Baranya Vármegyei SzC Radnóti Miklós Közgazdasági Technikum

Szakma megnevezése: Szoftverfejlesztő és -tesztelő

A szakma azonosítója: 5 0613 12 03

Vizsgaremek

Tutorialbase

Készítették:

Hegedűs Adrienn

Jegenyés Bence

Vajda Gergő

Pécs

2025

Tartalomjegyzék

Projekt ismertetése

Bevezetés

Ugyan nem újdonság, hogy az interneten keressük a választ bármilyen problémánkra, viszont eddig nem találkoztunk még olyan platformmal, ami kifejezetten az ún. tutorial videók tárháza lenne. A mi platformunk egy közösségi kollaboratív profilt céloz meg, melynek az alapelve az információ megosztása, és újra-felhasználása. Ez azt jelenti, hogy a platform felhasználója egyszerre akár kétféle szerepet tölthet be: a tartalommegosztó, aki egy adott témában rendelkezik információval, tapasztalattal, és ezt szeretné megosztani, bemutatni. A másik pedig a célközönség, aki adott témában tartalmat keres, ha talál, azt megvalósítja, és azt követően beszámolhat a tartalommegosztónak saját tapasztalatáról, ezzel segítve az ő munkáját, és a többi célközönség könnyebb eligazodását. Ezzel a hozzáállással fejlődésorientált közösségeknek adhat otthont a platform, amiben a felhasználó akár egy új hobbit, szélsőségesebb esetben akár egy új életpályát is találhat magának. Mindezeken felül az integritás megőrzése végett minden információ a célnak megfelelően rendszerezve és moderálva lesz elérhető, hogy a lehető legszélesebb életkor-intervallumban megtekinthető legyen minden tartalom.

A projekt célja

A mindennapi életünkben sokszor felmerülnek problémák, ötletek, amiket bárhol is legyünk, szeretnénk megoldani, megvalósítani. Legyen szó akár otthon szerelésről, főzésről, vagy bármiről, amit két kezünkkel megtehetünk, a „Hogyan…?” kérdésekre szeretnénk választ kínálni. Ezért csapatunkkal úgy döntöttünk, hogy TutorialBase néven egy videófeltöltésekre alapuló közösségi platformot hozunk létre, ami „csináld magad” (DIY) témában nyújt értékes információkat.

Használt technológiák

Programozási nyelvek

Frontend:

- HTML
- CSS
- TypeScript
- Angular (keretrendszer)

Backend:

- PHP
- Laravel (keretrendszer)

Adatbázis

- MariaDB (SQL)

