

SZAKDOLGOZAT



MISKOLCI EGYETEM

Közösségi Videómegosztó Szoftver Tervezése és Implementálása

Készítette:

Nagy Máté

Programtervező Informatikus

Témavezető:

Fazekas Levente Áron

MISKOLC, 2024

MISKOLCI EGYETEM

Gépészmérnöki és Informatikai Kar

Alkalmazott Informatikai Intézeti Tanszék

Intézményazonosító: FI 87515

Azonosító szám: GEIAK-INZV77/2021-22-1f/BSc

SZAKDOLGOZAT FELADAT

Nagy Máté (U3ROFS) BSc programtervező informatikus jelölt részére.

A szakdolgozat tárgyköre: Webtechnológiák

A szakdolgozat címe: Közösségi Videómegosztó Szoftver Tervezése és Implementálása

A feladat részletezése:

1. A szakdolgozat célja egy valós idejű videómegosztásra alkalmas webes applikáció tervezése és implementálása.
2. Mutassa be a WebSocket és egyéb webes, valós idejű kommunikációra alkalmas API-kat!
3. Tervezzon meg és implementáljon egy szoftvert, amely képes az alábbi funkciók elvégzésére:
 - (a) Felhasználós nyilvántartása
 - (b) Videómegosztásra alkalmas szobák létrehozása, nyilvántartása
 - (c) Egyidejű vetítés azonos szobában tartózkodó felhasználók számára
 - (d) Felhasználók közötti chat
4. Mutassa be az esetleges továbbfejlesztési lehetőségeket!

Tervezésvezető: Fazekas Levente Áron (tanszéki mérnök)

A feladat kiadásának ideje: 2023.09.01

A feladat beadásának határideje: 2024.05.12 Ezt majd pontosan ki kell tölteni nyomtatás előtt

.....
tanszékvezető

1.

A szakdolgozat feladat módosítása	szükséges (módosítás külön lapon)
	nem szükséges

Miskolc,

tervezésvezető

2. A feladat kidolgozását ellenőriztem:

tervezésvezető

(1)

(2)

(3)

(4)

dátum, tervezésvezető aláírása

3. A szakdolgozat	beadható
	nem adható be

Miskolc,

konzulens

tervezésvezető

4. A szakdolgozat

..... szövegoldalt

..... db ábrát

..... db CD mellékletet

..... egyéb mellékletet tartalmaz.

5.

A szakdolgozat bírálatra	bocsátható
	nem bocsátható

A bíráló neve, címe:

Miskolc,

tanszékvezető

6. A szakdolgozat osztályzata

a témavezető javaslata:

a bíráló javaslata:

a Záróvizsga Bizottság döntése:

Miskolc,

a Záróvizsga Bizottság Elnöke

EREDETISÉGI NYILATKOZAT

Alulírott **Nagy Máté**; Neptun-kód: U3R0FS a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős Programtervező Informatikus szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírással igazolom, hogy *Közösségi Videómegosztó Szoftver Tervezése és Implementálása* című szakdolgozatom saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szó szerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc, év hó nap

.....

Hallgató

Tartalomjegyzék

1. Bevezetés	1
2. Tervezés	2
2.1. Felhasználói felület tervezése	2
2.2. Szerveroldali Logika és API-k	5
Programkódok jegyzéke	iii

1. fejezet

Bevezetés

A digitális korban élünk, ahol a technológia gyorsan fejlődik és az embereknek egyre több lehetőségük van az online interakcióra. Azonban a közösségi médiával és az egyéb digitális platformokkal eltöltött idő gyakran személytelen és egyoldalú.

A "WatchWithFriends" névre keresztelt alkalmazásom arra hivatott, hogy ezen változtasson. Az alkalmazás lehetőséget nyújt arra, hogy a felhasználók együtt nézzenek YouTube-videókat, miközben élő chaten kommunikálnak egymással. Az alkalmazásban létrehozott "szobák" révén a felhasználók egyszerűen csatlakozhatnak, és megoszthatják egymással kedvenc videóikat egy közös lejátszási listán keresztül. Az alkalmazás nem csupán egy online videómegosztó platformot egészít ki, hanem egy új közösségi élményt is kínál. Ezzel az eszközzel a barátok és családok, akár távoli földrészeken is, összehozhatók egy közös élmény kapcsán. A "WatchWithFriends" az együtt töltött idő minőségét kívánja javítani, lehetővé téve, hogy az emberek valós időben osszák meg reakcióikat, érzéseiket egy-egy videóval kapcsolatban.

