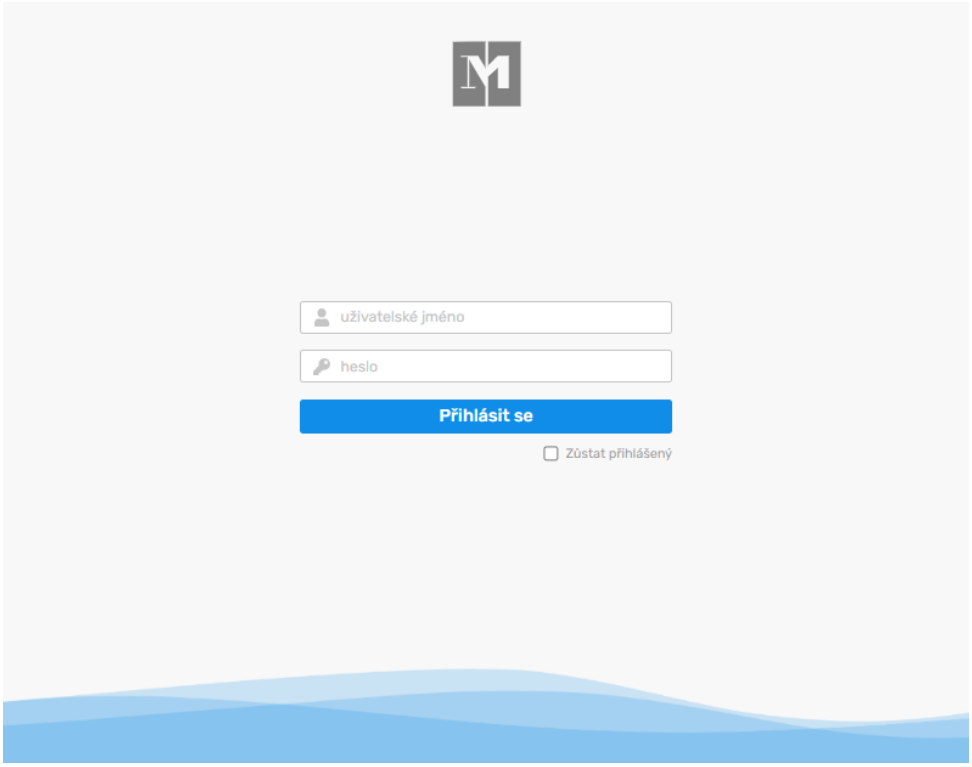
3.2 Navigační lišta

Navigační lišta (viz Obrázek 3.2) se vždy nachází na vrchu celé stránky a jde o prvek, který sjednocuje veškeré stránky této aplikace. Levá strana lišty je vyhrazené místo pro logo aplikace, případně další branding prvky, a pravá strana obsahuje samotné ovládání pro navigaci.

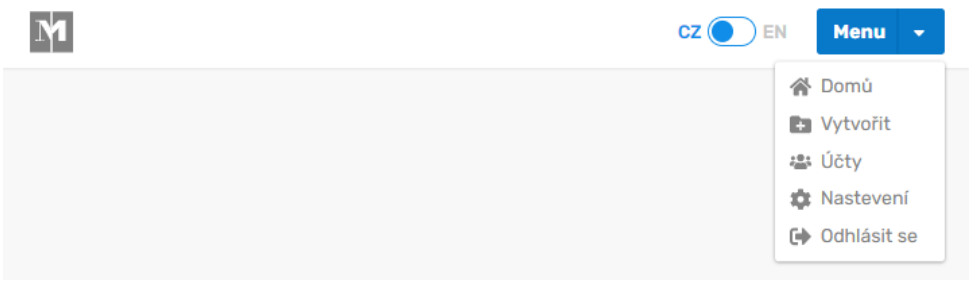
Hlavním prvkem této lišty je velké modré tlačítko s nápisem Menu, které svojí barvou kontrastuje s bílým pozadím a poutá tak na sebe nejvíce pozornosti [2]. Najetí na toho tlačítko slouží jako gesto pro odkrytí podřazeného pop-up menu, které obsahuje seznam všech hlavních sekcí aplikace a při kliknutí na určitou položku tak spustí navigaci do této sekce.

Vedle tlačítka menu se pak nachází dvou-stavový přepínač pro změnu jazyka

stránky. Design tohoto přepínače je inspirován ikonickým přepínačem systému iOS [7] a díky tomu, že je vždy snadno viditelný na navigační liště, tak je uživatel hned při první návštěvě informován o dostupnosti více lokalizací.



Obrázek 3.1: Login stránka



Obrázek 3.2: Navigační lišta

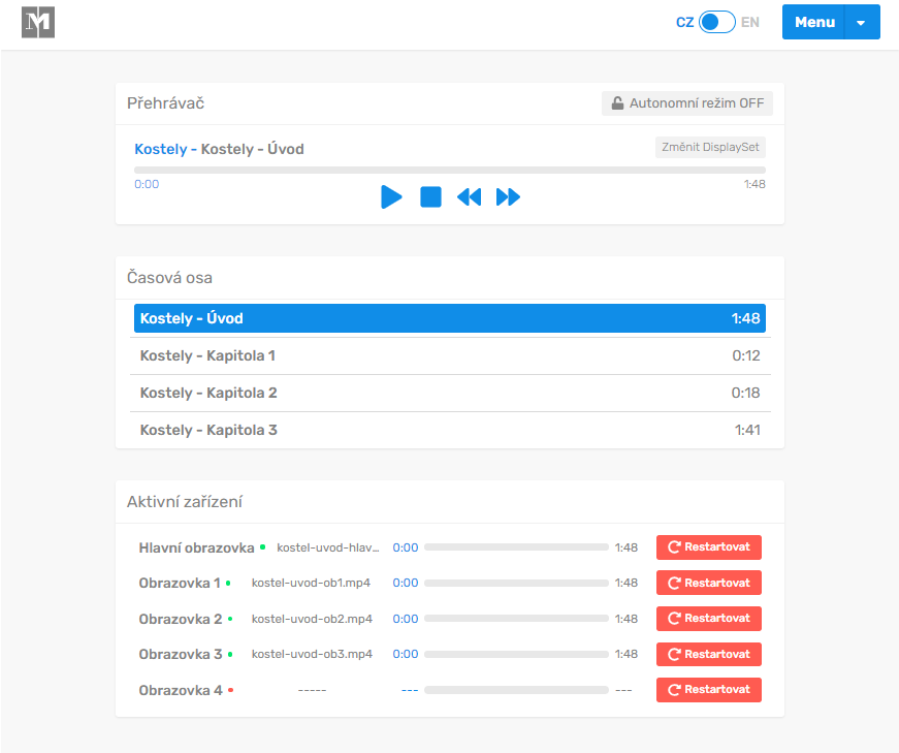
3.3 Home

Tato stránka (viz Obrázek 3.3) slouží jako hlavní kontrolní panel pro přehrávání multi-projekčního systému. Nejvýše postaveným prvkem na obrazovce je tak samotné okno přehrávače, které zobrazuje informace o aktuálně přehrávaném médiu, vizualizuje časovou linku aktuálního DisplayTracku a nabízí ovládací prvky, které jsou vyobrazené standardní ikonografií pro spuštění

(šipka), pauzu (dvě čáry), zastavení (čtverec) a přetáčení (dvojitá šipka). Vedle informací o aktuálním přehrávání se pak nachází tlačítko pro změnu DisplaySetu, čímž je indikována návaznost mezi těmito prvky [2]. Nad celým tímto rozhraním přehrávače pak leží tlačítko pro vypnutí a zapnutí autonomního módu a je podpořenou ikonou zámku. To uživateli komunikuje, že toto nastavení je nadřazené přehrávači a pokud je tak aktivován autonomní mód, tak je deaktivováno manuální přehrávání a kontrolu nad obrazovkami získává automatizovaný systém.

Pod oknem přehrávače se nachází okno Timeline. Zde uživatel získá více detailní přehled o právě přehrávaném DisplaySetu a v chronologicky seřazeném seznamu může vidět, které DisplayTracky tento DisplaySet obsahuje a jak dlouhý každý tento DisplayTrack je. Položky v tomto seznamu také zároveň fungují jako tlačítka a kliknutím na specifický DisplayTrack tak může uživatel přetočit přehrávání na jeho začátek.

Posledním oknem této stránky je Active Devices, které informuje uživatele o aktuálním stavu připojených obrazovek. Zde je vidět seznam všech zařízení a informace o tom, zda je daná obrazovka zrovna v provozu (zelený/červený indikátor vedle názvu), název souboru, který obrazovka přehrává, časová osa pro tento soubor a tlačítko pro nouzové restartování klientského zařízení. Tlačítko Restart je pak zvýrazněno červenou barvou, jelikož jde o destruktivní akci, která dané zařízení vyřadí z provozu po dobu restartu.

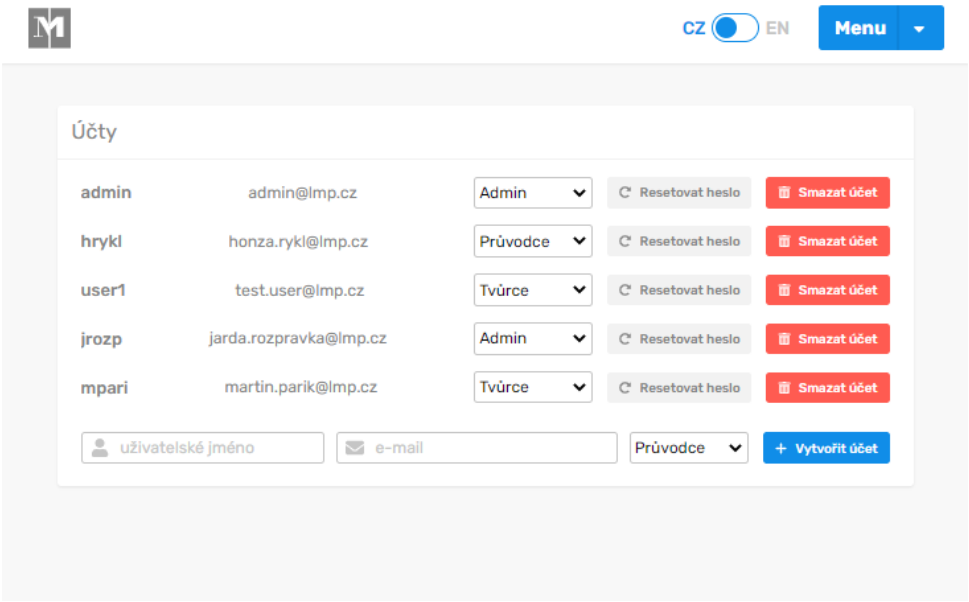


Obrázek 3.3: Home stránka

3.4 Accounts

Stránka Accounts (viz Obrázek 3.4) obsahuje jediné velké okno, ve kterém se nachází tabulka informací o uživatelských účtech. Každý řádek této tabulky zobrazuje uživatelské jméno daného účtu, přiřazenou e-mailovou adresu, drop-down menu pro změnu uživatelské role, šedé tlačítko Reset Password a červené tlačítko Delete Account. Kliknutím na tlačítko Reset Password může administrátor vygenerovat nové heslo pro vybraný účet a toto heslo bude odesláno právě na přiřazenou e-mailovou adresu. Tlačítko Delete Account pak slouží ke kompletnímu smazání uživatelského účtu, a proto je zvýrazněno červenou barvou, jelikož jde o destruktivní a nevratnou akci.