Projektmanagement és tervezési technológiák

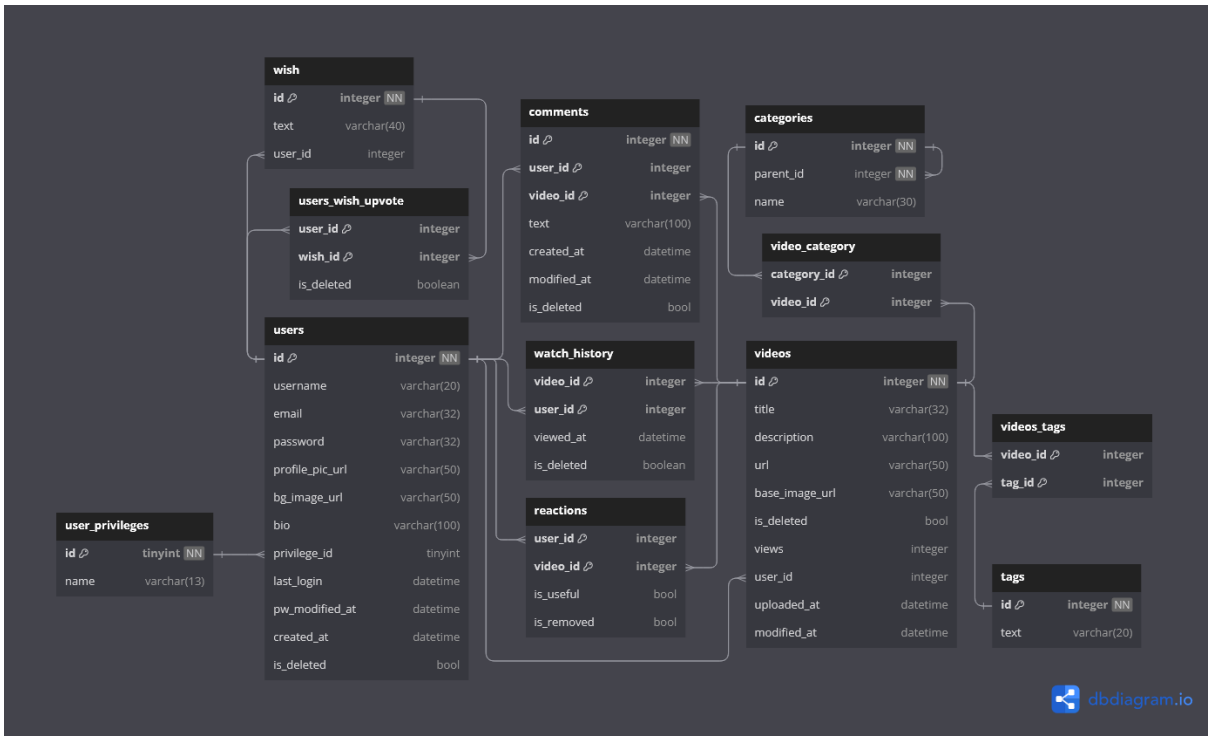
- **Jira**: Fejlesztéskövető és projektmenedzsment eszköz, amelyet az Atlassian fejlesztett ki. Eredetileg hibakövetésre (bug tracking) és feladatmenedzsmentre készült, de mára sokkal szélesebb körben használják, főleg agilis (Scrum, Kanban) csapatok munkafolyamatainak támogatására.
- **GitHub**: Online platform, amely a Git verziókezelő rendszerre épül, lehetővé téve a fejlesztők számára, hogy kódot tároljanak, kezeljenek, verziózzanak és együttműködjenek projektekben, támogatja a pull requesteket, az issue-kezelést, a kód reviewt, automatizálható CI/CD folyamatokat kínál GitHub Actions segítségével, nyílt forráskódú és privát projektekhez egyaránt használható, és mára az egyik legfontosabb központja lett a globális fejlesztői közösségnek.
- **Figma**: Modern, felhőalapú UI/UX tervezőeszköz, amely böngészőből érhető el, valós idejű együttműködést tesz lehetővé több felhasználó között, támogatja az interaktív prototípusok készítését, komponens-alapú rendszert használ a hatékonyabb dizájnhoz, könnyen integrálható más platformokkal API-n keresztül, és mára az egyik legnépszerűbb választássá vált digitális termékek tervezésére.

Adatbázis ismertetése

A projekthez tartozó adatbázis az adatokat relációs modellben tárolja. Ez azt jelenti, hogy az adatok egymáshoz rendszer-szerűen kapcsolódó táblákban vannak elhelyezve, melyek egy egyedet reprezentálnak. Az egyedek tulajdonságait pedig oszlopokban helyezzük el. Az egyedek vagy táblák kapcsolattípusát az egymáshoz viszonyított relációja határozza meg.

Mivel a backend modell-rétegje kapcsolódik közvetlen az adatbázishoz, ezért az adatbázis adatszerkezetének efféle elrendezése teszi lehetővé, hogy az objektum-orientált programozást elősegítse a fejlesztő részére: hiszen az adatbázis egy táblája egy modell-osztályt tud reprezentálni, az oszlopai pedig az osztály tulajdonságait. Ez a backend bővíthetőségét, skálázhatóságát és karbantartását segíti elő, illetve könnyíti meg.

Az alábbi képen a projekt adatbázisának adatmodell-diagramja tekinthető meg:



A diagram részletes leírása a következő oldalon található.

1. **Felhasználókezelés**: A projekt felhasználó-adatait a ***users*** nevű tábla tartalmazza. Minden felhasználó rendelkezik egy jogosultságkörrel, amit a ***user_privileges*** tábla tart számon. A lehetséges jogosultságok: Admin, Moderator, User – ezeknek azonosítójuk ugyanebben a sorrendben: 1, 2, 3.