Továbbá a chat funkcióval a felhasználók azonnal megvitathatják a látottakat, megoszthatják tippeiket vagy akár következő videó választási ötleteiket is.

A szoftver nem csak szórakoztató, de oktatási célokra is felhasználható. Tanárok és diákok könnyedén nézhetnek együtt oktatási anyagokat, és azonnal megbeszélhetik azok tartalmát. Ezen kívül a vállalati prezentációknál is hasznos lehet a közös videó-megtekintés funkció.

A "WatchWithFriends" egy olyan platform, ami a társasági interakciókat nem csak kiterjeszti, de egy új dimenzióba is emeli. Az alkalmazás tehát nem csak egy technológiai újítás, hanem egy közösségi eszköz is, amely összeköti az embereket függetlenül fizikai távolságuktól.

2. fejezet

Tervezés

Ebben a fejezetben a "WatchWithFriends" alkalmazás tervezési folyamatát mutatom be. A fejezet első részében a felhasználói felület tervezését ismertetem, majd a szerver és a kliens közötti kommunikáció megvalósítását.

2.1 Felhasználói felület tervezése

Az alkalmazás felhasználói felületét a React könyvtár segítségével valósítottam meg. A React egy nyílt forráskódú JavaScript könyvtár, amelyet a Facebook fejlesztett ki. A könyvtár célja, hogy a felhasználói felület fejlesztését egyszerűbbé tegye. A React egy komponens alapú könyvtár, amely lehetővé teszi a fejlesztők számára, hogy újra felhasználható komponenseket hozzanak létre, amelyeket összeállítva komplex felhasználói felületeket hozhatnak létre.

Komponensek

A React komponensek egyfajta sablonok, amelyek a felhasználói felület egy részét írják le. A komponensek egy egyszerű JavaScript objektumok, amelyeknek van egy render metódusa, amely visszaadja a komponens felhasználói felületét.

Komponensek főbb tulajdonságai:

- Paraméterek (props)
- Gyerekei (children)
- Állapotai (state)
- Életciklus metódusai (lifecycle)
- Eseménykezelői (event)
- Stílusai (style)
- Referenciák (ref)
- Kontextusai (context)

-
- Típusai (type)
 - Kulcsai (key)

Miért a React?

Sokat gondolkodtam, hogy React-ot vagy Angulart használjak az alkalmazás fejlesztéséhez. A React mellett döntöttem, mert a React egy könnyű, gyors és hatékony könyvtár. Viszont sokat Angularoztam is, és a kettő között nem sok különbséget találtam.

Külömbőség a React és az Angular között:

- **Egyszerűség és Tanulási Görbe**
 - React: Relatív egyszerűbb és könnyebben tanulható.
 - Angular: Több beépített funkció, hosszabb tanulási idő.
- **Flexibilitás és Testreszabhatóság**
 - React: Nagyon rugalmas, lehetőség más könyvtárak és eszközök beillesztésére.
 - Angular: Teljes megoldás, kevesebb szabadság de nincs szükség külső könyvtárakra.
- **Teljesítmény**
 - React: Virtuális DOM, gyakran gyorsabb nagy listák esetén.
 - Angular: Real-time Two-Way Data Binding, néha lassabb.
- **Közösség és Ökoszisztéma**
 - React: Nagyobb közösség, több harmadik féltől származó könyvtár.
 - Angular: Kevesebb, de minőségibb beépített eszköz.
- **Fejlesztői tapasztalat**
 - React: JSX és modern JavaScript funkciók.
 - Angular: TypeScript alapú, előny vagy hátrány a preferenciától függően.

Redux vagy Context API?

