Inkli

Projekt és fejlesztői dokumentáció

Tartalomjegyzék

A projekt ismertetése

A koncepció

A mai világban egyre inkább csak a digitális térben vagyunk jelen. Egyre rövidebb a fókuszálási képességünk, kevesebbet olvasunk - főleg könyvet. Az igény mégis meglenne a könnyed olvasásra, kikapcsolódásra. Az emberek vágynak a történetekre, arra, hogy elmerülhessenek mások gondolataiban, világában, de gyakran nincs idejük vagy energiájuk hosszú, több száz oldalas művekbe belekezdeni.

Az Inkli ezt oldaná meg. Az Inkli egy weboldal ahol rövid, vagy akár hosszabb történeteket oszthatnak meg a felhasználók egymással, legyen az valós vagy kitalált. A közösség tagjai szabadon böngészhetnek a történetek között, előre meghatározott kategóriák mentén – például romantikus, sci-fi, humor vagy bármi más. Emellett lehetőségük van más felhasználókat követni, így mindig naprakészek lehetnek kedvenc íróik legújabb történeteivel kapcsolatban. A legjobb történeteket a szerkesztők kiemeltté nyilváníthatják, hogy több emberhez elérjen.

Inspirációk

A projekt ötlete és vonása Redditből inspirálódott. Továbbá összehasonlítható egyes részeiben a WattPaddal. A design letisztultságra törekszik, ez is a Redditből és a mai nagyobb egységes designú oldalakból, pl.: Facebook.

Csoport tagjai

- Dienes Kristóf
- Dikácz Márton

Változások a fejlesztés alatt

Eredetileg hárman, Horváth Olivérrel, közösen dolgoztunk a projekten, viszont több hetes kórházi kezelését és az ebből eredő lemaradása miatt nélküle fejeztük be.

A weboldal funkciói

A weboldal koncepciója viszonylag egyszerűnek mondható, mégis sok és hasznos funkciót tettünk bele, hogy felhasználói szempontból kényelmes legyen a használata. A weboldal használatához szükség van regisztrált profilra, emellett meglévő profilba és természetesen be tudnak jelentkezni. A felhasználók a profiljaikat személyre szabhatják. Megváltoztathatják a nevüket és profilképüket. Amennyiben úgy kívánja az összes adatával együtt a profilját is törölheti.

Minden felhasználónak joga van az összes történetet megtekinteni és sajátot létrehozni. A történetek markdown formátumban írhatják. Megtekintés során pedig lehetőségük van olvasó módot bekapcsolni, így kényelmesebb az olvasás.

A történeteket, beleértve sajátjukat is likeolhatják vagy dislikeolhatják. Viszont minden történetre egyszer lehet szavazatot leadni, ezt később módosítani nem lehet. A likeokhoz hasonlóan a történeteket is csak egyszer lehet megírni, után módosítani nem lehet, se törölni.

A weboldalon lehetőség van keresésre. A keresés megkeresi az összes olyan felhasználót és történetet amiben a nevében szerepel a keresési szöveg. Ezen felül lehet szűrni a történeteket kategóriájuk alapján. A főoldalon továbbá lehet szortírozni abc sorrend, feltöltés dátuma és likeok száma alapján is.

Felhasználók követhetik egymást, illetve egyes kategóriákat így nem maradnak le kedvenc műveikről sem. A weboldal adminisztrátorai pedig kiemelhetnek történeteket amik kifejezetten jók lettek.

A felhasználók ahogy interaktálnak az oldallal érmeket gyűjthetnek, ezeket a profiljukon találják meg, megtekinthetik mások eredményeit is.

Az oldal adminisztrátorainak számára külön oldal van ahol tudják kezelni a felhasználókat, a kategóriákat illetve a történteket.

Asztali alkalmazás funkciói

Az asztali alkalmazás lehetővé teszi mindenki számára, hogy történeteket olvasson. Nincs szükség a felhasználónak bejelentkezésre. A történetet egy letisztult felületen olvashatja, közben lehetősége van beépített relaxáló hangok hallgatására is.