2. **Videók**: A projekt alappillére, melynek tulajdonságai a ***videos*** táblában találhatóak. Minden (regisztrált) felhasználó tölthet fel videót, ezért kapcsolódik a felhasználó táblához.

3. **Kommentek**: A felhasználók hozzászólásokat írhatnak a videókhoz, ezért a ***comments*** tábla a felhasználók és a videók tábláját kapcsolja össze.

4. **Kategóriák**: Minden videó rendelkezik egy kategóriával, amiket a ***categories*** tábla tartalmaz. Néhány kategóriához tartozik szülőkategória, ezért a kategória azonosítóhoz kapcsolódik egy szülőkategória azonosító is. A videó és kategória táblákat a ***video_category*** tábla kapcsolja össze.

5. **Reakciók**: A felhasználók reagálhatnak videókra. Ezt a ***reactions*** nevű táblában tároljuk.

6. **Megjelölések**: Minden videót kulcsszavakkal jelölhet meg a feltöltő (max. 5). A ***tags*** táblában gyűjtjük össze a lehetséges kulcsszavakat, majd a ***videos_tags*** táblában fogjuk látni, milyen videókhoz mely kulcsszavak tartoznak.

7. **Megtekintés-előzmények**: A ***watch_history*** táblában tároljuk a felhasználó, és a megtekintett videó azonosítóját egy időbélyeggel ellátva.

Fejlesztői dokumentáció

Fejlesztési technológiák

Frontend: Visual Studio Code

Backend: JetBrains PhpStorm

Adatbázis tárolt eljárások: JetBrains DataGrip

Http protokollok

A frontend és backend közötti kommunikáció során két fajta http kérést használtunk:

- **GET**: Kifejezetten adatlekérdezésre használjuk, nem módosítja a szerveren tárolt erőforrásokat. Mivel nincs törzse (body), minden paramétere az URL-be kerül be. Nagy előnye, hogy a böngészők vagy CDN-ek könnyen tudják gyorsítótárazni a szerverről érkező választ.
- **POST**: Szerver felé használjuk adatküldésre. Ebből kifolyólag ennek a kérésnek van törzse, így a paraméterek nem az URL-ben fognak szerepelni. Ugyanaz a kérés többszöri elküldése is eredményezhet új adatot.

Backend leírása

Projektünk backend részének fejlesztéséhez a Laravel keretrendszert választottuk. Ez egy népszerű PHP keretrendszer, amely az MVC (Model-View-Controller) architektúrát követi. Egy Laravel projekt felépítése átlátható és jól strukturált, amely megkönnyíti a fejlesztést, karbantartást és bővítést.

Felépítése

Az alábbiakban a backend projekt felépítését részletezzük:

- **app/** – Az alkalmazás fő logikáját tartalmazza.
  - Http/ – Itt találhatók a kontrollerek, middleware-ek, request osztályok.
  - Models/ – Az Eloquent modellek, amelyek az adatbázissal kommunikálnak.
  - Providers/ – Szolgáltatásnyújtók (service providers), amelyek konfigurálják az alkalmazás komponenseit.
- **bootstrap/**
  - A rendszer inicializálásához szükséges fájlokat tartalmazza.
  - Az app.php állítja be az alkalmazást induláskor.
- **config/**
  - Itt találhatóak a konfigurációs fájlok.

- **database/**
    - Adatbázis-migrációk (migrations/)
    - Adatfeltöltők (seeders/)
    - Modellgyárak (factories/)
- **public/**
    - Nyilvánosan elérhető fájlok (pl. index.php, CSS, JS, képek)
    - Az index.php az alkalmazás belépési pontja.
- **resources/**
    - Nézetek (views/) – Blade sablonfájlok
    - Nyelvi fájlok (lang/)
    - Frontend erőforrások (js/, sass/)
- **routes/** – Itt definiáljuk az alkalmazás útvonalait
    - web.php – webes felülethez
    - api.php – API végpontokhoz
    - console.php –artisan parancsok
    - channels.php – valós idejű eseményekhez
- **storage/**
    - Naplók, gyorsítótár, feltöltött fájlok, ideiglenes fájlok
    - A storage/app/public tartalma szimbolikus linkelve van a public/storage alá
- **tests/**
    - PHPUnit funkcionális- és egységtesztek
- **vendor/**
    - A composer által letöltött csomagokat tartalmazza