Felmerült bennem a kérdés, hogy a Redux vagy a Context API-t használjam az alkalmazás állapotának kezelésére. A Redux egy állapotkezelő könyvtár, amely lehetővé teszi az alkalmazás állapotának tárolását egy központi helyen. A Context API egy React API, amely lehetővé teszi az alkalmazás állapotának tárolását és megosztását a komponensek között. Nem láttam értelmét, hogy egy ilyen kis alkalmazásnál Redux-ot használjak, ezért a Context API-t választottam.

Ismertetem a Context API főbb tulajdonságait:

- **Beépített Megoldás:** A Context API része a React alapkönyvtárnak, nincs szükség külső függőségekre.

-
- **Egyszerűség:** A Context API egyszerűbb és könnyen megérthető, kevesebb boilerplate kóddal.
 - **Nincs Middleware:** Nem támogat middleware-eket alaphoz, így aszinkron állapotfrissítéseket manuálisan kell kezelni.
 - **Nem Optimalizált Nagy Alkalmazásokhoz:** Nagy alkalmazásokban hatékonytalan lehet, mert minden Context változásra újrendereli az összes fogyasztót, ha csak nem optimalizáljuk manuálisan.
 - **Kevesebb Eszköz:** Kevesebb beépített eszköze van, mint a Redux-nak, de ez nem mindig hátrány, függ a projekt igényeitől.

Példa a Context API állapotkezelésre:

```
// context
const CounterContext = React.createContext();

// state and update
const [count, setCount] = useState(0);

// state refresh
setCount(count + 1);
```

Ismertetem a Redux főbb tulajdonságait:

- **Optimalizáció:** Redux kifejezetten optimalizált a nagyobb alkalmazások számára, és lehetővé teszi az állapot frissítéseinek finom szabályozását.
- **Middleware Támogatás:** Redux lehetőséget ad middleware-ek használatára, amelyek lehetővé teszik az aszinkron állapotfrissítések könnyebb kezelését.
- **Tesztelhetőség:** A Redux architektúrája miatt a tesztelés egyszerűbb, minden action egy független egység, és a reducer funkciók tiszta függvények.
- **Eszközök és Közösség:** Redux-nak nagy közössége van, és sok kiegészítő eszköz, például a Redux DevTools.
- **Boilerplate Kód:** Több boilerplate kód szükséges, hogy elindítsunk egy Redux-alapú állapotkezelést.

Példa a Redux állapotkezelésre:

```
// action
const incrementAction = { type: 'INCREMENT' };

// reducer
function counterReducer(state = 0, action) {
  if (action.type === 'INCREMENT') {
    return state + 1;
  }
  return state;
}

// dispatch
dispatch(incrementAction);
```

TypeScript vagy JavaScript?

A TypeScript egy nyílt forráskódú, szigorúan típusos programozási nyelv, amely a JavaScriptre épül, lehetővé teszi a statikus típusok használatát, és minden érvényes JavaScript kód érvényes TypeScript kódnak is számít. Inkább a TypeScript mellett döntöttem, mert a statikus típusok használata növeli a kód minőségét és a fejlesztési sebességet, illetve csökkenti a hiba lehetőségek számát.

Összegzés

A felhasználói felület tervezéséhez a React könyvtárat használtam, a felhasználói felület állapotának kezeléséhez a Context API-t, a kód minőségének javításához a TypeScriptet.