Technikai ismertető

Csoport Koordináció

A munka során szükség volt arra, hogy soron tudjuk követni a projekt haladását, illetve mindenki tudjon egyszerre dolgozni. Emellett fontos volt a kommunikáció, szükség esetén egymás segítése. Ezekre több programot és weboldalt is használtunk.

A feladatok kiosztására és azok haladásának nyomonkövetésére Trello-t használtunk. Ezen belül címkékkel láttuk el a feladatokat, így tudtuk, hogy milyen típusú a feladat (pl.: Backend, Frontend).

A felírt feladatokat, azok részleteit és egyéb dolgokat Discord csoportos beszélgetésben beszéltük meg. Amennyiben szükség volt, itt segítettünk egymásnak.

Az egységes kinézetet egy megosztott Figma fájlban készítettük el, így mindenki hozzáfért, készíthette el az alapján a weboldal egyes részeit.

A projekt forrásfájljait Git-el verzió kezeltük, ennek repositoryt hoztunk létre GitHubon (https://github.com/Gilbike/Inkli).

Az adatbázis

A projekt az adatok tárolására egy egyszerű fájlban tárolt SQLite adatbázist használ, ötvözve a könnyed kezelést és gyorsaságot.

Az adatbázis táblái, oszlopai

Az adatbázis a következő táblákból épül fel:

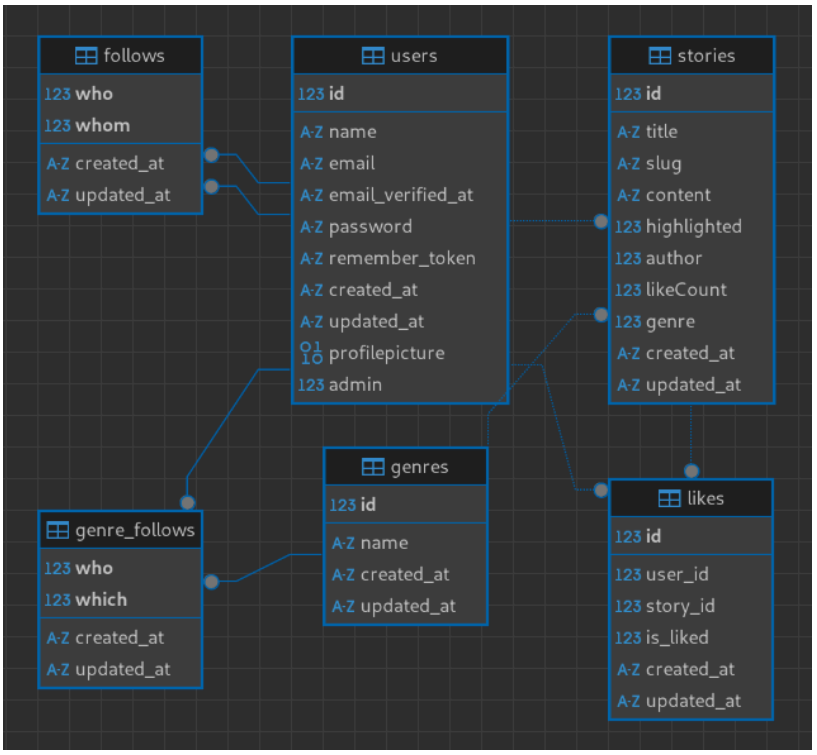
- **users** tábla: a regisztrált felhasználókat tárolja
  - *id*: Egyedi azonosító minden felhasználónak.
  - *name*: A felhasználó neve.
  - *email*: A felhasználó e-mail címe.
  - *email_verified_at*: Az időpont, amikor a felhasználó e-mail címe megerősítésre került.
  - *password*: A felhasználó jelszava
  - *remember_token*: Token az automatikus bejelentkezéshez (pl. "Emlékezz rám" funkció).
  - *created_at*: Létrehozás időpontja.
  - *updated_at*: Utolsó frissítés időpontja.
  - *profilepicture*: A felhasználó profilképe base64 formátumban
  - *admin*: Logikai érték, amely jelzi, hogy az adott felhasználó admin-e (1 = igen, 0 = nem).
- **follows** tábla: a felhasználók közötti követési kapcsolatot tárolja
  - *who*: A felhasználó, aki követ másokat (külső kulcs a users.id-re).
  - *whom*: A felhasználó, akit követnek (szintén a users.id-re mutat).
  - *created_at*: Követés létrehozásának időpontja.
  - *updated_at*: Követés utolsó frissítésének időpontja.
- **genre_follows** tábla: a felhasználók és követett kategóriákat tárolja
  - *who*: A felhasználó, aki követ kategóriákat (külső kulcs a users.id-re).
  - *whom*: A kategória, amit követnek (a genres.id-re mutat).
  - *created_at*: Követés létrehozásának időpontja.
  - *updated_at*: Követés utolsó frissítésének időpontja.
- **genres** tábla: a történetek műfajait tárolja
  - *id*: Egyedi azonosító minden műfajhoz.
  - *name*: A műfaj neve (pl. Sci-Fi, Romantikus, Thriller).
  - *created_at*: Létrehozás időpontja.
  - *updated_at*: Utolsó frissítés időpontja.
- **stories** tábla: a történeteket, azok tartalmát és adatait tárolja
  - *id*: Egyedi azonosító minden történethez.
  - *title*: A történet címe.
  - *slug*: URL-barát változata a címnek (pl. "az-en-tortenetem").
  - *content*: A történet tartalma (szöveg).
  - *highlighted*: Logikai mező, ami jelzi, hogy ki van-e emelve a történet.
  - *author*: A történet szerzője, külső kulcs a users.id-re.
  - *likeCount*: Hány kedvelést kapott a történet.
  - *genre*: A történet műfaja, külső kulcs a genres.id-re.
  - *created_at*: Létrehozás időpontja.
  - *updated_at*: Utolsó frissítés időpontja.

