Střední průmyslová škola, Ústí nad Labem, Resslova 5

Playstorm

Dokumentace k ročníkové práci

Autor: Martin Černý

Třída: 4ITA

Vedoucí práce: Bc. Tomáš Ulrich

2025/2026

Prohlášení

Prohlašuji, že jsem ročníkovou práci na téma „Playstorm“ vypracoval samostatně s použitím uvedené literatury a pramenů.

V Ústí nad Labem dne 30.12.2025

…….……..

Poděkování

Chtěl bych poděkovat Bc. Tomáši Ulrichovi za vedení mé ročníkové práce, cenné rady a odborný dohled. Děkuji také Bc. Tomáši Ulrichovi za pomoc při gramatické kontrole práce.

Anotace

Tento dokument představuje kompletní technickou dokumentaci k ročníkové práci „Playstorm“, jejímž cílem je vývoj webové aplikace pro synchronizované streamování multimediálního obsahu. Text je strukturován do dvou hlavních celků. Teoretická část obsahuje rešerši existujících platforem, jako jsou Netflix a Watch2Gether, které sloužily jako hlavní funkční a designová inspirace. Dále je zde podrobně popsán technologický stack zahrnující moderní nástroje pro frontend (React, Vite, Tailwind CSS), backend (NestJS, Go, Python) a správu dat i infrastruktury (PostgreSQL, Prisma, Docker).

Praktická část dokumentace se věnuje samotné realizaci projektu, počínaje návrhem databázového schématu (ERD) a architekturou backendového rozhraní API. Stěžejní část textu tvoří popis produktizace, kde jsou na konkrétních ukázkách zdrojového kódu vysvětleny klíčové mechanismy aplikace, jako je vyhledávání médií, autorizace streamů a zejména logika real-time synchronizace uživatelů v rámci funkce WatchParty. Dokumentaci uzavírá uživatelský popis, seznam použitých informačních zdrojů a obrazových příloh.

Klíčová slova

Kód, Media-Server, ERD, Databáze

Obsah

Obsah

Úvod

Hlavním cílem ročníkové práce je vytvořit webovou aplikaci pro streamování filmů, seriálů a anime. Klíčovou součástí projektu je implementace funkce „Watchparty“ přímo do rozhraní aplikace. Ta umožní skupině lidí sledovat video společně v reálném čase, což znamená, že všichni uvidí totéž a ovládání – jako pauza nebo přetáčení – bude synchronizované pro všechny účastníky najednou.

Téma jsem si vybral, protože mě zajímá systémový design velkých platforem jako Netflix nebo HBO. Chtěl jsem do hloubky pochopit, jak tyto systémy fungují a jakým způsobem pracují s videem ve vysokém rozlišení. Je to pro mě příležitost si v praxi vyzkoušet technologie, které stojí za moderním streamováním.

Od práce očekávám, že se naučím efektivně propojit frontend s backendem a vyřeším technické výzvy spojené se synchronizací uživatelů. Cílem je vytvořit plynulou a funkční aplikaci, která lidem nabídne kvalitní zážitek ze společného sledování.

1 Teoretická část

1.1 Rešerše

1.1.1 Netflix

Netflix je globální streamovací platforma zaměřená na online přehrávání filmů a seriálů s důrazem na jednoduché ovládání a personalizaci obsahu. Inspirací pro aplikaci PlayStorm je především přehledné uživatelské rozhraní, které klade obsah do popředí, a princip doporučování na základě chování uživatele. Důležitým prvkem je také stabilní a plynulé streamování, které zajišťuje kvalitní uživatelský zážitek.

Obrázek 1 - Netflix Logo

1.1.2 Watch2Gether

Online služba umožňující více uživatelům sledovat video obsah společně v reálném čase. Hlavní inspirací pro aplikaci PlayStorm je koncept synchronizovaného přehrávání a sdíleného zážitku na dálku. Tato služba ukazuje, jak lze propojit přehrávání videa s interakcí mezi uživateli, což je klíčový princip celé aplikace PlayStorm.

Obrázek 2 – Watch2Gether Logo

1.2 Technologie

1.2.1 React & Vite

React je moderní knihovna v JavaScriptu, která se používá pro tvorbu uživatelských rozhraní a je postavená na komponentách, díky čemuž je kód přehledný a dobře se udržuje. Pracuje s virtuálním DOM, což umožňuje rychlé a efektivní aktualizace stránky. Vite je nástroj určený pro vývoj a sestavení aplikací, který výrazně zrychluje práci díky okamžitému spuštění projektu a rychlému obnovení změn bez nutnosti zdlouhavého buildování projektu.