Poslední řádek tabulky pak slouží jako formulář k vytvoření nového uživatelského účtu, a obsahuje tak textová pole pro zadání uživatelského jména a e-mailové adresy a drop-down menu pro zvolení uživatelské role. Prvotní heslo tohoto účtu je vygenerováno náhodně a je odesláno na zadanou e mailovou adresu. Vytvoření účtu se pak potvrzuje modrým tlačítkem Create Account, které svojí barvou kontrastuje s tlačítkem Delete Account a vizuálně tak poslední řádek tabulky ihned odlišuje od zbytku seznamu.

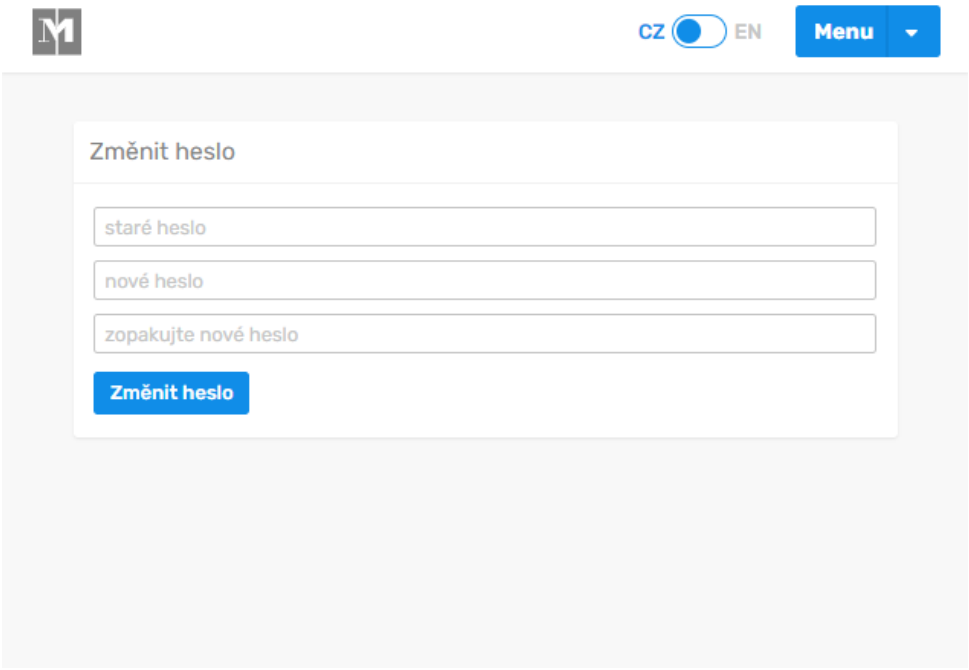


Obrázek 3.4: Accounts stránka

3.5 Settings

Tato obrazovka (viz Obrázek 3.5) slouží k nastavení osobních preferencí přihlášeného uživatele. V tuto chvíli nabízí pouze jednoduchý formulář pro

změnu uživatelského hesla. V tomto formuláři je potřeba nejprve vyplnit stávající heslo pro ověření indentity a následně dvakrát zadat nové heslo. Tato textová pole budou mít zakázáno vkládání textu ze schránky. Změna hesla bude provedena po stisknutí tlačítka Change password.



Obrázek 3.5: Settings stránka

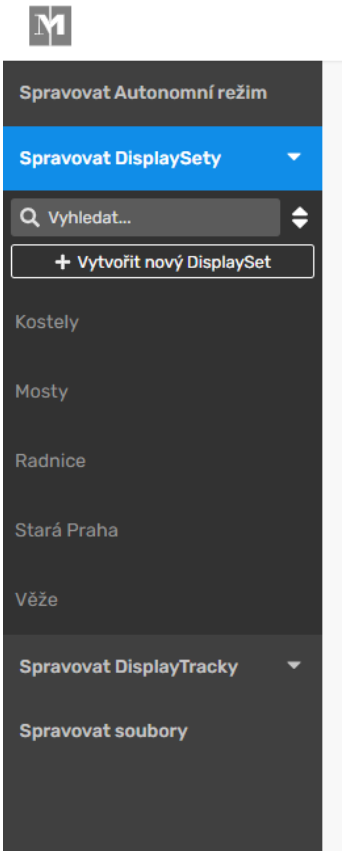
3.6 Create – Postranní panel

Tento postranní panel (viz Obrázek 3.6) je sdílený mezi všemi obrazovkami ze sekce Create a slouží k navigaci mezi nimi. Navigace mezi těmito podsekcemi záměrně není součástí hlavní navigační lišty, jelikož tento přístup pomáhá vytvořit přehlednější hierarchii mezi obrazovkami a uživatel díky tomu ví, že na tomto postranním panelu nalezne všechno, co se týká tvorby a správy multimediálního obsahu.

Postranní panel je rozdělený do 4 hlavních položek a to Manage DisplaySets, Manage DisplayTracks, Manage Files a Manage Autonomous Mode. Položky Manage Files a Manage Autonomous Mode fungují jako jednoduchá tlačítka a po kliknutí na ně dojde k jejich zvýraznění modrou barvou a navigaci do příslušné podsekce.

Položky Manage DisplaySets a Manage DisplayTracks oproti tomu fungují jako nadpis seznamu a to je indikováno šedou šipkou vedle jejich názvu. Při kliknutí na ně dojde k rozrolování příslušného seznamu, ve kterém může uži-

vatel procházet jednotlivé DisplaySety nebo DisplayTracky. Na vrchu tohoto listu se také nachází textové pole, díky kterému může uživatel vyhledávat mezi položkami podle názvu. Význam tohoto vyhledávácího pole je podpořený ikonou lupy, která je často používaným symbolem vyhledávání. Vedle tohoto textového pole leží zároveň tlačítko pro řazení seznamu (ikona dvou opačných šipek) a po kliknutí na toto tlačítko se otevře pop-up menu, ve kterém může uživatel zvolit, zda chce položky řadit podle data vytvoření, data poslední úpravy nebo podle názvu. Pod vyhledávacím polem je pak umístěno tlačítko Create DisplaySet / DisplayTrack, které při kliknutí otevře editor pro tvorbu nového DisplaySetu nebo DisplayTracku. Při kliknutí na libovolnou položku seznamu dojde k otevření editoru pro úpravu vybraného DisplayTracku nebo DisplaySetu.



Obrázek 3.6: Postranní panel Create stránky

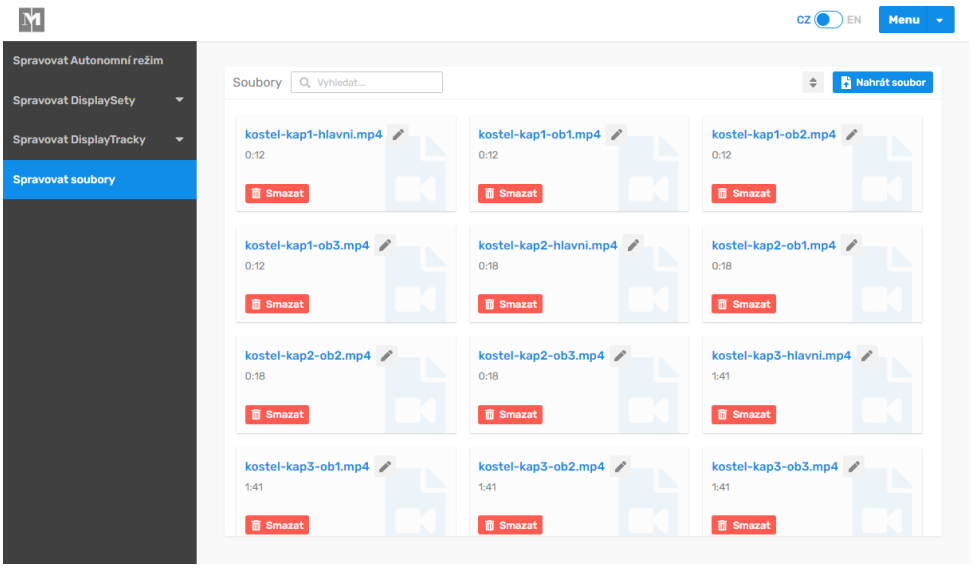
3.7 Create – Manage Files

Na této obrazovce (viz Obrázek 3.7) se nachází pouze jedno velké okno, které slouží jako průzkumník nahraných souborů. Soubory jsou znázorněny obdélníky uspořádanými do tvaru mřížky a každý z těchto obdélníků zobrazuje

název daného souboru, délku videa a velkou světle modrou ikonu, které vizuálně indikuje, zda jde o video či obrazový soubor. Kromě toho se zde pak nachází i ovládací prvky ve formě tlačítka Delete file, které slouží k odstranění souboru, a šedého tlačítko s ikonou tužky, které je umístěno vedle názvu souboru a slouží k jeho přejmenování.

Tlačítko Delete file je zvýrazněno červenou barvou, která indikuje destruktivní a nevratnou akci, a je podpořeno ještě ikonou koše, která je všeobecně známá jako symbol odstranění. Šedé tlačítko tužky při kliknutí změní název souboru na textové pole a uživatel tak dostane možnost zadat nový název.

Na horní liště průzkumníkového okna se nachází textové pole pro vyhledávání souboru podle názvu, tlačítko s ikonou šipek pro seřazení souborů a modré tlačítko Upload File pro nahrání nového souboru. Tlačítko Upload File po kliknutí otevře dialogové okno pro výběr souboru. Soubor je také možné nahrát metodou drag and drop, tak že uživatel soubor přetáhne kamkoliv do okna průzkumníku.



Obrázek 3.7: Sekce Manage Files na Create stránce

3.8 Create – Manage DisplayTracks

Okno na této stránce (viz Obrázek 3.8) reprezentuje vybraný DisplayTrack. Uvnitř tohoto okna jsou do tvaru mřížky uspořádány obdélníky, kde každý z nich reprezentuje klientské zařízení multi-projekčního systému. Tento význam je znázorněn světle modrou ikonou obrazovky. Každý z těchto obdélníků má ve svém pravém horním rohu ikonu křížku, pomocí které lze dané zařízení odstranit ze seznamu, a dále zobrazuje informaci o tom, který multimediální

soubor je přiřazený k přehrávání na tomto zařízení. Vedle něj se pak nachází tlačítko s ikonou šipky, které slouží k výběru nebo změně souboru.

Při kliknutí na toto tlačítko se na obrazovce objeví embedovaný prohlížeč souborů (viz Obrázek 3.9), který je funkcionálně identický jako prohlížeč na stránce **Create – Manage Files**, ale při kliknutí na obdélník souboru je tento embedovaný průzkumník opět skrytý a vybranému klientskému zařízení je přiřazený právě soubor, na jehož obdélník uživatel klikl. Díky tomu, že tento prohlížeč nabízí plnohodnotnou funkcionalitu, je tak možné, aby uživatel nahrál nový mutlimediální soubor teprve při tvorbě DisplayTracku a není tedy limitovaný tím, že by musel veškeré potřebné soubory nahrát striktně před začátkem tvorby.

Na horní liště okna DisplayTracku se v levém rohu nachází název tohoto DisplayTracku a vedle něj ikona tužky, která při kliknutí změní název v textové pole, kde může uživatel tento název upravit. V pravém rohu se pak nachází počítadlo, které určuje, pro kolik klientských zařízení je daný DisplayTrack určen. Pokud je tak číslo zvýšeno, tak je na konec mřížky přidán nový obdélník klientského zařízení, a pokud je číslo sníženo, tak je poslední obdélník odebrán. Ke snížení počítadla také dojde, pokud je některý obdélník odebraný pomocí své ikony křížku.