Fontos fájlok:

- .env – Környezeti változók (adatbázis, e-mail beállítások stb.)
- artisan – Laravel parancssori eszköze
- composer.json – PHP csomagkezelő konfiguráció
- package.json – JS csomagkezelő konfiguráció (Node alapú frontend esetén)

Rétegek felépítése

A kontroller réteg feladata a frontendről érkező kérések fogadása, továbbítása más rétegek felé, majd a kérés feldolgozása után az adatok válaszként való visszaküldése.

- AuthController: A felhasználókhoz kapcsolódó műveletek irányításáért felelős. Ilyen például a ki- és bejelentkeztetés, felhasználó létrehozása, valamint adatainak lekérdezésére irányuló kérések kezelése.
- UserController: A felhasználók adatainak lekérdezésére és módósitására irányuló kéréseket szabályozza.
- VideoController: A szerveren levő videók eléréséhez kapcsolódó műveleteket kezeli, illetve az ahhoz kapcsolódó tulajdonságok lekérdezéseit (pl.: megtekintésszámláló, reakciók).
- CommentController: A videókhoz tartozó kommentek lekérdezésére, létrehozására, illetve módosítására irányuló műveleteket kezeli.
- ReactionController: A felhasználók videókra történő reakcióinak műveletét kezeli.

A modell réteg osztályai felelősek az adatbázisból érkező adatok objektummá alakításáért, mely közben ellenőrzik és biztosítják a ki- és bemenő adatok megfelelőségét, ezzel elősegítve az alkalmazásban szereplő adatok reprezentálását.

- User: A felhasználó adatmodellje.
- Video: A videók adatmodellje.
- Comment: A kommentek adatmodellje
- Reaction: A reakciók adatmodellje
- Category: A kategóriák adatmodellje
- Tag: A megjelölések adatmodellje

Frontend leírása

Könyvtárstruktúra

Projektünk átlátható és könnyen kezelhető könyvtárstruktúrával rendelkezik.

Az alapvető felépítés a következő elemekből áll:

- src: Az alkalmazás forráskódját tartalmazza.
- app: Az alkalmazás központi könyvtára, amelyben minden komponens, szolgáltatás és modell megtalálható.
- assets: Az alkalmazás statikus erőforrásait, például képek tartalmaz.
- guards: Az alkalmazás navigációs védelmét biztosító fájlokat tartalmazza. Ha a felhasználó nincs hitelesítve, automatikusan átirányításra kerül a kezdőoldalra.
- services: Az alkalmazás logikáját és az API-hívásokat kezelő szolgáltatások kapnak itt helyet, amelyek biztosítják a komponensek közötti adatáramlást.
- Interfaces: Az alkalmazásban használt típusdefiníciókat és adatmodelleket tartalmazza, ezzel segítve a típusbiztonságot és a kód átláthatóságát.
- environments: A környezeti beállításokat tartalmazó mappa, jelenleg egy development.ts fájllal, amely fejlesztési környezetre vonatkozó beállításokat (az API elérési útját) definiálja.

Komponensek

Az alkalmazás minden oldalát és funkcióját különálló komponensekre bontjuk, hogy a kód könnyen karbantartható, átlátható és újra felhasználható legyen. Az alkalmazásban vannak olyan komponensek, amelyek egy-egy teljes oldal megjelenítéséért felelnek, és kisebb, újra felhasználható komponensekből épülnek fel.

- Navbar:
  - Egy navigációs komponens, amely az alkalmazás főbb oldalai közötti gyors és egyszerű váltást biztosít, tartalmazza a logót, két lenyíló menüt (kategóriák, profil), valamint egy kereső mezőt. Dinamikusan változtatja a megjelenését attól függően, hogy a felhasználó be van-e jelentkezve: ha nem, bejelentkezés gombot jelenít meg, ha igen, akkor a profilmenüt.
- Login-modal:
  - Egy modális ablakban megjelenő bejelentkezési űrlap, amely lehetővé teszi a felhasználók számára a belépést, és az ablak alján egy link segítségével átnavigál a regisztrációs felületre, ha még nincs fiókjuk.
- Register-modal:
  - Egy modális ablakban megjelenő regisztrációs űrlap, amely új fiók létrehozását teszi lehetővé, és az ablak alján található link segítségével visszanavigál a bejelentkezési felületre.