1.2.2 Typescript

Typescript je nadstavba jazyka Javascript která nabízí statické typování a další nástroje pro kontrolu kódu a zamezení nekonzistenci typů při práci s daty.

1.2.3 Tailwind CSS

Tailwind CSS je moderní CSS Framework který umožnuje úpravu a tvorbu CSS stylů přímo v html kódu bez nutnosti tvorby stylových souborů CSS

1.2.4 Socket.io

Socket.io je knihovna využívající webové sokety pro komunikaci v reálném čase mezi klientem a serverem. Umožňuje práci s daty mezi klientem a serverem bez nutnosti obnovování nebo znovu načítání stránky a změna dat se tedy projeví hned jakmile se provede.

1.2.5 Vidstack

Vidstack je moderní knihovna pro práci s videem na webu. Slouží jako alternativa k běžným HTML5 přehrávačům. Poskytuje již vytvořené komponenty a API pro snadnou integraci videa/streamů nebo vlastních ovládacích prvků pro aplikaci. Jeho hlavním benefitem je absolutní volnost stylizace přehrávače a nastavení argumentů pro přijímaný obsah.

1.2.6 Zustand

Zustand je knihovna pro správů stavů v aplikaci. Umožňuje ukládat a spravovat globální stavy bez složité konfigurace. Díky jednoduchému API je práce se stavem přehledná a efektivní, což z ní dělá vhodné řešení jak pro menší projekty

1.2.7 NestJS

NestJS je moderní backendový framework postavený na Node.js, který využívá TypeScript a je inspirovaný architekturou Angularu. Je založený na modulárním principu, používá dependency injection a jasně odděluje jednotlivé části aplikace, což výrazně zvyšuje přehlednost a udržovatelnost kódu. NestJS se často používá pro tvorbu REST API nebo real-time serverů, je dobře škálovatelný a vhodný jak pro menší projekty, tak i pro rozsáhlé backendové aplikace.

1.2.8 Prisma

Prisma je moderní ORM nástroj pro práci s databázemi v JavaScriptu a TypeScriptu, který výrazně zjednodušuje komunikaci mezi aplikací a databází. Umožňuje definovat databázový model pomocí přehledného schématu, ze kterého automaticky generuje typově bezpečný klient pro práci s daty. Díky tomu je kód čitelnější, méně náchylný k chybám a dobře se používá například v kombinaci s NestJS nebo Express, nejčastěji s databázemi jako PostgreSQL nebo MySQL.

1.2.9 Postgresql

PostgreSQL je výkonný open-source relační databázový systém, který se používá pro ukládání a správu dat v moderních aplikacích. Nabízí vysokou spolehlivost, bezpečnost a podporu pokročilých funkcí, jako jsou transakce, indexy nebo práce s relačními daty. Díky dobré škálovatelnosti a kompatibilitě s nástroji jako Prisma nebo backendovými frameworky typu NestJS je PostgreSQL častou volbou pro menší i rozsáhlé projekty.

1.2.10 Argon 2 & JWT

Argon2 a JWT jsou technologie používané především v oblasti autentizace a zabezpečení aplikací. Argon2 je moderní a bezpečný algoritmus pro hashování hesel, který chrání uživatelská data tím, že hesla ukládá v nevratné podobě a je odolný vůči útokům hrubou silou. JWT (JSON Web Token) slouží k bezpečnému přenosu informací o uživateli mezi klientem a serverem, nejčastěji ve formě přihlašovacího tokenu. Společně se tyto technologie používají k vytvoření bezpečného přihlašovacího systému v moderních webových aplikacích.

1.2.11 Go (Golang)

Golang (Go) je moderní programovací jazyk vyvinutý společností Google, který je navržený s důrazem na výkon, jednoduchost a efektivní práci s paralelními procesy. Nabízí přehlednou syntaxi, rychlou kompilaci a vestavěnou podporu pro práci s více vlákny pomocí gorutin. Go se často používá pro vývoj backendových služeb, API nebo serverových aplikací, kde je klíčová vysoká rychlost, stabilita a dobrá škálovatelnost.