Pod celým oknem DisplayTracku se nachází dvě tlačítka. Modré tlačítko Save slouží k vytvoření nového DisplayTracku, případně k uložení provedených změn. Význam je podpořený použitím modré barvy a ikonou diskety, která je známou ikonografickou značkou pro ukládání. Tlačítko Delete se zobrazuje pouze v případě, že uživatel upravuje již existující DisplayTrack a slouží k jeho permanentnímu odstranění. Zde je opět použita ikona koše a červená barva, která oproti zbytku stránky zvýrazňuje citlivost tohoto tlačítka.

3.9 Create – Manage DisplaySets

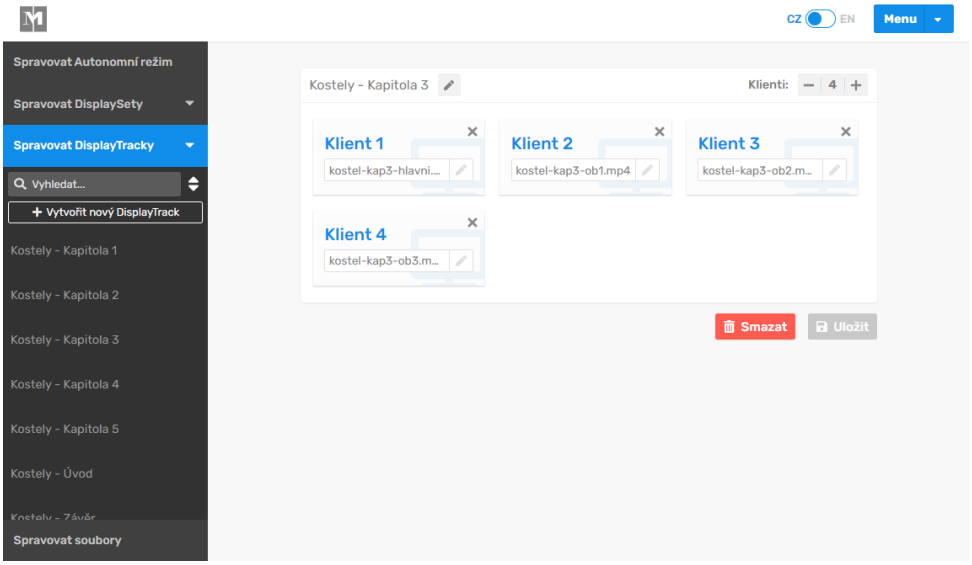
Na této stránce (viz Obrázek 3.10) se nachází jediné okno, které reprezentuje DisplaySet a na jeho horní liště je umístěný název tohoto DisplaySetu a tlačítko pro úpravu názvu, obdobně jako tomu je na obrazovce **Create – Manage DisplayTracks**. V samotném okně pak leží grafika, které zobrazuje sérii teček spojených jednou souvislou čarou. Zatímco tato čára reprezentuje časovou osu DisplaySetu, tak tečky položené na ní slouží jako vizualizace DisplayTracků umístěných na této ose. Celá tato grafika tak představuje chronologický seznam DisplayTracků uvnitř tohoto DisplaySetu a její design je inspirovaný například grafem historie commitů na stránce GitLab.

Napravo každé tečky je umístěna informační bublina, která sděluji informaci o názvu DisplayTracku a jeho celkovou délku. Dále napravo této bubliny

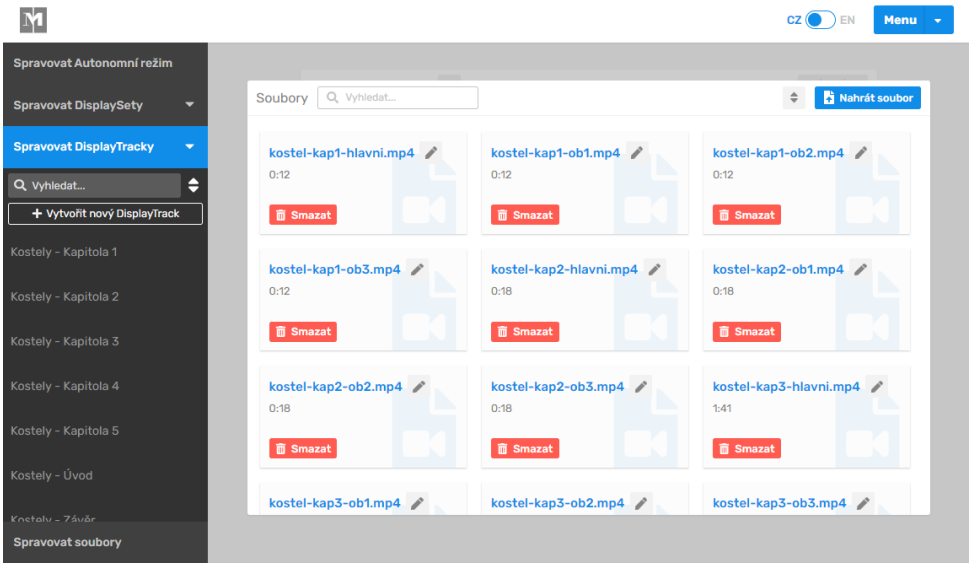
pak nalezneme dvě tlačítka. Modré tlačítko s ikonou tužky slouží pro změnu DisplayTracku a při kliknutí otevře pop-up menu, které slouží k procházení DisplayTracků. Uvnitř tohoto menu nalezneme tak nejen seznam všech DisplayTracků, ale také vyhledávácí pole pro filtraci dle názvu a také ikonu dvou šipek, pomocí které může uživatel změnit pořadí seznamu. Červené tlačítko s ikonou křížku pak slouží pro kompletní odebrání tohoto DisplayTracku z časové osy.

Poslední bod časové osy není reprezentací DisplayTracku, ale jeho bublina slouží jako tlačítko k přidání nového DisplayTracku. Tento rozdíl je zobrazen nejen modrou barvou bubliny, ale také modrým čárkovaným okrajem bodu na časové ose. Tím je dáno najevo, že tento bod zatím na časové ose neexistuje, ale je možné ho vytvořit interakcí s bublinou vedle něj, uvnitř které může uživatel vidět ikonu plusu. Při kliknutí na tuto bublinu se vedle ní objeví stejné pop-up menu jako v případě modrého tlačítka tužky.

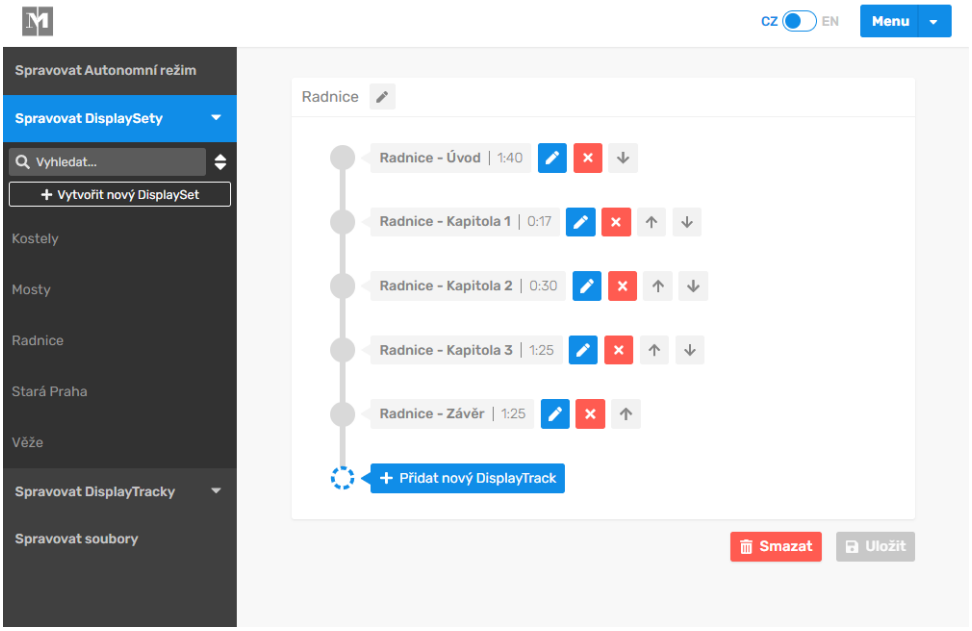
Pod celým oknem DisplaySetu se nachází tlačítka Delete a Save, která fungují identicky jako v případě obrazovky **Create – Manage DisplayTracks**.



Obrázek 3.8: Sekce Manage DisplayTracks na Create stránce



Obrázek 3.9: Embedovaný průzkumník souborů v sekci Manage DisplayTracks



Obrázek 3.10: Sekce Manage DisplaySets na Create stránce

Kapitola 4

Technologické řešení

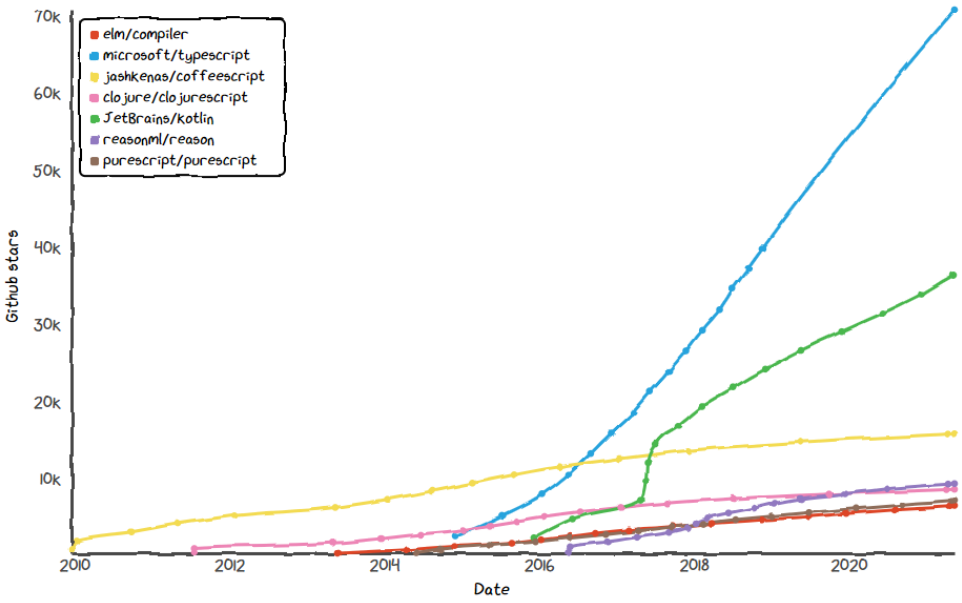
Aplikace by měla být renderovaná na straně klienta. Toto rozhodnutí vychází z toho, že zobrazovaný obsah je velmi dynamický a také z toho, že aplikace bude distribuována i v desktopové verzi, kde serverové renderování nedává smysl. Společně s tím jsem se pak rozhodl realizovat aplikaci ve formě single-page application (SPA) [3], což nabídne rychlou navigaci a potenciálně nižší duplikaci front-endového kódu, ačkoliv za cenu většího prvotního stažení. Vzhledem k tomu, že desktopová verze však bude klientskou část aplikace obsahovat už jako součást spustitelného balíčku, tak se tato volba jeví jako optimální.

4.1 Skriptovací jazyk

Jelikož je tato aplikace cílena především na webovou platformu, tak je zřejmé, že výsledný kód bude muset být napsán v jazyku JavaScript. Tento jazyk je však naneštěstí poměrně známý svým velmi slabým typovým systémem a to může často vést ke vzniku neočekávaných chyb a také to značně snižuje čitelnost a dlouhodobou udržitelnost kódu. Díky velké popularitě webu však existuje nepřeberné množství jiných jazyků, u kterých lze nastavit JavaScript jako jejich kompilační cíl, a každý z nich nabízí jiné výhody a slabosti oproti JavaScriptu. Jedna z těchto technologií však v aktuální době výrazně vyvstává nad ostatní a tou je jazyk TypeScript.