- Home:
  - A kezdőoldal komponens, amely egy bannert és több videószekciót jelenít meg kategóriák szerint csoportosítva. Minden kategóriában vízszintesen görgethető videósorok vannak, amelyeken automatikus halványítás segíti a görgetés vizuális jelzését. A videók a VideoService-ből származnak, és VideoCardComponent-eken keresztül jelennek meg.
- Video-card:
  - Ez a komponens egy videókártyát jelenít meg, amely tartalmazza a videó borítóképét, a feltöltő profilképét, a videó címét és a feltöltő nevét. A kártyára kattintva átnavigál a videó oldalára.
- Video:
  - Ez a komponens egy videót jelenít meg, amelyet a VideoService-ből nyer API kérés segítségével. A videó tartalmazza a lejátszót, a videó információit (pl. cím, feltöltő neve, feltöltő profilképe, nézettség) és a leírást. A felhasználók reagálhatnak is a videóra.
- Comment:
  - A videóhoz tartozó kommenteket jeleníti meg, beleértve a felhasználó nevét, avatárját és a komment szövegét. Az adatokat a video-service biztosítja, amely API kérés útján tölti be a szükséges információkat a video-page oldalon.
- Add-comment:
  - A felhasználók számára lehetőséget ad arra, hogy új kommenteket adjanak hozzá egy videóhoz. A felhasználó beírja a komment szövegét, majd a komment elküldése a VideoService segítségével, API-n keresztül történik.
- Video-page:
  - A VideoPageComponent felelős egy videó teljes oldalának megjelenítéséért. A komponens betölti és megjeleníti a videó részletes információit a VideoComponent segítségével, a hozzá tartozó kommenteket a CommentComponent és AddCommentComponent segítségével, valamint az ajánlott videókat a VideoCardComponent-el. A videóhoz tartozó kommentek és az ajánlott videók az API-ból kerülnek betöltésre a VideoService segítségével.
- History-card:
  - Ez a komponens a videó előzmények kártyáját jeleníti meg, amely tartalmazza a videó címét, a feltöltő nevét, feltöltő profilképét, a feltöltés dátumát és a videó leírását. Az adatokat a VideoHistoryPageComponent biztosítja. A felhasználó a "×" gombra kattintva eltávolíthatja a videót az előzmények listájából, amely a UserService segítségével történik.
- Video-history-page:
  - Ez a komponens felelős az összes videó előzmény megjelenítéséért. Az adatokat, mint például a videókat és azok adatait, a UserService biztosítja. A komponens tartalmazza a HistoryCardComponent-eket, amelyeket dinamikusan generál, hogy megjelenítse az előzményeket.

- Video-upload:
  - Ez a komponens lehetőséget biztosít a felhasználók számára, hogy videót töltsenek fel a rendszerbe. A felhasználó feltölthet egy képet, beírhatja a videó címét, leírását, kiválaszthatja a kategóriát, alkategóriát, valamint tageket adhat a videóhoz. A videó és a kép feltöltése után a "Feltöltés" gomb segítségével a VideoService kezeli a fájlok feltöltését és az adatokat.

- Settings-page:
  - Ez a komponens lehetőséget biztosít a felhasználóknak a profiljuk beállításainak módosítására, beleértve a háttérkép, profilkép, valamint a bio frissítését. Az adatok feltöltése és módosítása a UserService segítségével történik. A felhasználók ezen kívül lehetőséget kapnak a fiókjuk törlésére is.

- Profile-page:
  - Ez a komponens a felhasználó profilját mutatja, beleértve a háttérképet, profilképet, felhasználónevet és bio-t. A felhasználó videóit a VideoService, a profil adatokat pedig a UserService biztosítja.