1.2.12 FFmpeg

FFmpeg je výkonný open-source nástroj určený pro práci s multimediálním obsahem, především pro zpracování videa a audia. Umožňuje převod formátů, úpravu kvality, extrakci zvuku, práci se streamy nebo základní editaci médií pomocí příkazové řádky. FFmpeg se často používá na serverech a v automatizovaných systémech, kde je potřeba efektivně zpracovávat multimediální soubory.

1.2.13 Python

Python je vysokoúrovňový programovací jazyk známý svou jednoduchou a čitelnou syntaxí, díky které je vhodný jak pro začátečníky, tak pro pokročilé vývojáře. Používá se v širokém spektru oblastí, jako je backendový vývoj, automatizace, práce s daty nebo umělá inteligence. Python má rozsáhlý ekosystém knihoven a frameworků, což umožňuje rychlý vývoj aplikací a snadné řešení komplexních úloh.

1.2.14 Docker

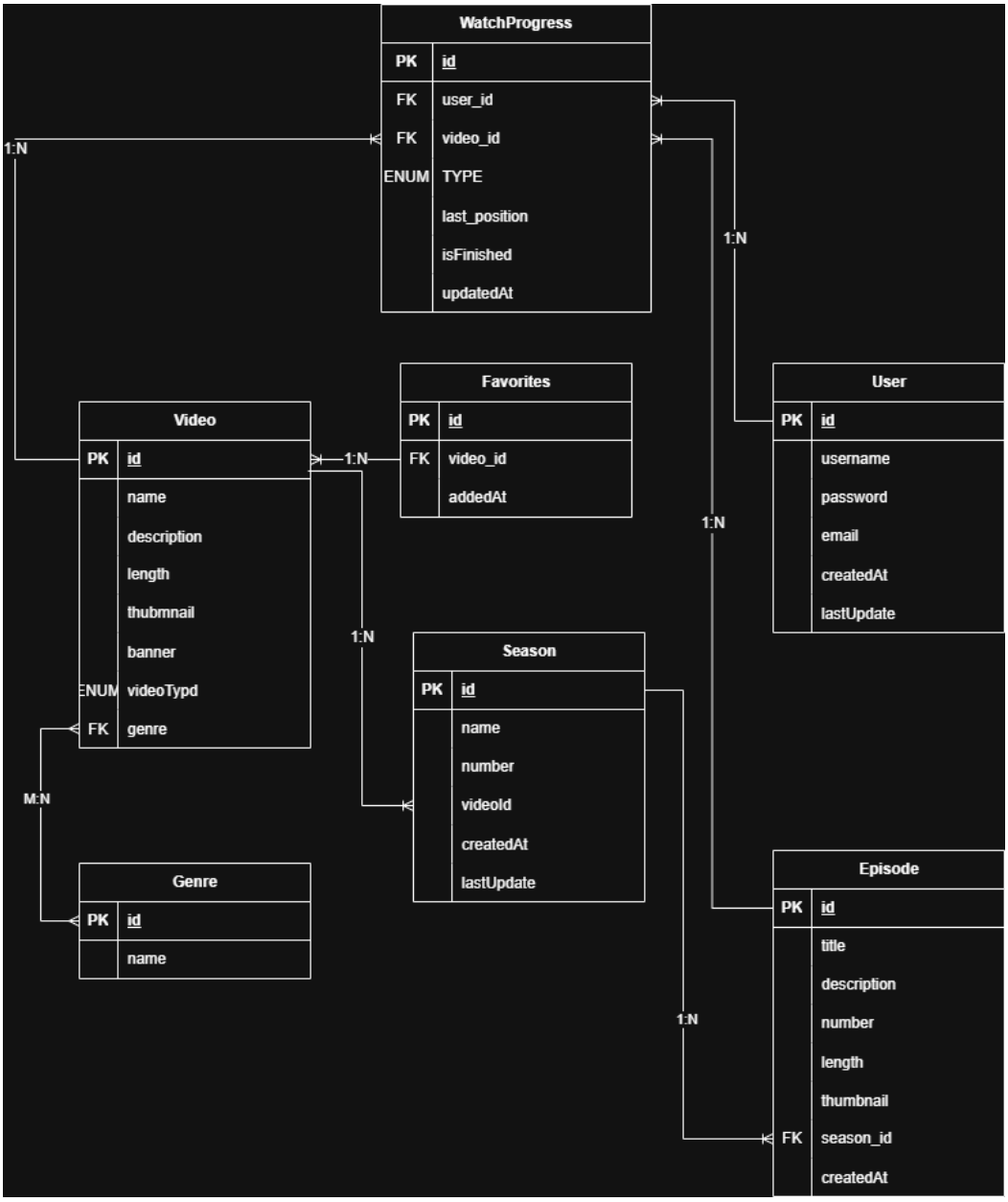
Docker je nástroj pro kontejnerizaci aplikací, který umožňuje zabalit aplikaci spolu se všemi jejími závislostmi do jednoho izolovaného prostředí zvaného kontejner. Díky tomu aplikace běží konzistentně na různých systémech bez ohledu na prostředí, ve kterém je spuštěna. Docker se běžně používá pro nasazování aplikací, zjednodušení správy serverů a vývoj moderních backendových i full-stack projektů, kde je důležitá přenositelnost a škálovatelnost.