- **likes** tábla: a felhasználók kedveléseit tárolja el, melyeket a történetekre adtak
    - *id*: Egyedi azonosító minden kedvelés rekordhoz.
    - *user_id*: Annak a felhasználónak az azonosítója, aki a kedvelést tette (külső kulcs a users.id-re).
    - *story_id*: Annak a történetnek az azonosítója, amelyet a felhasználó kedvelt (külső kulcs a stories.id-re).
    - *is_liked*: Logikai mező, amely jelzi, hogy a kedvelés pozitív vagy negatív (like/dislike).
    - *created_at*: A kedvelés létrehozásának időpontja.
    - *updated_at*: A kedvelés utolsó frissítésének időpontja (pl. ha visszavonás vagy újrakedvelés történt).

Kapcsolat a táblák között

A users táblának van a legtöbb kapcsolata. Kapcsolódik a follows, genre_follows, likes táblához. Ezen felül megjelenik a stories táblában, ahol mint szerző van feltüntetve.

A stories tábla pedig kapcsolatban van a genres táblával. Innen tudja, hogy az adott történet milyen kategóriába lett besorolva.



Az adatbázis felépítése, kapcsolatai

A weboldal

A projekt legfontosabb eleme a weboldal, ez a fő elérési módja az Inklinek, a felhasználók csak ehhez férnek hozzá. A weboldal tökéletesen megjelenik számítógépes nézetben. Külön figyelmet fordítottunk, hogy mobilon is megfelelően használható legyen. Így aki az úton, vagy csak szimplán mobilon, esetleg tableten akarja megtekinteni az oldalt, van rá lehetősége.

Az oldal kinézete

A design letisztultságra törekszik, hogy mindenki számára könnyen kezelhető legyen. A legnagyobb inspirációk a Reddit és a Facebook voltak. Mindkét oldal modern, letisztult és átlátható, ezt szerettük volna mi is elérni.
A weboldalt Figmában terveztük meg. Itt létrehoztunk egy alap design sémát, meghatároztuk a fő színeket. Meghatároztuk a színeket, betűtípusokat. Utána egy két oldal megtervezése után kialakult az oldal egysége kinézete.
Az oldal fejlesztése közben törekedtünk arra, hogy minél több ember használhassa az oldalt. Így mobilon és tökéletesen működik az oldal. Továbbá a felhasználók válthatnak világos és sötét téma között is. Mindkét témának saját fő színe van, ezzel feldobva kicsit az oldal hangulatát.