- Search-result-card:
  - Ez a komponens a keresés eredmények kártyáját jeleníti meg, amely tartalmazza a videó címét, a feltöltő nevét, feltöltő profilképét, a feltöltés dátumát és a videó leírását. Az adatokat a SearchResultPageComponent biztosítja.

- Search-result-page:
  - Ez a komponens a keresési eredményeket jeleníti meg, amelyeket a SearchService biztosít. Minden találat egy SearchResultCard komponensben kerül megjelenítésre, amely tartalmazza a videó alapvető információit.

- Category-page:
  - Ez a komponens a kategória oldalát jeleníti meg, ahol a videók alkategóriákra bontva, soronként jelennek meg VideoCardComponent-ek segítségével. Az adatokat a VideoService biztosítja.

Stíluslapok

A projekt stílusait CSS-ben valósítjuk meg, minden komponenshez külön stílusfájlok tartoznak, hogy a kód jól strukturált és könnyen kezelhető legyen. Az alkalmazás minden eleménél egységes vizuális megjelenésre törekszünk, miközben lehetővé tesszük a komponensek egyedi stílusainak megőrzését. A CSS kód minimalista és optimalizált, csak a szükséges szabályokat alkalmazzuk a megfelelő megjelenés eléréséhez.

A reszponzív dizájn biztosítja, hogy az alkalmazás minden eszközön jól jelenjen meg, a mobiltelefonoktól kezdve a nagyobb képernyőkig. Az alkalmazás a különböző kijelző-méretekhez automatikusan alkalmazkodik, így biztosítva a felhasználói élmény zökkenőmentességét.

Szolgáltatások

A projektben a szolgáltatások (services) kizárólag API hívások kezelésére szolgálnak, amelyek az alkalmazás adatkezelését biztosítják. Minden szolgáltatás felelős a különböző adatok lekéréséért és módosításáért külső API végpontokon keresztül. Az alkalmazás komponensei ezen szolgáltatások segítségével kommunikálnak az adatforrásokkal, mint például videók, kommentek vagy felhasználói adatok. Ez lehetővé teszi az adatkezelés és az üzleti logika elszigetelését, így a komponensek kizárólag a felhasználói felülettel és az adatmegjelenítéssel foglalkoznak.

- User-auth-service
  - A UserAuthService bejelentkezést, regisztrációt, kijelentkezést, jelszó hash-elést és hitelesítési állapot kezelését végzi API-hívásokkal, a tokeneket a localStorage-ban tárolja.
- Video-service
  - A VideoService különböző API-hívásokat valósít meg videók részleteinek, ajánlott videóknak, kategóriák és alkategóriák videóinak, felhasználói feltöltéseknek a lekérésére, valamint videók feltöltésére.
- User-service
  - A UserService a felhasználók profiladatainak kezeléséért felel: lehetővé teszi a profil és háttérkép frissítését, a bemutatkozás módosítását, a fiók törlését, valamint a megtekintett videók előzményeinek lekérését.
- Search-service
  - A SearchService a megadott keresőkifejezés alapján videókat keres az adatbázisban, és visszaadja a találatokat.
- Report-service
  - A ReportService lehetővé teszi videók és kommentek jelentését szabályszegés vagy nem megfelelő tartalom esetén.
- Admin-service
  - Az AdminService adminisztrátori jogosultsággal videók és kommentek törlésére szolgál.

Angular főbb modulok és szolgáltatások

A projekt Angular 19-ben íródott, és az Angular keretrendszer számos beépített moduljával és szolgáltatásával támogatja az alkalmazás fejlesztését. Az alábbi lista azokat a legfontosabb Angular modulokat, operátorokat és eszközöket tartalmazza, amelyeket a projektben használunk az alkalmazás felépítésében és az aszinkron logika kezelésében. Ezek a modulok és eszközök biztosítják, hogy a projekt gyorsan, hatékonyan és skálázható módon működjön.

- Angular Modulok
  - @angular/core: Az Angular alapvető modulja, amely biztosítja a komponensek, direktívák, pipe-ok és szolgáltatások működését.