2 Praktická část.

2.1 Návrhy

2.1.1 Návrh Databáze

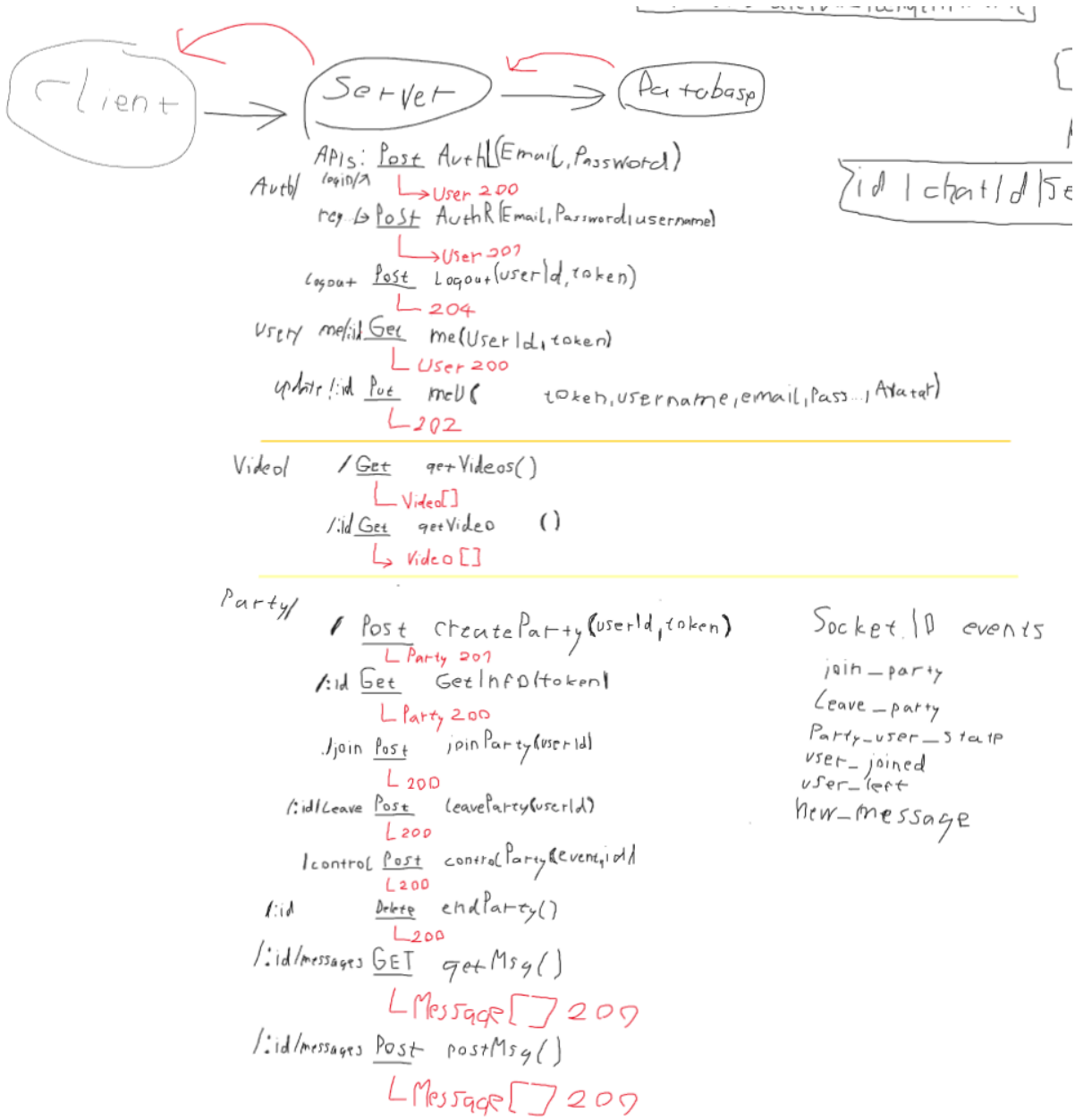
Vytvořil jsem si ERD návrh databáze aby pro mě byla tvorba schéma databáze v Prismě jednodušší a abych mohl lépe pracovat s případnými opravami a úpravami kterých bylo během tvoření projektu nemálo. Do rozložení tohoto návrhu jsem se inspiroval z různých schémat větších streamovacích platforem jako Netflix, HBO apd. Databáze odkazuje spoustu 1:N vztahů což je u těchto streamovacích služeb celkem častý jev. tento návrh mi pomohl při hledání problémů během vývoje backendu a při tvorbě návrhu pro celou restful API. V databázi jsou zvlášť řešené filmy a zvlášť seriály.