2.2 Szerveroldali Logika és API-k

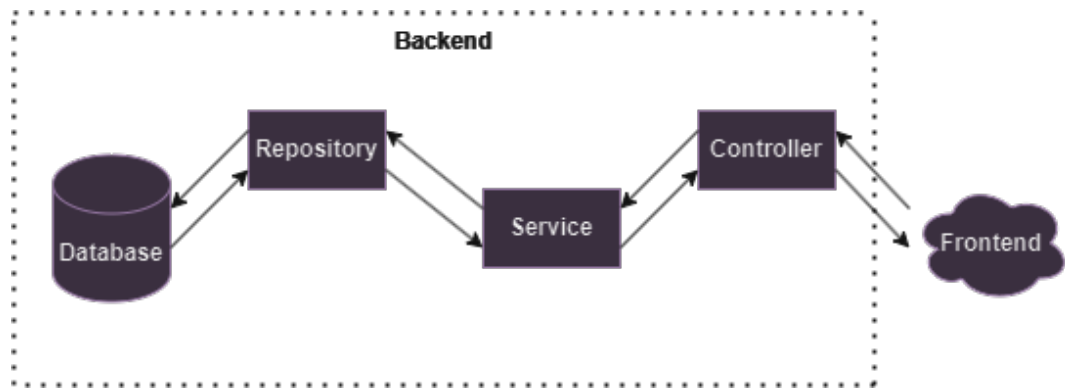
A szerver és a kliens közötti kommunikáció megvalósításához az ASP.NET Core-t használtam. Az ASP.NET Core egy nyílt forráskódú, cross-platform, magas teljesítményű keretrendszer, amelyet a Microsoft fejlesztett ki. Az ASP.NET Core-t a webalkalmazások és a webes API-k fejlesztésére használják. A környezetet a C# programozási nyelvhez tervezték, de támogatja a többi .NET nyelvet is. Én a megvalósítás során a C# nyelvet használtam. Ebben a fejezetben bemutatom a szerveroldali logikát és az API-kat.

Milyen architektúrát használtam?

Az alkalmazásnál MVC (Model-View-Controller) architektúrára építettem, illetve bővítettem a Repository és a Service rétegekkel. Az MVC architektúra egy architektúrális minta, amely a felhasználói felületet (View), az alkalmazás logikáját (Controller) és az adatokat (Model) három különálló részre osztja. A Repository réteg a Model réteghez tartozik, a Service réteg pedig a Controller réteghez. Azért bontottam tovább a rétegeket, mert így jobban elkülönülnek a felelősségi körök, és a kód is átláthatóbb lesz, illetve könnyebben bővíthető, és tesztelhető lesz.

Ismertetem a rétegek főbb tulajdonságait:

- **Adatbázis:** Az alkalmazásom adattára. Itt tárolódnak a statikus és dinamikus adatok.
- **Repositoryk:** Az adatbázis és az alkalmazás logikája közötti köztes réteg. CRUD műveletek, komplex lekérdezések.
- **Servicek:** Az üzleti logika helye. Itt történnek a komplex számítások, validációk és egyéb logikai műveletek.
- **Controllerek:** Az API végpontokat kezelő réteg. Fogadják a kliens kéréseit, és válaszokat generálnak.



Ábrák jegyzéke

Táblázatok jegyzéke

Programkódok jegyzéke

Irodalomjegyzék

CD Használati útmutató

A CD melléklet struktúrája az alábbi:

```
/
├── doc/
│   ├── chapters/
│   ├── images/
│   ├── cover/
│   ├── listings/
│   ├── plots/
│   ├── styles/
│   │   └── dolgozat.sty
│   └── tables/
├── src/
├── dolgozat.pdf
└── README.md
```

A `doc/` jegyzékben találhatóak azok a `.tex` fájlok, melyek a szakdolgozat elkészítéséhez szükségesek. Ebben a `chapters/` jegyzék tartalmazza az egyes fejezetek forrásait. A `cover/` jegyzékben vannak a borító forrásai. Az `images/` jegyzékben kaptak helyet a dolgozatban használt képek. A `listings/` jegyzékben vannak azok a forráskódok, melyek a dolgozatban is megjelennek. A `plots/` jegyzékben azon ábrák forrásai vannak, melyek \LaTeX segítségével vannak kirajzolva. A `styles/` jegyzékben vannak azok az `.sty` fájlok, melyek a dolgozat kinézetét határozzák meg. A `tables/` jegyzékben vannak a táblázatok forrásai.

A `s/` jegyzékben található az alkalmazás forráskódja.

A `dolgozat.pdf` fájl az, ami a szakdolgozatot tartalmazza pdf formátumban. Ez a fájl a \LaTeX állományok fordításának végeredménye.

A `README.md` fájl egy útmutató a CD melléklet használatához.