- @angular/common: Az Angular alapvető eszközeit és direktíváit (pl. ngIf, ngFor) tartalmazza, amelyek széles körben alkalmazhatóak az alkalmazás különböző részeiben.
- @angular/router: Az útvonalkezelést biztosítja, lehetővé téve a komponensek dinamikus betöltését az útvonalak alapján.
- @angular/forms: Az Angular űrlapkezeléséhez szükséges modul, beleértve a reaktív és template-driven formákat.
- ReactiveFormsModule: A reaktív űrlapok kezelésére szolgáló Angular modul, amely lehetővé teszi a dinamikus validációkat és a formák deklaratív kezelését.

- RxJS és Egyéb Könyvtárak
  - fromEvent: Az RxJS egyik operátora, amely lehetővé teszi DOM eseményekből Observable adatfolyamok létrehozását.
  - Subscription: Az RxJS egyik osztálya, amely lehetővé teszi az Observable-ok figyelését, és a tőlük érkező adatfolyamok kezelését.
  - debounceTime: Az RxJS operátora, amely késlelteti az események feldolgozását.
  - rxjs: Az Angular aszinkron adatkezeléséhez használt könyvtár, amely különböző operátorokat biztosít az adatfolyamok kezelésére.
  - CryptoJS: Külső könyvtár, amely a titkosítási műveletekhez használható.

- Angular Direktívák és Egyéb Eszközök
  - RouterLink: Az Angular direktívája, amely lehetővé teszi a navigálást egy másik komponensre az útvonalak segítségével.
  - ActivatedRoute: Az aktuális útvonal paramétereinek elérésére szolgál.
  - Renderer2: Biztonságos módon manipulálja a DOM-ot, amely lehetővé teszi, hogy az alkalmazás ne közvetlenül módosítsa a DOM-ot, hanem biztonságosan dolgozzon vele.
  - HostListener: Az események kezelésére szolgál, amelyek a komponens host elemén történnek.
  - Injectable: Az Angular dekorátor, amely lehetővé teszi, hogy egy osztály szolgáltatásként injektálható legyen a konstruktorokba.

Felhasználói dokumentáció

Az oldalunkat három különböző felhasználói típus igényeihez igazítva alakítottuk ki:

- Bejelentkezés nélküli felhasználó
- Bejelentkezett felhasználó
- Adminisztrátor

Bejelentkezés nélküli felhasználó

Az alkalmazást úgy terveztük, hogy bejelentkezés nélküli felhasználók is képesek legyenek hozzáférni az alapvető funkciókhoz, regisztráció nélkül. Ugyanakkor a bejelentkezés nélküli használat esetén bizonyos funkciók nem állnak rendelkezésre.

Az így elérhető funkciók:

- Videók és kommentek megtekintése
- Regisztrált felhasználók profiljának megtekintése
- Keresés

Bejelentkezett felhasználó

A bejelentkezett felhasználók teljes hozzáférést kapnak az alkalmazás minden funkciójához. Bejelentkezést követően az alábbi további lehetőségek válnak elérhetővé számukra:

- Videók feltöltése
- Videókra adott reakciók
- Kommentelés
- Videók vagy kommentek jelentése
- Saját profil megtekintése és testreszabása
- Megtekintési előzmények nyomon követése
- Fiók törlésének lehetősége

Adminisztrátor

Az adminisztrátorok számára a következő extra funkciók állnak rendelkezésre:

- Videók és kommentek törlése

Csapatmunka

Scrum

1. **Kis léptékű fejlesztési szakaszok** A Scrum módszertan által biztosított rövid, általában 2–4 hetes szakaszok (sprintek) lehetővé tették, hogy az új funkcionalitásokat rendszeresen kézhez kapjuk. Ezáltal időben elemezhettük a teljesítményt, és gyors visszajelzést kaptunk a csapattól és az érintettektől.
2. **Hatékony együttműködés és nyitottság** A Scrum keretrendszer hangsúlyozta az erős csapatmunkát és az átláthatóságot. Az állandó napi stand-up megbeszélések és a folyamatos backlog kezelés segített mindenkit naprakészen tartani, egyértelmű célokat tűzve ki a csapat számára.
3. **Gyors alkalmazkodás a változásokhoz** Az iteratív megközelítés révén könnyen reagálhattunk a projekt során felmerülő váratlan igényekre vagy kihívásokra. A sprint végi retrospektívák segítettek finomhangolni a folyamatokat, és lehetőséget nyújtottak a folyamatos fejlődésre mind technikai, mind szervezeti szinten.