Obrázek 3 – ERD Diagram

2.1.2 Návrh backend routing a API

Zde je obrázek původního návrhu celého backendu. Je to původní návrh a během tvorby aplikace musel projít pár změnami. dost mi ale pomohl naučit se vytvářet HLD (High Level Design) a LLD (Low-Level Design) který můžeme vidět na obrázku níže. jedná se o nástřel základní architektury celého backendu a cest které na něm běží. jsou zde zvlášť tři základní cesty Autorizace, Video, a WatchParty plugin. najdeme v návrhu jak metody které používáme pro volání cesty (GET,POST,PUT,...) tak i jejich názvy.



Obrázek 4 – Navrh Backendu a API

2.2 Produktizace

2.2.3 Metoda getVideo()

Tato metoda se nachází na backendu na URL …/video/stream/:id. Tato metoda přijímá jako parametr UUID. Toto UUID se rovná některému požadovanému filmu/epizodě seriálu Tato metoda se první pokusí najít zda poskytnuté ID patří některé epizodě seriálu. Pokud ano vrátí jeho název a typ. pokud ne podívá se zda se jedná o film a případně vrátí název filmu. Tato metoda slouží jako rychlá autorizace zda film/seriál má záznam a tudíž se nachází na media serveru a lze ho načíst a spustit.

```
async getVideo(id: string) {
  try {
    const ep = await this.prisma.episode.findUnique({
      where: { id },
      include: { season: { include: { video: { select: { name: true } } } } }
    });

    if (ep) return {
      type: 'EPISODE',
      name: `${ep.season.video.name} - ${ep.title}`
    };

    const video = await this.prisma.video.findUnique({ where: { id } });

    if (!video) throw new NotFoundException("Video neexistuje");
    if (video.videotype === 'SERIES') {
      throw new BadRequestException('ID patří seriálu, nikoliv konkrétní epizodě.');
    }

    return { type: 'MOVIE', name: video.name };

  } catch (err) {
    if (err instanceof HttpException) throw err;
    throw new InternalServerErrorException(err.message);
  }
}
```

Kód 1 - getVideo()

2.2.3 Vyhledávání epizody na media serveru

Tato metoda vyhledává složku která obsahuje .m3u8 segment playlist. Složky mají strukturované jmenování (číslo epizody - název - ID). jak můžeme vidět media server načte celou složku se všemi epizodami/filmy a uloží si tento list do proměnné. Poté se podívá na každou položku z tohoto listu pomocí for loopu a vezme 3 část z názvu složky, jelikož složky mají jednotný oddělovač dat. ve třetí části se nachází id epizody/filmu toto id proovná s tim které uživatel požaduje a pokud ho najde vrátí mi .m3u8 soubor, který obsahuje list segmentů a jak na sebe navazují.

```
func findFolderByID(rootDir, id string) (string, error) {
    entries, err := os.ReadDir(rootDir)
    if err != nil {
        return "", err
    }

    for _, entry := range entries {
        if entry.IsDir() {
            parts := strings.Split(entry.Name(), " - ")
            if len(parts) == 3 && parts[2] == id {
                return entry.Name(), nil
            }
        }
    }
    return "", fmt.Errorf("folder with id %s not found", id)
}
```