Ten se v posledních několika letech začal těšit ohromné popularitě jak u společností tak i individuálních vývojářů a v počtu GitHub stars překonává veškerou konkurenci (viz Obrázek 4.1) a to včetně jazyků, které nejsou primárně určeny ke kompilaci do JavaScriptu jako například Kotlin. Díky své masivní komunitě jde také o jazyk nejvíce podporovaný ze strany externích knihoven a frameworků a pro mě, jakožto primárně webového vývojáře, je

pak velikou výhodou i to, že se nejedná o kompletně nový jazyk s naprosto odlišnou syntaxí, ale jde především o nadstavbu JavaScriptu, která pouze rozšiřuje existující jazyk o nové funkce jako například statický typový systém nebo objektově orientované koncepty jako rozhraní či abstraktní třídy [14]. V kombinaci s tím, že balíčkový manažer npm již ve výchozím stavu nabízí typové deklarace pro většinou populárních knihoven, tak z TypeScriptu dělá dobrou volbu pro tento projekt.



Obrázek 4.1: Porovnání GitHub stars pro populární jazyky kompilované do JavaScriptu

4.2 Reaktivní framework

Použití reaktivního frameworku bylo pro mou aplikaci poměrně očividnou volbou. Nejen, že jsou tyto frameworky ideálním řešením při stavbě SPA, ale také poskytují mnohé funkce, které značně ulehčí vývoj. Nejvíce očividnou funkcí je jejich reaktivní systém, který mi dovolí psát aplikaci vysoce deklarativním způsobem, kdy se samotný framework postará o synchronizaci uživatelského rozhraní s interním stavem aplikace. Další důležitou funkcí jsou poté komponenty, které mi umožní znovupoužití mnohých UI prvků a jsou užitečnou abstrakcí pro organizaci kódu. Hlavními kandidáty v této oblasti se pak staly frameworky React, Vue.js a Svelte, jelikož jsou v dnešní době jedny z nejpoužívanějších a také s nimi mám již značné zkušenosti (např. oproti frameworku Angular), což výrazně urychlí práci na projektu.

4.2.1 React

React je v aktuální době rozhodně nejvíce populární volbou mezi reaktivními frameworky [8]. Díky tomu se může pyšnit opravdu rozsáhlou knihovnou balíčků, které jsou s ním kompatibilní a obří vývojářskou komunitou, která pro něj poskytuje velké množství tutoriálů. Další velikou výhodou je také velmi kvalitní podpora jazyku TypeScript a společně s tím i skvělé vývojářské nástroje pro editor Visual Studio Code.

Rozsáhlý ekosystém okolo této knihovny se však může rychle stát i jeho hlavní nevýhodou, jelikož nabízí několik různých řešení pro skoro každou situaci a je tak často problematické zjistit, které řešení je zrovna optimální.

Největší překážkou k použití Reactu je však jeho samotný reaktivní systém. Ten je oproti ostatním framworkům často velmi omezený a jedná se o poměrně nedokonalou abstrakci, jejíž problémy musí být kompenzovány použitím funkcí jako useMemo, useCallback nebo useRef [9]. To, v kombinaci s tím, že stav aplikace musí být vždy neměnný (immutable), tak dělá z Reactu poměrně nevhodnou volbu pro tuto aplikaci, jelikož rozsah projektu není dostatečný na to, aby se opravdu projevili benefity velkého ekosystému a React by tak představoval spíše přidanou komplexitu.

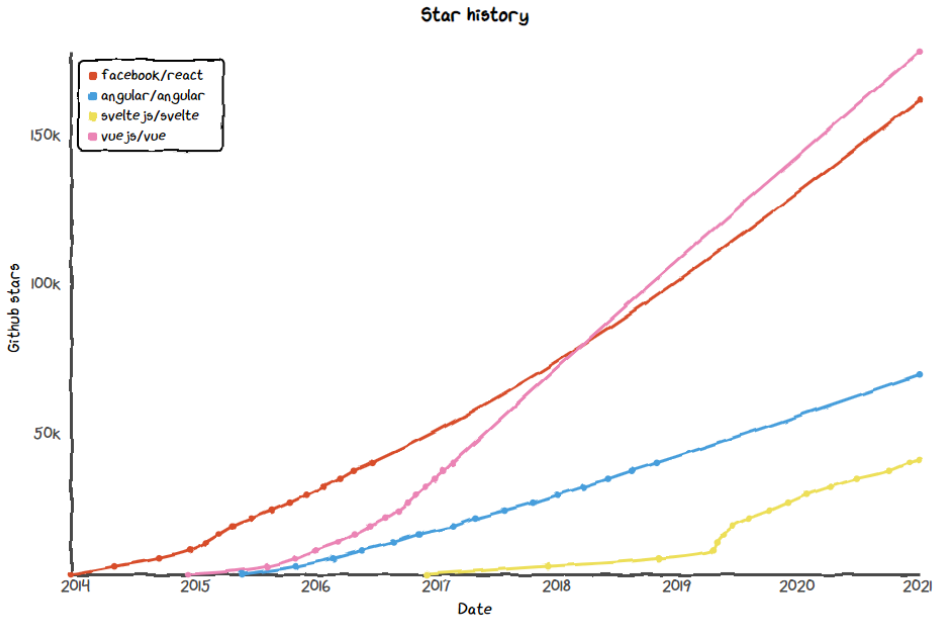
4.2.2 Vue.js

Vue.js je sice oproti Reactu méně používaný framework, ale hlavně v posledních letech si začal získávat opravdu vysokou popularitu a v momentální době má nejvíce hvězd na GitHubu ze všech reaktivních frameworků (viz Obrázek 4.2). Díky tomu, že je oproti Angularu a Reactu výrazně novější technologií, tak se ve svém designu mohl inspirovat svými předchůdci a zkombinovat tak jejich nejlepší vlastnosti.

Svým přístupem k externím knihován se podobá více monolitickému frameworku jako je Angular, než méně zaujaté knihovně jako je React, a tak sice nenabízí veškeré nástroje přímo jako součást základního balíčku, ale poskytuje několik externích knihoven jako třeba Vuex nebo Vue Router, které jsou udržované přímo oficiálním vývojářským týmem a jsou všeobecně doporučovaným řešením.

Oproti Reactu také nabízí opravdu pokročilý reaktivní systém, díky kterému lze přímo mutovat stav aplikace a také využívat vestavěných schopností jako například two-way data binding [13]. Společně s tím je skvěle integrovaný s knihovnou Vuex a je tak velmi snadné spravovat globální stav aplikace.

Hlavní problém u Vue bohužel nastává s podporou jazyku TypeScript. Ta



Obrázek 4.2: Porovnání GitHub stars pro reaktivní frameworky

byla dlouhou dobu velmi omezená a ačkoliv nová verze Vue 3.x nabízí značně vylepšenou integraci, tak podpora ze strany Vuex (verze 4.x) bohužel stále není optimální [12].

4.2.3 Svelte

Svelte je na poli reaktivních frameworků tím nejnovějším hráčem a opravdový zájem o něj se začal zvedat teprve s poměrně nedávným vydáním verze 3. S ostatními frameworky se tak nemůže rovnat v množství knihoven ani rozsahu komunity. Menší komunitu však Svelte vyrovnává velmi přehledně zpracovanou dokumentací a také svým příslibem menšího výsledného balíku (bundle size) a vyššího výkonu oproti jiným frameworkům.

Zatímco normální přístup k reaktivitě totiž stojí na technologii virtuálního DOMu [4] (VDOM), který za běhu programu porovnává verze VDOMu a pak zjišťuje, které změny aplikovat [10], tak Svelte žádný runtime ve skutečnosti nemá a slouží pouze jako kompilátor. Místo použití VDOM pak pouze sleduje, které proměnné ovlivňují které elementy na stránce a je tak schopný reagovat na změny, aniž by porovnával dvě vygenerované verze VDOMu [11].

Kromě toho se také skvěle integruje s jazykem TypeScript, má opravdu kvalitní vývojářské nástroje pro editor Visual Studio Code a nabízí i pokročilé funkce jako two-way data binding nebo global state management bez použití externí knihovny [16]. Po zvážení všech pro a proti jsem se tak nakonec

rozhodl využít při stavbě této aplikace právě framework Svelte, který by sice kvůli svému menšímu ekosystému nebyl ideální možností pro velké webové aplikace, ale na tento projekt se jeví jako ideální možnost.

4.3 Stylování

Tvorba optimálního uživatelského rozhraní je jednou z klíčových funkcí tohoto projektu a na vizuální vzhled aplikace je zde tak kladen vysoký důraz. Z toho důvodu jsem se rozhodl vytvořit veškeré vizuální komponenty ručně a nespoléhat se na již existující komponentové knihovny. I to bylo konec konců jedním z důvodů, proč jsem si mohl dovolit implementovat aplikaci ve frameworku Svelte, navzdory tomu, že oproti ostatním frameworkům nabízí výrazně menší výběr externích knihoven. Mojí osobní preferencí je pak zcela oddělovat sémantickou a stylovou stránku mých aplikací, a proto jsem se rozhodl také proti použití CSS frameworků jako např. Bootstrap nebo Tailwind.

Pro pomoc s psaním kaskádovýách stylů jsem se však rozhodl využít CSS preprocesoru a konkrétně jsem sáhl po preprocesoru SASS s využitím jeho SCSS syntaxe, která se více podobá klasickému CSS. Tato volba padla převážně na základě toho, že mám se SASS největší předchozí zkušenosti z jiných projektů. Jsem si však jistý, že i ostatní preprocesory, jako např. LESS nebo PostCSS, by byly pro tento projekt vyhovující možností. Hlavními funkcemi SASS, které jsem pak v rámci tohoto projektu využil, byly vnořené selektory, proměnné pro ukládání klíčových barev a mixiny [18].

4.4 SPA Router

Jelikož je aplikace stavěna ve formě SPA, tak nedochází ke klasickému serverovému routování, kdy prohlížeč odesílá GET požadavek pro každou URL cestu a stahuje nový dokument. Místo toho jsou veškeré cesty aplikace zabalené v jediném JavaScript balíku a vzdálený server tak slouží pouze jako poskytovatel API rozhraní. Navigace mezi obrazovkami aplikace tak probíhá čistě na straně klienta a je tedy potřeba využít knihovny, která se při navigaci postará o úpravu DOM elementů na stránce a změnu viditelné URL adresy pomocí prohlížeči poskytovaného History API [6].

Na routování ve frameworku Svelte existuje poměrně velké množství různých knihoven. Oficiální možností přímo od tvůrce Svelte by byl nadřazený framework Sapper, který jako jednu ze svých vlastností umožňuje právě i routování. Ten však bohužel v aktuální verzi nepodporuje režim klientského

routování pro SPA a nemůže být ani považován za produkčně stabilní, jelikož ještě zatím nedosáhl verze 1.0.

Mezi ostatními možnostmi (např. svelte-spa-router) jsem se nakonec rozhodl jít s knihovnou Routify. Ta nabízí kvalitní dokumentaci, vysokou flexibilitu (podpora režimu pro single-page application, server-side rendering i static site generation) a také s ní mám již předchozí zkušenosti, které mi značně usnadní práci.

4.5 Bundler

Jelikož je kód ve frameworku Svelte psaný do souborů s příponou .svelte, tak bude potřeba vytvořit build process, který tento kód zkompiluje a následně ho zabalí do jediného výstupního JavaScript souboru.