Szerepek

Jegenyés Bence feladatai és felelősségei:

1. **Projektvezető**
    - A projekt átfogó irányítása és felügyelete volt a fő feladata.
    - A csapatmunka összehangolásával biztosította a zökkenőmentes együttműködést, miközben a fejlesztési folyamatokat is ütemezte.
    - Felelős volt a hatékony kommunikációért a csapattagok és az érintettek között.
2. **Backend fejlesztő**
    - Kialakította és implementálta a rendszer háttérstruktúráját.
    - Kidolgozta a szerveroldali logikát és funkciókat, garantálva a gördülékeny működést.
    - Biztosította, hogy az alkalmazás technikai alapjai megbízhatóak és optimálisak legyenek.
3. **Unit tesztelő**
    - Ellenőrizte és tesztelte a háttérrendszer elemeit, ideértve a funkciókat és osztályokat.
    - Unit teszteket írt, futtatott, és garantálta a kód minőségét.

- Gondoskodott arról, hogy a backend megfeleljen a specifikációknak és szabványoknak.

Hegedűs Adrienn feladatai és felelősségei:

1. **Frontend fejlesztő**
    - Megtervezte és kivitelezte az alkalmazás felhasználói felületét.
    - Felelt az előtéri funkcionalitások megvalósításáért és azok hatékony működéséért.
2. **Manuális tesztelő**
    - Tesztelések révén visszajelzésekkel segítette a fejlesztőcsapatot.
    - Biztosította, hogy a kész alkalmazás megfeleljen az előírt minőségi követelményeknek.

Vajda Gergő feladatai és felelősségei:

1. **Adatbázis fejlesztő**
    - Az adatbázis tervezése és felépítése tartozott a feladatai közé.
    - Kialakította az adatbázis szerkezetét, valamint a rendszerhez szükséges kapcsolódási logikát.
    - Gondoskodott az adatok integritásáról és a rendszer hatékony működéséről.

A projekt során szerzett tapasztalatok alapján megállapítottuk, hogy a hatékony csapatmunka alapja a nyílt kommunikáció és a közös célok egyértelmű meghatározása. A csapat tagjai közötti együttműködés erősítette az innovatív megoldások kialakítását, és segítette a kihívások rugalmas kezelését. A közös munka során azonosítottunk olyan területeket, ahol a jövőben tovább kívánunk fejlődni, mint például a feladatok egyenletesebb elosztása és a felelősségek pontosabb meghatározása, hogy még hatékonyabban érjük el céljainkat.

Jövőbeli terveink

Jelszóváltoztatás:

A felhasználók a saját profiljukon belül biztonságosan megváltoztathatják a jelszavukat, ami növeli a fiókbiztonságot és a felhasználói élményt.

Statisztika:

Részletes statisztikák jelennek majd meg, például a felhasználók aktivitásáról, kedvelt tartalmakról vagy a csatornák növekedéséről, ami segít jobban átlátni a használati szokásokat.

Világos mód:

A sötét mód mellett világos felület is elérhető lesz, hogy a felhasználók a saját vizuális preferenciáiknak megfelelően tudják használni az oldalt.

Értesítések:

A rendszer értesítésekkel tájékoztatja majd a felhasználókat például új követőkről, hozzászólásokról vagy friss tartalmakról, így mindig naprakészen informáltak maradhatnak.

Feliratkozás csatornára:

A felhasználók különböző csatornákra tudnak majd feliratkozni, így könnyebben követhetik az őket érdeklő tartalomkészítőket, és azonnal értesülhetnek az új feltöltésekről.

Wishlist (kívánságlista):

A felhasználók megadhatnak legfeljebb öt kívánságot, hogy milyen típusú vagy témájú videókat szeretnének a jövőben látni, így a tartalomkészítők ezek alapján tudnak új videókat készíteni, személyre szabottabb élményt nyújtva.