Kód 2 - findFolderByID

2.2.4 Ovládání přehrávání ve WatchParty

Tato metoda obsahuje celou logiku ovládání přehrávače v pluginu WatchParty. Při vyvolání tohoto eventu pomocí socket.io Klienta si první načtu přehrávač daného uživatele, který obdržel tento event a načtu si jakou akci mám provést. Pokud přehrávač neexistuje tak vrátím chybovou hlášku. Pokud ho mám vezmu si přímou referenci na přehrávač a podívám se, zda existuje. Pokud existuje (nenastala chyba při načítáni obsahu, chyba serveru atp…) Pomocí switch statementu vyvolám určitý case. Tyto cases jsou rozděleny pomocí enumerátoru. Pokud video pouze pozastavím nebo spustím vyvolám věstavěnou funkci play/pause dle potřeby. Pokud však přetáčím video musím si první vypočítat z aktuálního času na kterém uživatel je vypočítat nový čas a poté přehrávači říct, aby se přesunul na tento čas a pokračoval v přehrávání pomocí věstavěné metody. seek ().

```
_handlePlayback_action: (data: PlayerAction) => {
    const remote = get().remote;
    const { action } = data;

    if(!remote) {
      console.error('Video element not found.');
      return;
    }

    const player = remote.getPlayer();

    if(!player){
      return
    }

    switch(action){
      case 'PLAY':
        player.remoteControl.play()
        break;
      case 'PAUSE':
        player.remoteControl.pause();
        break;
      case 'SEEK_FRW':
        const newTimeFrw = player.currentTime + 10;
        player.remoteControl.seek(newTimeFrw);
        break;
      case 'SEEK_BCK':
        const newTimeBck = player.currentTime - 10;
        player.remoteControl.seek(newTimeBck);
        break;
    }
  },
```

Kód 3 - _handlePlayback_action

2.2.5 Modal Provider

Komponenta ModalProvider nabízí vskutku zajímavý postup renderování modalů. Tato komponenta se sama nasazuje a funguje jako overlay který je permanentně kreslený přes celou stránku, ale nic nezobrazuje, pokud některý z modalu není vyvolán a nastaven jako že se má zobrazit. Pomocí zustandu je vytvořeno kontextové úložiště, které je základní struktura pro každý modal. Obsahuje základní parametry jako zda je otevřen, o jaký modal se jedná, metody pro otevření a zavření modalu. a také jaké modaly tento provider obsahuje.

```
export function ModalProvider() {
  const [isMounted, setIsMounted] = useState(false);

  useEffect(() => {
    setIsMounted(true)
  }, [])

  if(!isMounted) return null;

  return(
    <>
      <VideoModal />
      <ProfileModal />
      <PartyModal />
      <JoinPartyModal />
    </>
  )
}

type ModalType = 'video' | 'profile' | 'party' | 'joinParty' | null;

interface ModalStore {
 isOpen: boolean;
 type: ModalType;
 videoId: string | null;
 onOpen: (type: Exclude<ModalType, null>, videoId: string) => void;
 onClose: () => void;
}

export const useModal = create<ModalStore>((set) => ({
 type: null,
 isOpen: false,
 videoId: null,
 onOpen: (type, videoId) => set({isOpen: true, type, videoId}),
 onClose: () => set({type: null, isOpen: false, videoId: null}),
}))
```

Kód 4 - ModalProvider() + ModalStore

2.2.6 Dynamické přidávání nastavení

```
{Object.entries(settingsConfig).map(([key, { name, icon: Icon }]) => (
            <li
              key={key}
              onClick={() => setActiveTab(key as keyof typeof settingsConfig)}
              className={`w-full flex items-center gap-4 px-4 py-3 text-lg cursor-pointer rounded-md
transition ${
                activeTab === key
                  ? "bg-gray-700/40 text-white"
                  : "text-gray-400 hover:bg-gray-600/20"
              }`}
            >
              <Icon className="w-5 h-5" />
              <span>{name}</span>
            </li>
          ))}
```

Kód 5 - Funkce pro načtení nastavení

2.3 Popis pro uživatele

Sem napište uživatelský tutoriál.