Klasickou a nejvíce populární volbou je zde pak bundler Webpack, který je výchozím bundlerem většiny frameworků. Svelte je v tomto případě však výjimkou, jelikož jeho tvůrce, Rich Harris, je zároveň také tvůrcem bundleru Rollup, a tak je jeho konfigurace obsažena v základním templatu pro Svelte. Ačkoliv je tak teoreticky možné Svelte používat s jakýmkoliv bundlerem (Webpack, Parcel, atd.), tak v tomto případě jsem se rozhodl zůstat u výchozí konfigurace, jednak proto, že to značně usnadní prvotní setup projektu, a jednak proto, že Rollup je známý svojí snadnou pochopitelností a je velmi jednoduché v něm upravovat nastavení buildu nebo dokonce psát vlastní pluginy, pokud je potřeba dosáhnout více specifické funkcionality.

Další výhodou zde je pak to, že výchozí konfigurace Svelte projektu není ukrytá za jakoukoliv abstrakcí a je tak výrazně snazší nastavení Rollupu manipulovat, než tomu je třeba u Reactu, kde je build proces defaultně skrytý za utilitou create-react-app (CRA) a pokud ho chce vývojář upravit, tak musí celý projekt takzvaně „ejectovat“, čímž se vzdá jakékoliv pomoci, kterou mu CRA nabízí.

4.6 Desktop wrapper

Jelikož má být tato aplikace distribuována současně ve formě webové a desktopové aplikace, tak bylo již od začátku jasné, že nejsnazším řešením z hlediska vývoje bude sdílet co nejvíce kódu mezi oběma platformami. Místo stavby nativní aplikace pro desktop jsem se tak rozhodl využít některého z frameworků, který mi umožní využít webové technologie při stavbě desktopové verze, díky čemuž budu moct založit obě verze na stejné sdílené codebase.

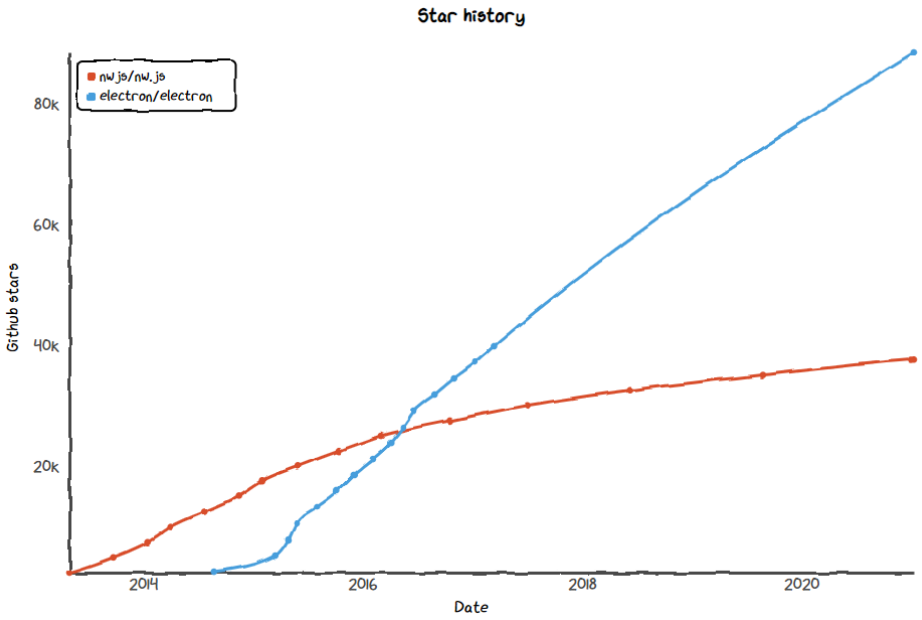
Technologie, které jsem v tomto případě uvažoval, byly Electron a NW.js. Obě tyto technologie pak slouží jako hybridní prostředí mezi prohlížečem (Chromium v případě Electronu), který se stará o samotné renderování grafického prostředí, a runtimem Node, díky kterému lze překročit hranice standardního prohlížečového sandboxu [30] a využít nativních API systému [17].

Oba tyto frameworky pak nabízejí dostatečnou funkcionalitu na to, aby byly v kontextu mé aplikace použitelné, ale já jsem se nakonec rozhodl pro využití Electronu. Toto rozhodnutí bylo informováno především tím, že jde o výrazně populárnější volbu (viz Obrázek 4.3), a nabízí tak oproti NW.js rozsáhlou komunitu vývojářů. To by mělo být přínosné hlavně při integraci s ostatními technologiemi (jako např. frameworkem Svelte), jelikož bude možné nalézt větší množství návodů a tutoriálů.

4.7 Ikonografický set

Jelikož grafický návrh aplikace využívá mnohé ikonografické znaky, které pomáhají se snadným pochopení funkcionality, tak je potřeba vybrat ikonografický set, který zajistí vizuální konzistenci designu skrz celou aplikaci. Hlavním požadavkem v tomto případě bylo, aby byly ikony dostupné ve formátu SVG, jelikož to oproti rastrovým formátům jako např. PNG nejen sníží velikost přenášených dat po síti, ale také to zaručí kompatibilitu s libovolným rozlišením obrazovky, na kterém může být aplikace zobrazena.

Po porovnání několika nejpopulárnějších balíčků nakonec volba padla na free verzi balíku Font Awesome. Ten nabízí nejen nepřeberné množství ikon, které by měli bez problémů pokrýt veškeré mé případy užití, ale také exceluje svojí webovou integrací. Kombinace s npm balíčkem fa-svelte mi umožní vkládat ikony ve formě jednoduchých Svelte komponent, a tak nebude potřeba, abych přímo manipuloval s SVG kódem daných ikon. Font Awesome také podporuje tree-shaking, takže by výsledný JavaScript bundle nikdy neměl obsahovat nadbytečné ikony [15].



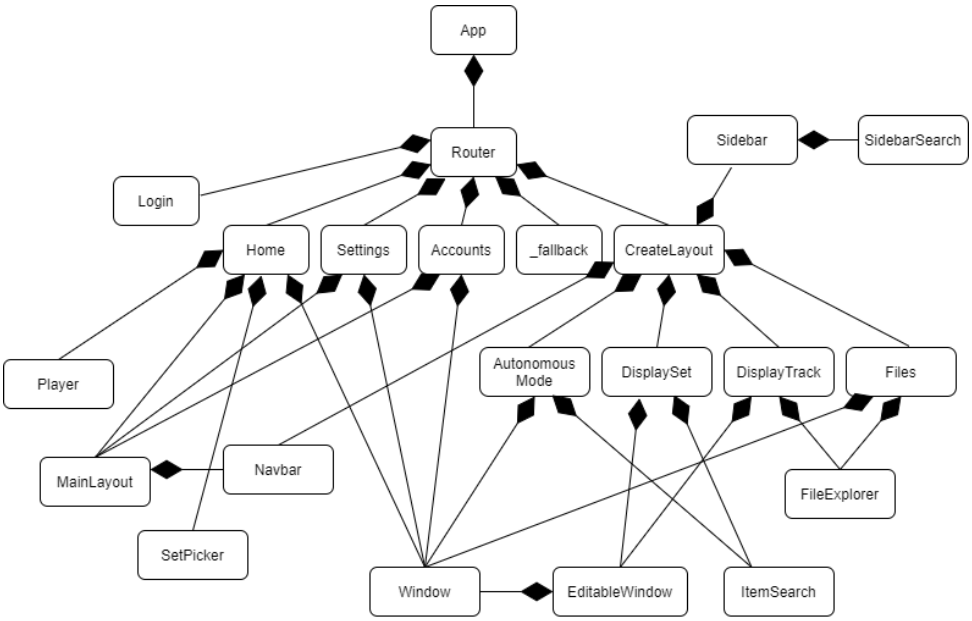
Obrázek 4.3: Porovnání GitHub stars pro Electron a NW.js

Kapitola 5

Implementace

5.1 Komponenty

Komponenty jsou základním stavebním blokem frameworku Svelte a jedná se o celky, které v rámci sebe zapouzdřují markup, styly a funkcionalitu určité části aplikace. Celkový strom komponent lze vidět na Obrázku 5.1.



Obrázek 5.1: Diagram komponentového stromu aplikace

Komponenty, které reprezentují jednotlivé stránky jsou uloženy ve složce **src/pages**, což je jejich výchozí lokace pro routovací knihovnu Routify. Hierarchie uvnitř této složky pak slouží jako popis URL cesty ke každé z těchto stránek [19]. Tyto komponenty nemají zejména žádné props [20],

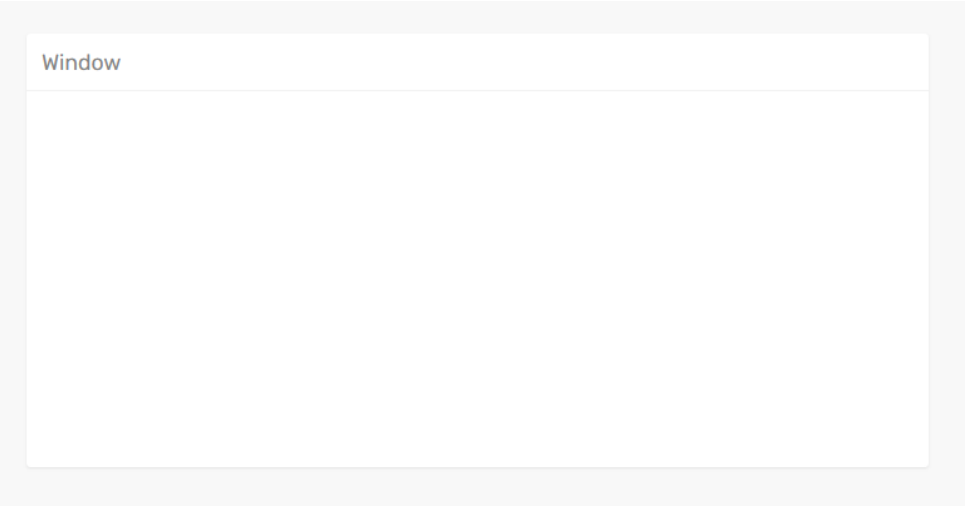
jelikož se jedná o vstupní body celé aplikace, to však s výjimkou stránek s dynamickou cestou (např. /create/set/:id), jejichž komponenty dostávají jako props své URL parametry.

Komponenty stránek jsou pak doplněné komponentami layoutů, které se nacházejí ve složce **src/layouts** a reprezentují sdílené rozhraní mezi několika stránkami. Konkrétně se zde nachází layout MainLayout, který je použit na stránkách Home, Accounts a Settings a poskytuje horní navigační lištu aplikace, a layout CreateLayout, který k ní přidává ještě postranní panel, u kterého bylo zásadní, aby zachoval svůj stav během navigace mezi stránkami v sekci Create.

Poslední kategorií komponent jsou pak znovupoužitelné komponenty, které jsou využívané mnohonásobně v rámci jednotlivých stránek. Ty se nacházejí ve složce **src/components** a každá z nich plní jinou specifickou funkci a nabízí konkrétní, s ní související props.