Závěr

V aplikaci jsem dokázal zaimplementovat základní funkčnosti jako přihlášení, registraci, načtení obsahu z databáze, změna hesla, změna profilového obrázku, přehrávání obsahu a jeho ovládání pomocí synchronizačního pluginu WatchParty.

Také jsem vytvořil skript který automaticky převede celý seriál a každý díl pomocí ffmpegu a překonvertuje na .m3u8 playlist a uloží informace o epizodě do databáze. Také jsem vytvořil celý frontend a různá bezpečnostní ošetření a další věci.

Seznam použitých zdrojů

React: *React: The library for web and native user interfaces* [online]. Meta Platforms, © 2024 [cit. 2026-01-17]. Dostupné z: https://react.dev/

Vite: *Vite: Next Generation Frontend Tooling* [online]. Vitejs.dev, © 2024 [cit. 2026-01-17]. Dostupné z: https://vitejs.dev/

TypeScript: *TypeScript: JavaScript with syntax for types* [online]. Microsoft, © 2024 [cit. 2026-01-17]. Dostupné z: https://www.typescriptlang.org/

Tailwind CSS: *Tailwind CSS: Rapidly build modern websites without ever leaving your HTML* [online]. Tailwind Labs Inc., © 2024 [cit. 2026-01-17]. Dostupné z: https://tailwindcss.com/

Zustand: *Zustand: A small, fast and scalable bearbones state-management solution using simplified flux principles* [online]. GitHub, Inc., © 2024 [cit. 2026-01-17]. Dostupné z: https://github.com/pmndrs/zustand

Socket.IO: *Socket.IO: Bidirectional and low-latency communication for every platform* [online]. Socket.io, © 2024 [cit. 2026-01-17]. Dostupné z: https://socket.io/

Vidstack: *Vidstack: High-performance media player components for the web* [online]. Vidstack, © 2024 [cit. 2026-01-17]. Dostupné z: https://www.vidstack.io/

FFmpeg: *FFmpeg: A complete, cross-platform solution to record, convert and stream audio and video* [online]. Ffmpeg.org, © 2024 [cit. 2026-01-17]. Dostupné z: https://ffmpeg.org/

NestJS: *NestJS: A progressive Node.js framework for building efficient, reliable and scalable server-side applications* [online]. Nestjs.com, © 2024 [cit. 2026-01-17]. Dostupné z: https://nestjs.com/

Go (Golang): *The Go Programming Language* [online]. Google, © 2024 [cit. 2026-01-17]. Dostupné z: https://go.dev/

Python: *Python Programming Language* [online]. Python Software Foundation, © 2024 [cit. 2026-01-17]. Dostupné z: https://www.python.org/

Prisma: *Prisma: Next-generation Node.js and TypeScript ORM* [online]. Prisma Data, Inc., © 2024 [cit. 2026-01-17]. Dostupné z: https://www.prisma.io/

PostgreSQL: *PostgreSQL: The World's Most Advanced Open Source Relational Database* [online]. The PostgreSQL Global Development Group, © 2024 [cit. 2026-01-17]. Dostupné z: https://www.postgresql.org/

Argon2: BIRYUKOV, Alex, Daniel DINU a Dmitry KHOVRATOVICH. *Argon2: the memory-hard function for password hashing and other applications* [online]. GitHub, Inc., 2017 [cit. 2026-01-17]. Dostupné z: https://github.com/P-H-C/phc-winner-argon2

JWT (JSON Web Token): *JSON Web Tokens: Introduction* [online]. Auth0, © 2024 [cit. 2026-01-17]. Dostupné z: https://jwt.io/introduction/

Docker: *Docker: Accelerated Container Application Development* [online]. Docker Inc., © 2024 [cit. 2026-01-17]. Dostupné z: https://www.docker.com/

Netflix Logo: NETFLIX. *Netflix Logo* [online, obrázek]. 2024 [cit. 2026-01-17]. Dostupné z: https://brand.netflix.com/

Watch2Gether Logo: WATCH2GETHER. *Watch2Gether Logo* [online, obrázek]. 2024 [cit. 2026-01-17]. Dostupné z: https://www.watch2gether.com/

Seznam obrázků

Seznam kódů