5.1.1 Window

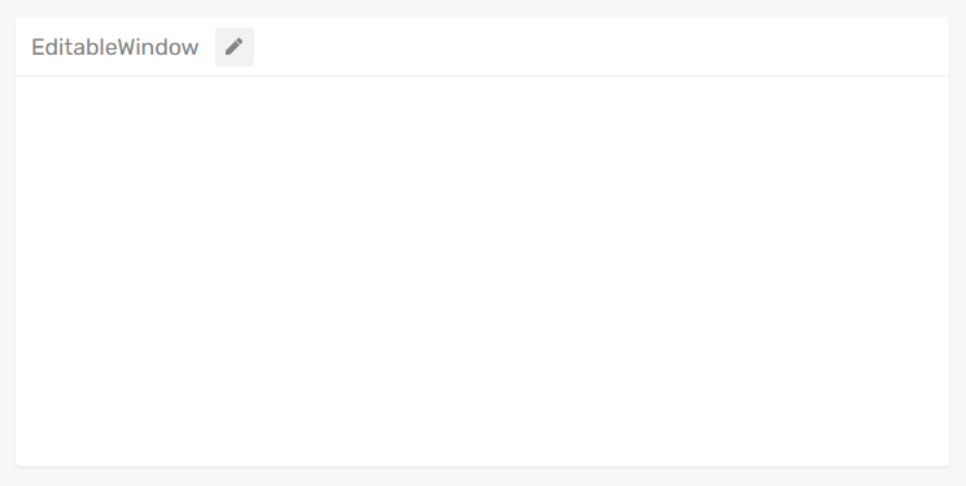
Tato komponenta představuje jeden ze základních stavebních prvků uživatelského rozhraní a jedná se o element okna, který je použit na všech stránkách aplikace (s výjimkou Login) k zapouzdření obsahu a jeho oddělení od pozadí (viz Obrázek 5.2). Tato komponenta využívá tvz. sloty [21], aby rodičovské komponenty mohli vkládat obsah nejen do samotného těla okna, ale také do jeho horního řádků.



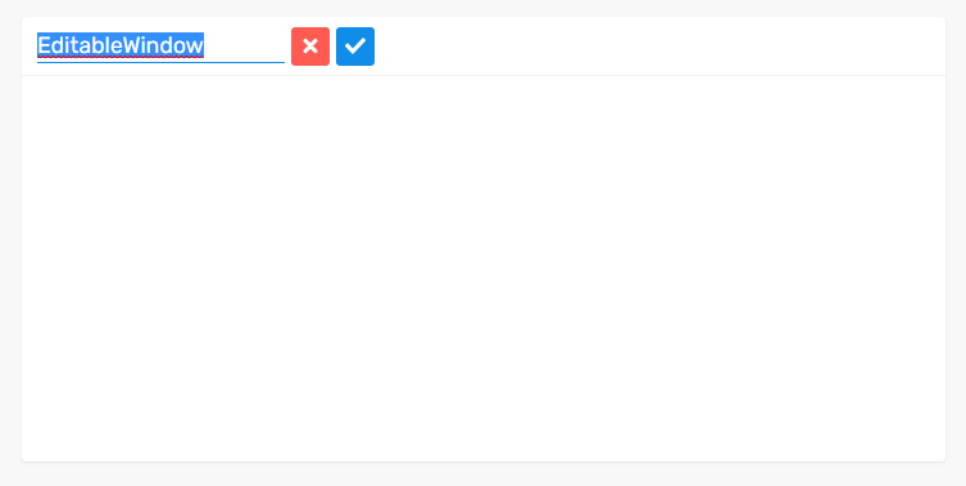
Obrázek 5.2: Komponenta Window

5.1.2 EditableWindow

Komponenta EditableWindow je přímou nadstavbou komponenty Window. Místo statického názvu v horním řádku okna však obsahuje editor, kde může uživatel aplikace název libovolně měnit (viz Obrázek 5.3 a Obrázek 5.4). Tato komponenta je využita v sekci Create u editoru DisplayTracků a editoru DisplaySetů. Horní řádek a tělo okna jsou opět rozšiřitelné pomocí slotů.



Obrázek 5.3: Komponenta EditableWindow

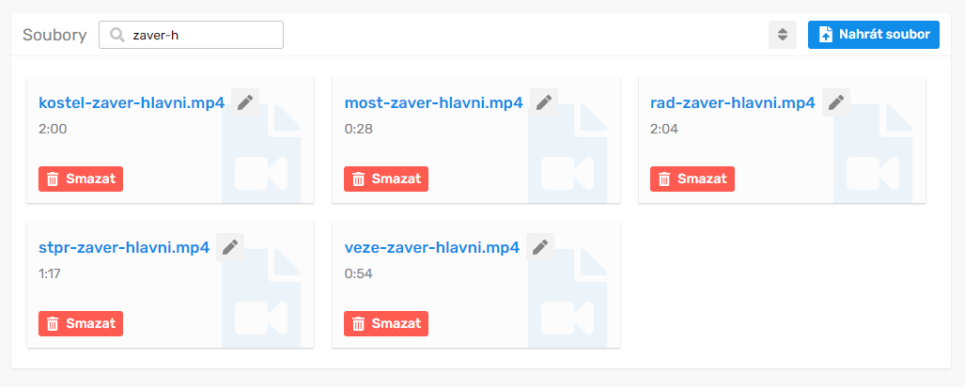


Obrázek 5.4: Komponenta EditableWindow s aktivním editorem

5.1.3 FileExplorer

Tato komponenta přestavuje prohlížeč multimediálních souborů (viz Obrázek 5.5) a byla zásadní, jelikož jsem chtěl dosáhnout toho, aby uživatel mohl spravovat soubory jak na samostatné stránce aplikace, tak i rovnou během tvorby

DisplayTracků. Díky této komponentně je tak možné využít identické rozhraní prohlížeče samostatně na stránce Manage Files i jako vnořené vyskakovací okno na stránce Manage DisplayTracks.



Obrázek 5.5: Komponenta FileExplorer

5.1.4 Navbar

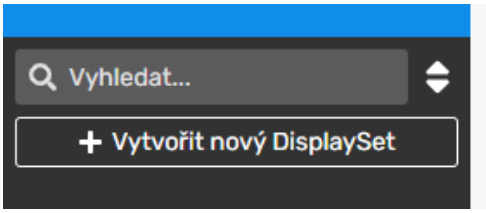
Tato komponenta obsahuje horní navigační lištu, jež je sdílena skrz všechny stránky aplikace, a to včetně souvisejícího dropdown menu (viz Obrázek 3.2). Instance této komponenty jsou využity v komponentách layoutů MainLayout a CreateLayout.

5.1.5 Sidebar

Komponenta Sidebar představuje postranní panel sdílený mezi stránkami v sekci Create (viz Obrázek 3.6) a je použita v rámci layoutu CreateLayout.

5.1.6 SidebarSearch

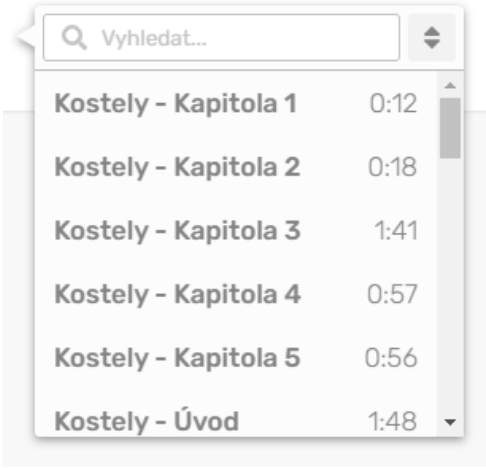
Tato komponenta reprezentuje hlavičku expandovatelných sekcí (Manage DisplayTracks, Manage DisplaySets) na postranním panelu (viz Obrázek 5.6). Navzdory svému názvu tak kromě vyhledávače také obsahuje tlačítko pro vytvoření nové položky a tlačítko pro seřazení seznamu (ačkoliv neobsahuje samotné popup menu pro výběr typu řazení).



Obrázek 5.6: Komponenta SidebarSearch

5.1.7 ItemSearch

Tato komponenta obsahuje popup menu, které je využito na stránkách Manage DisplaySets a Manage Autonomous Mode k přidání a úpravě DisplayTracku / DisplaySetu (viz Obrázek 5.7). Kromě zobrazení samotného seznamu položek nabízí také možnost vyhledávání a řazení tohoto seznamu.



Obrázek 5.7: Komponenta ItemSearch

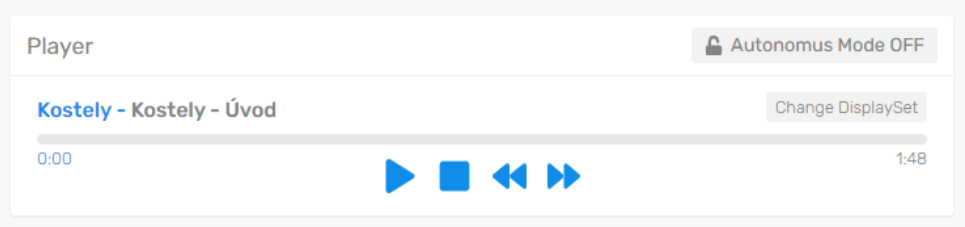
5.1.8 SetPicker

Jde o pouhou modifikaci komponenty ItemSearch, která byla upravena pro účely výběru DisplaySetu uvnitř přehrávače. V budoucí iteraci aplikace by bylo optimální více modularizovat ItemSearch, aby umožnil stejnou funkcionalitu a nedocházelo k duplikaci kódu.

5.1.9 Player

Player jako jediný nesplňuje definici znovupoužitelné komponenty a je použitý pouze na stránce Home (viz Obrázek 5.8). K jeho oddělení do samostatného

souboru došlo pouze za účelem organizace kódu.



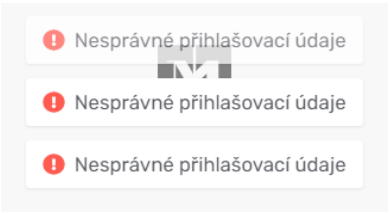
Obrázek 5.8: Komponenta Player

5.2 Sdílené widgety

Sdílené widgety jsou speciální komponentové knihovny, jejichž komponenta má v rámci celé aplikace pouze jedinou sdílenou instanci a ostatní komponenty s nimi mohou komunikovat skrz invokaci funkcí. Tyto widgety se nachází ve složce **src/lib** a jsou složené ze souboru **index.ts**, který exportuje potřebná data daného widgetu, souboru **store.ts**, který slouží jako interní uložiště reaktivního stavu pro widget, a samotné komponenty, která implementuje UI funkcionalitu.

5.2.1 Alert

Widget Alert slouží k zobrazování notifikačních zpráv aplikace a vykresluje je uprostřed u horního okraje okna (viz Obrázek 5.9). Nová zpráva může být přidána do notifikačního queue voláním fuknce `openAlert()` a každá notifikace expiruje po 1,7 sekundách. K animaci těchto notifikací je použit modul svelte/transition [22], který je součástí framworku Svelte a se stará o to, aby každá zpráva prošla animací před mountováním a unmountováním v DOM.



Obrázek 5.9: Widget Alert

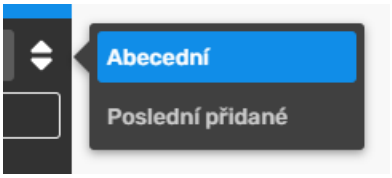
5.2.2 Modal

Tento widget je implementací klaisckého modálního okna (viz Obrázek 5.10) a je použito k vyžádání explicitního potvrzení od uživatele před provedením potenciálně kritické akce. Modal lze otevřít voláním funkce `openModal()` a widget nabízí callbacky [3] při potvrzení i zrušení akce. Na rozdíl od widgetu Alert lze u tohoto widgetu otevřít v jednu chvíli pouze jediné modálové okno.

Obrázek 5.10: Widget Modal

5.2.3 Sort Menu

Tento widget slouží jako popup menu pro řazení položek uvnitř postranního panelu v sekci Create (viz Obrázek 5.11). K otevření tohoto widgetu slouží funkce `openSortMenu()`, která jako argumenty bere HTML element, vedle kterého se popup automaticky objeví, a reaktivní proměnnou, do které widget uloží, jaký typ řazení uživatel vybral z menu.



Obrázek 5.11: Widget Sort Menu

5.3 Připojení k REST API

Stávající administrační web multiprojekčního systému z velké části spoléhá na server-side renderování a v době implementace mé aplikace tak pro většinu dotazů neexistovalo RESTové rozhraní. Jedinou alternativou tak mohlo být

scrapování [23] existujících stránek a extrakce dat ze staženého HTML dokumentu, což by však bylo velmi neefektivní a do budoucna neudržitelné řešení. V kombinaci s tím, že se moje architektura aplikace v určitých částech rozchází s existujícím backendem, jsem tak nakonec došel k rozhodnutí obsáhnout mocková data přímo v klientské aplikaci a dotazům na API se tak z velké části zcela vyhnout. Aktuální návrh aplikace však počítá s eventuální možností připojení k API a její přidání v budoucí iteraci by tak nemělo způsobovat žádné problémy. Napojení by v takovém případě mohlo probíhat 2 různými způsoby:

1. V případě nízkého až středního množství celkových dat v systému by bylo možné veškerá data vložit do stahovaného HTML dokumentu pomocí templatovacího enginu [24] nebo vytvořit jediný API endpoint pro stažení všech dat najednou (což by bylo potřeba zejména v případě desktopové aplikace). Tato možnost by byla nejsnazší na implementaci a eliminovala by veškeré další GET požadavky na API, díky čemuž by byla jakákoliv navigace v rámci aplikace instantní bez zobrazování načítacích spinnerů. Nevýhodou by byl přenos většího množství dat během prvotního načtení a také potenciální ztráta integrity dat, v případě že dojde k jejich modifikaci z jiného zařízení.

2. V případě velkého množství celkových dat by dávalo největší smysl využít lazy-loadingu a data načítat jen v případě, že jsou potřeba (např. při otevření DisplaySetu v editoru). Tento přístup by zpomalil navigaci mezi obrazovkami aplikace, vyžadoval by přidání spinnerů a celkově by byl výrazně komplexnější z hlediska implementace a cachování požadavků. Na druhou stranu by však snížil množství přenášených dat během prvotního načtení a implementace by mohla být zcela identická pro webovou i desktopovou verzi aplikace.

Jedinou výjimkou v aktuální verzi aplikace je proces přihlašování k uživatelskému účtu, který je již napojený na skutečné API skrz knihovnu Axios. V tuto chvíli probíhá autentizace skrz token ve formátu JWT [25], který je klientským JavaScriptem uložen jako cookie, což vychází z požadavků stávajícího backendu. Do budoucna bych rád přešel na řešení, kde bude cookie nastavena s headerem HttpOnly [26] na straně serveru a tělo odpovědi bude obsahovat pouze informaci o uživatelské roli přihlašovaného uživatele. Takové řešení by mělo eliminovat potenciál pro únik tokenu skrz XSS útok [27], který by byl v tomto případě zejména problematický, poněvadž platnost tokenu je v tuto chvíli nastavena na 10 dnů a neexistuje žádná metoda jeho invalidace.

5.4 Data management

Jelikož jsou data aplikace namockována (viz Sekce 5.3), tak dochází k jejich kompletnímu načtení při startu aplikace. Tato data jsou načítána ze souboru **src/mockData.json** do globálního uložiště **src/store.ts** a to je implementováno pomocí reaktivních proměnných typu Writable, které jsou součástí frameworku Svelte [16]. Tyto proměnné mohou být importovány a modifikovaný z jakéhokoliv souboru či jakékoliv komponenty aplikace a toto uložiště tak naplňuje stejnou funkci, jakou by u jiných frameworků zastávaly celé knihovny jako Redux nebo Vuex.

Vzhledem k tomu, že aplikace není připojena k API, ze kterého by mohla refetchovat data v případě modifikace, tak bylo potřeba postavit poměrně komplexní logiku k zachování integrity během mutace dat. To bylo zejména obtížné u sekce Create, kde jsou všechna data provázána a změna jediného objektu může mít kaskádující dopad. Příkladem může být třeba smazání souboru, které ovlivní všechny DisplayTracky, které tento soubor obsahují, a změny těchto DisplayTracků můžou mít dále dopad na DisplaySety a Autonomní režim. V normálním případě by se za takovéto situace dalo spoléhat na databázové kaskády, to však není možné bez připojení k backendu. Zde se tak o tuto funkcionalitu stará například utilita **src/utils/CascadeDelete.ts**. Tato implementace client-side kaskádování však bude mít pozitivní dopad i pro budoucí iterace aplikace, jelikož umožní kompletní cachování API requestů bez nutnosti jakéhokoliv refetchování.

5.5 Desktopová verze

Typická Electron aplikace se zejména skládá z 2 různých typů procesů a to main procesu [28] a renderer procesu [29]. Zatímco main proces představuje vstupní bod celé aplikace a má přístup ke všem nativním API runtimu Node, tak renderer procesy jsou spouštěné právě z main procesu a představují webová okna prohlížeče Chromium. Renderer proces na rozdíl od main procesu beží v sandboxovém prostředí [30] a nemá přímý přístup k systémovým API. Může však využívat meziprocesové komunikace [31] k tomu, aby posílal zprávy do main procesu [32], který může v reakci na ně interagovat se systémovými API.

Vzhledem k tomu, že tato aplikace byla prvotně vyvíjena ve své webové verzi, tak kód renderer procesu je z větší části identickou kopií této verze. I přes to však v určitých částech muselo dojít k úpravám specifickým pro desktopovou platformu (např. v případě práce s cookies) a proto jsem nakonec došel k rozhodnutí vytvořit pro obě verze 2 zcela nezávislé Git repozitáře.

Main proces pak obsahuje zcela nový kód, který se nachází v souborech **main.js** a **install.js**, a stará se například o vytvoření okna, spuštění renderer procesu a veškerou přímou interakci se systémem.

K produkčnímu sestavení je pak využita knihovna electron-builder, která pro celou aplikaci vytvoří instalátor ve frameworku Squirrel. Tento framework funguje na bázi toho, že během instalace / odinstalace spouští main proces se specifickým flagem a vývojář tak může provést změny v systému přímo ze svého JavaScriptu kódu. V tomto konkrétním případě jsem použil knihovnu electron-squirrel-startup k automatickému provedení typických systémových úprav (vytvoření ikony na ploše, záznam ve Windows menu atd.) a poté provedl úpravu Windows Registru z mého souboru **install.js**. Tato úprava je potřeba k přidání položky „Upload to Langweil Model Projection“ do kontextového menu souborů (viz Subsekce 5.5.1) a její implementace je inspirována open-source kódem editoru Atom, který také využívá framework Squirrel.

5.5.1 Porovnání s webovou verzí

Desktopová aplikace nabízí oproti webové verzi přídavné funkce, díky možnosti využít nativních systémových API. Těmito funkcemi jsou:

- Shození na lištu
  - Desktopovou aplikaci lze minimalizovat na ikonu na liště. Díky tomu jí lze vždy rychle a snadno znovu otevřít a také to umožní posílání notifikací, jelikož aplikace bude neustále běžet v pozadí.
- Systémové notifikace
  - Aplikace využívá systémového notifikačního systému, aby uživatele informovala o změnách stavu přehrávače. (Jelikož aplikace není v aktuální chvíli připojená k API, tak se notifikace na ukázku zobrazují, když uživatel sám spustí / zastaví přehrávání. V reálném provozu by měli být použité spíše ve chvíli, kdy stav změní někdo jiný.)
- Zapamatování uživatelských účtů
  - Aplikace si lokálně ukládá záznam o účtech, pomocí kterých proběhlo v minulosti přihlášení do systému. Následně pak může zobrazovat seznam již známých účtů a uživatel tak nemusí pokaždé zadávat své uživatelské jméno, ale pouze heslo. Tento seznam vypadá podobně jako přihlašovací obrazovka v systému Windows (viz Obrázek 5.12).

- Asociace se soubory
  - Aplikace při instalaci zaregistruje položku „Upload to Langweil Model Projection“ do kontextového menu souborů. Výběr této položky pak vybraný soubor automaticky nahraje do systému. Jedná se tak o alternativu k tlačítku Upload File na obrazovce **Create – Manage Files** a k metodě drag and drop.

Obrázek 5.12: Obrazovka Login v desktopové verzi aplikace

Kapitola 6

Testování

Aplikace byla testována pomocí uživatelských testů, do kterých se zapojilo celkem 5 účastníků s různou úrovní IT znalostí. Uživatelům bylo prezentováno 11 různých úkolů, které je měli provést všemi kritickými částmi aplikace a co nejblíže imitovat skutečné používání systému. Při testování byla použita desktopová verze aplikace a během vysvětlování funkce systému jsem se záměrně vyhnul používání klíčových slov (např. DisplaySet, DisplayTrack). Uživatelé tak testování započali jen s velmi všeobecnou představou o tom, jak multiprojekční systém funguje.

6.1 Úkoly uživatelských testů

1. Uživatel se poprvé přihlašuje do aplikace přes nový uživatelský účet, který mu byl přidělen. Po přihlášení tak chce změnit jazyk na češtinu a nastavit si své vlastní heslo místo dočasného, které dostal k prvotnímu přihlášení.
    - Tento úkol zvládli všichni uživatelé bez jakýchkoliv problémů. První okamžitou interakcí po přihlášení bylo kliknutí na jazykový přepínač a nalézt formulář pro změnu hesla na obrazovce Settings se všem také povedlo bez potíží.
2. Do muzea přišli návštěvníci a uživatel dostal za úkol jim přehrát prezentaci na téma Stará Praha.
    - Tento úkol byl jediný, u kterého uniformně narazili na problém všichni účastníci testu. Nalezení přehrávače na obrazovce Home sice nepředstavovalo žádnou komplikaci, ale problematická byla změna DisplaySetu na „Stará Praha“. To bylo způsobené jednak tím, že tlačítko „Change DisplaySet“ je poměrně nevýrazné, a

jednak proto, že uživatelé v tuto chvíli nebyli seznámeni s termínem „DisplaySet“. Prvním místem, kde tak začali změnu DisplaySetu hledat, byla sekce Timeline na obrazovce Home. Účastníkům se nakonec podařilo tlačítko najít bez jakékoliv pomoci z mojí strany, ale v budoucí iteraci aplikace by bylo optimální tlačítko „Change DisplaySet“ zvětšit a zvýraznit.

3. Návštěvníci muzea pospíchají a nemají čas sledovat celou prezentaci. Uživatel tak musí přetočit DisplaySet rovnou na Kapitolu 3.

   - Všem uživatelům se podařilo bez prodlení nalézt tlačítko na přetočení a posunout přehrávání DisplaySetu na DisplayTrack „Stará Praha – Kapitola 3“. V jednom případě se uživatel nejdříve pokusil přetočit kliknutím do sekce Timeline a v budoucí iteraci by měla být tato interakce implementována jako alternativa pro přetočení.

4. Během přehrávání se zaseklo přehrávání na zařízení Hlavní obrazovka. Uživatel tak musí toto zařízení restartovat.

   - Uživatelé bez komplikací našli zařízení v sekci Active Devices a provedli jeho restart.

5. Přehrávání prezentace skončilo. Uživatel přepne systém do režimu automatického přehrávání.

   - Všem uživatelům se podařilo spojit si zadání tohoto úkolu s výrazem „Autonomní režim“ a nedělalo jim problém nalézt přepínač v komponentně přehrávače.

6. Nadřízený nyní chce, aby uživatel sám vytvořil DisplaySet na téma „Pražský hrad“. Uživatel tak dostal přihlašovací údaje k administrátorskému účtu a video soubory pro tento DisplaySet.

   - Tento úkol byl z celého testu výrazně nejkomplexnější a zahrnoval v sobě veliké množství interakcí se systémem. Vzhledem k poměrně vágnímu zadání tohoto úkolu a uživatelovo neznalostí používaných termínů pak považuji výsledky tohoto testu za velmi pozitivní. Uživatelům se podařilo přepnout uživatelský účet bez jakýchkoliv komplikací a stejně tak velmi rychle nalezli sekci Create. Zde jim chvilku trvalo zorientovat se v celkové hierarchii aplikace, ale ve výsledku dokázali sami pochopit používanou terminologii a prošli celým procesem tvorby DisplaySetu skoro bezproblémově. Dva uživatelé začali nejprve s tvorbou DisplaySetu, než jim došlo, že musí nejprve vytvořit DisplayTracky, nicméně se nejednalo o žádnou zásadní překážku a z této chyby se dokázali rychle zotavit. Rekurentním problémem však bylo ukládání názvu DisplayTracku, jelikož tato interakce vyžaduje, aby uživatel nejprve odkliknul textové pole a poté uložil celý DisplayTrack. V několika případech uživatelé neprovedli obě tyto akce a DisplayTrack se tak uložil pod

názvem „UntitledTrack“. Svojí chybu si uživatelé sice rychle uvědomili a již jí neopakovali při tvorbě dalších DisplayTracků a při tvorbě DisplaySetu, ale i přes to bych navrhoval upravit aplikaci tak, aby se název uložil ihned při odkliknutí textového pole nebo bez odkliknutí při uložení celého DisplayTracku. Dále bych navrhoval, aby FileExplorer automaticky odscrolloval k nově nahraným soubrům, jelikož v tuto chvíli chybí jakákoliv vizuální indikace v případě alfabetického řazení souborů.

7. Kolega zapomněl své přihlašovací heslo do systému. Uživatel musí heslo tohoto účtu resetovat.

   - Tento úkol se obešel bez jakýchkoliv komplikací. Uživatelé našli obrazovku Accounts a resetovali heslo pro zadaný účet. V jediném případě uživatel nejprve navigoval na obrazovku Settings než přešel na Accounts, ale v tuto chvíli bych to považoval spíše za anomálii než skutečný problém z hlediska UI designu.

8. Nadřízený se rozhodl, že chce nově vytvořenou prezentaci „Pražský hrad“ nastavit jako druhou položku v autonomním režimu.

   - Díky předchozím znalostem z tvorby DisplaySetu byl tento úkol velmi jednoduchý pro všechny účastníky testu.

9. Jeden ze zaměstnanců dal výpověď a místo něj byl přijat nový zaměstnanec. Uživatel musí smazat starý uživatelský účet a vytvořit nový se stejnou uživatelskou rolí.

   - Uživatelé byli v tuto chvíli již seznámeni s obrazovkou Accounts a nedělalo jim tak žádný problém provést tento úkol.

10. Ukázalo se, že DisplaySet „Radnice“ je nesprávně vytvořený. Uživatel ho musí opravit a přiřadit mu správné DisplayTracky.

    - Tento úkol dokázali vyřešit všichni uživatelé bez komplikací. Zajímavým poznatkem je, že účastníci měli větší tendency nesprávné DisplayTracky smazat a poté přidat nové místo toho, aby upravili již existující položku v DisplaySetu.

11. Nastal konec pracovního dne. Uživatel se musí odhlásit a ukončit aplikaci.

    - Samotné odhlášení proběhlo bezproblémově, ale většině uživatelů nedošlo, že aplikace zůstává běžet na pozadí i po zavření okna. Nepředstavuje to zásadní problém, ale do budoucna by bylo lepší tuto možnost nastavit jako volitelnou.

Kapitola 7

Závěr

Celkově tento projekt dokázal naplnit klíčové požadavky na výslednou aplikaci, a ačkoliv uživatelské testy odhalily jisté nedostatky v oblasti uživatelského rozhraní, tak se nejedná o žádné zásadní chyby v návrhu nebo architektuře. V budoucích iteracích by tak mělo být možné vylepšit uživatelský zážitek i bez větších úprav a v tomto ohledu bych se zaměřil hlavně na sekci Create, která je z celé aplikace nejkomplexnější a v aktuální chvíli tak obsahuje nejvíce nedostatků. Pokud jde o samotnou funkcionalitu, tak si myslím, že se podařilo pokrýt kompletní rozsah administračního systému a s výjimkou připojení k produkčnímu API tak aplikace nevyžaduje další změny.

Poslední oblastí, za kterou bych se chtěl ohlédnout, je pak tvorba desktop buildu. Během uživatelských testů jsme v případě jednoho účastníka narazili na problémy s instalací, kdy instalátor skončil s chybovou hláškou a instalace neproběhla úspěšně. Po přezkoumání jsem zjistil, že se jedná o známou chybu frameworku Squirrel, která může způsobovat selhání, pokud na počítači běží další aplikace, které také Squirrel využívají (Microsoft Teams, Atom, Skype,...) [33]. Do budoucna bych tak rád přešel na instalační systém NSIS, který je také podporovaný knihovnou electron-builder a mělo by jít o více spolehlivé řešení. Společně s tím bych chtěl také integrovat auto updater, který se postará o stahování a instalaci nových verzí aplikace.

Jedinou překážkou v nasazení do produkce je tak v aktuální chvíli integrace s backendovým serverem, která by sama o sobě neměla představovat zásadní problém, pokud server poskytne RESTové rozhraní pro veškeré vyžadované funkce.

Literatura

[1] O. Trojan. *Řídicí systém pro multiprojekci, Diplomová práce*. České vysoké učení technické v Praze, 2020.

[2] T. Lowdermilk. *User-Centered Design*. O'Reilly Media, 2013.

[3] D. Flanagan. *JavaScript - The Definitive Guide*. O'Reilly Media, 2006.

[4] J. Marini. *Document Object Model: Processing Structured Documents*. McGraw Hill Professional, 2002.

[5] Hugh E. Williams, David Lane. *Web Database Applications with PHP and MySQL*. O'Reilly Media, 2004.

[6] **Původní autor:** Peter Bengtsson. *Working with the History API*. MDN Web Docs, 2020.
`https://developer.mozilla.org/en-US/docs/Web/API/History_API/Working_with_the_History_API` (15. 5. 2021)

[7] Neznámý autor. *Human Interface Guidelines – Switches*. Apple Developers, Neznámý rok.
`https://developer.apple.com/design/human-interface-guidelines/macos/buttons/switches/` (15. 5. 2021)

[8] John Potter. *NPM Trends – React vs Vue vs Angular vs Svelte*. 2006.
`https://www.npmtrends.com/react-vs-vue-vs-svelte-vs-@angular/core` (15. 5. 2021)

[9] **Původní autor:** Sophie Alpert. *Hooks API Reference*. Facebook, 2018.
`https://reactjs.org/docs/hooks-reference.html` (15. 5. 2021)

[10] **Původní autor:** Alex Krolick. *Virtual DOM and Internals*. Facebook, 2017.
`https://reactjs.org/docs/faq-internals.html` (15. 5. 2021)

[11] Rich Harris. *Svelte 3: Rethinking reactivity*. Svelte Blog; 2019. https://svelte.dev/blog/svelte-3-rethinking-reactivity (15. 5. 2021)

[12] Kia King. *The State of VueX*. Vuejs Global, 2020. https://youtu.be/ajGglyQQDOk (15. 5. 2021)

[13] **Původní autor:** Chris Fritz. *Guide - Form Input Bindings*. Vue, 2016. https://vuejs.org/v2/guide/forms.html (15. 5. 2021)

[14] Neznámý autor. *TypeScript Documentation*. Microsoft, Neznámý rok. https://www.typescriptlang.org/docs/handbook (15. 5. 2021)

[15] Neznámý autor. *Tree Shaking*. Font Awesome, Neznámý rok. https://fontawesome.com/how-to-use/javascript-api/other/tree-shaking (15. 5. 2021)

[16] Neznámý autor. *API Docs - svelte/store*. Svelte, Neznámý rok. https://svelte.dev/docs#svelte_store (15. 5. 2021)

[17] **Původní autor:** Cheng Zhao. *Quick Start Guide - Application architecture*. Electron, 2014. https://www.electronjs.org/docs/tutorial/quick-start#application-architecture (15. 5. 2021)

[18] Neznámý autor. *Documentation*. Sass, Neznámý rok. https://sass-lang.com/documentation (15. 5. 2021)

[19] **Původní autor:** Jakob Rosenberg. *Guide - Navigation*. Routify, 2020. https://routify.dev/guide/navigation (15. 5. 2021)

[20] **Původní autor:** Chris Fritz. *API - Props*. Vue, 2016. https://vuejs.org/v2/api/#props (15. 5. 2021)

[21] Neznámý autor. *API Docs - <slot>*. Svelte, Neznámý rok. https://svelte.dev/docs#slot (15. 5. 2021)

[22] Neznámý autor. *API Docs - svelte/transition*. Svelte, Neznámý rok. https://svelte.dev/docs#svelte_transition (15. 5. 2021)

[23] Michael Heydt. *Python Web Scraping Cookbook*. Packt Publishing, 2018.

[24] W. J. Gilmore. *Beginning PHP 5 and MySQL*. Apress, 2004.

[25] Neznámý autor. *Introduction to JSON Web Tokens*. Auth0, Neznámý rok. https://jwt.io/introduction (15. 5. 2021)

[26] **Původní autor:** Peter Bengtsson. *Using HTTP cookies*. MDN Web Docs, 2020. https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies (15. 5. 2021)

[27] Neznámý autor. *Mitigating Cross-site Scripting With HTTP-only Cookies*. Microsoft Developer Network, 2008. https://docs.microsoft.com/en-us/previous-versions/ /ms533046(v=vs.85) (15. 5. 2021)

[28] Zeke Sikelianos. *Docs - Glossary - main process*. Electron, 2016. https://www.electronjs.org/docs/glossary#main-process (15. 5. 2021)

[29] Zeke Sikelianos. *Docs - Glossary - renderer process*. Electron, 2016. https://www.electronjs.org/docs/glossary#renderer-process (15. 5. 2021)

[30] Ian Goldberg, David Wagner, Randi Thomas, and Eric Brewer. *A Secure Environment for Untrusted Helper Applications (Confining the Wily Hacker)*. University of California, Berkeley, 1996.

[31] **Původní autor:** Michael Satran. *Interprocess Communications*. Microsoft Developer Network, 2018. https://docs.microsoft.com/en-us/windows/win32/ipc/ interprocess-communications (15. 5. 2021)

[32] **Původní autor:** Cheng Zhao. *Docs - API - ipcRenderer*. Electron, 2014. https://www.electronjs.org/docs/api/ipc-renderer (15. 5. 2021)

[33] @SeaweedNguyen. *Failure when 2 apps run Squirrel's auto-updater/installer*. GitHub, 2017. https://github.com/Squirrel/Squirrel.Windows/issues/1138 (15. 5. 2021)

Příloha A

Sestavení aplikace

Sestavení obou verzí aplikace vyžaduje runtime prostředí Node ve verzi 15.x a package manager npm (ve výchozím stavu obsažený s instalací Node). Po otevření repozitáře je pak potřeba spustit příkaz `npm i` pro instalaci závislostí a následně příkaz `npm run build` pro sestavení produkční verze aplikace. V případě webové verze jsou výsledné soubory obsaženy ve složce **public** a mohou být staticky sdíleny z produkčního serveru. U desktopové verze dojde k sestavení instalátoru ve formátu EXE a ten lze najít ve složce **dist/squirrel-windows**. Vývojová verze aplikace může být spuštěna příkazem `npm run dev`.