Fejlesztéshez használt technológiák

A weboldal PHP-ban íródott, Laravel frameworkkel. Az adatbázist a Laravel kezeli belsőleg SQLite fájl formájában. A könnyebb fejlesztéshez és interaktívabb felülethez a Laravel által biztosított Inertia.js segítségével, Reactben készítettük el a weboldal frontendjét. A Reacten felül, a gyorsaság és egyszerűség érdekében pedig Tailwind CSS-t használtunk.

Asztali alkalmazás

Az asztali alkalmazás lehetővé teszi mindenki számára, hogy történeteket olvasson. Nincs szükség a felhasználónak bejelentkezésre. A történetet egy letisztult felületen olvashatja, közben lehetősége van beépített relaxáló hangok hallgatására is. A fejlesztéshez C#-ot és WPF-et (Windows Presentation Foundation-t) használtunk. Az adatokat a Laravel projektből egy api routeon keresztül kérjük le.

Fejlesztéshez használt eszközök, programok

A fejlesztés során nagy segítséget nyújtottak egyes programok és hozzájuk tartozó bővítmények. A webes oldal elkészítéséhez Visual Studio Code-ot használtunk. A könnyebb és hibamentesebb fejlesztéshez pedig használtuk a PHP extension pack Visual Studio Code-os bővítményt. Továbbá a Tailwind megkönnyítéséhez pedig a Tailwind CSS IntelliSense bővítményt használtuk.

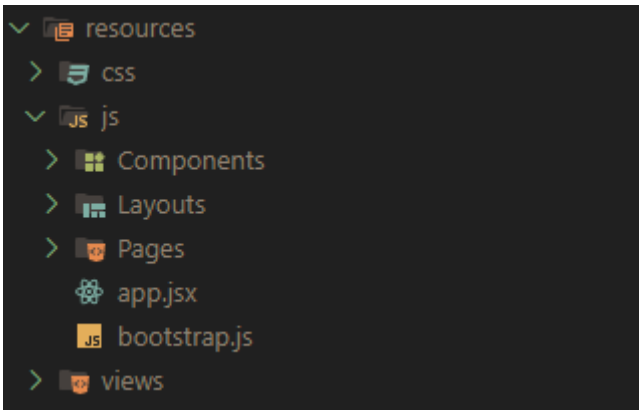
Fejlesztői dokumentáció

A frontend

A projekt frontendje Laravel, Inertia.js és React felhasználásával készült. A fejlesztés során törekedtünk a tiszta kód elveinek betartására és a jól érthető, karbantartható struktúra kialakítására. Ennek köszönhetően a frontend kódja könnyen áttekinthető és bővíthető.

A frontend struktúrája

A frontend struktúrája lehetővé teszi a komponensek egyszerű újrafelhasználását, a kód moduláris felépítését és a különböző oldalak közötti egységes megjelenést. A React használata pedig dinamikus, interaktív felhasználói felületet biztosít.



A frontend struktúrája

Mappaszerkezet ismertetése

- **js**: Ez a mappa tartalmazza a React alkalmazás forráskódját.
    - **components**: Ebben a mappában találhatók az önálló, újrahasznosítható React komponensek. Ezek a komponensek a felhasználói felület kisebb építőelemei (pl. gombok, űrlapok, listák).
    - **layouts**: Itt találhatók az alkalmazás alap elrendezését meghatározó komponensek. Ezek biztosítják az egységes megjelenést és struktúrát a különböző oldalak között (pl. fejléc, lábléc, oldalsáv).
    - **pages**: Ebben a mappában vannak az egyes oldalakhoz tartozó fő komponensek. Ezek a komponensek állítják össze az adott oldal tartalmát és funkcionalitását. A Laravel innen hívja meg az inertia komponenseket
- **css**: Ez a mappa tartalmazza a CSS fájlokat. Mivel a projekt Tailwind CSS-t használ így minimális módosítás lett itt alkalmazva
- **views**: Ez a mappa tartalmazza a Laravel view-kat. Az Inertia.js használata miatt ezek a view-k nagyon egyszerűek, és csak az alkalmazás kezdeti betöltéséért felelősek

Layoutok (elrendezések)

Az egész projekt egyetlen layouttal rendelkezik. Ez a komponens felel az oldal fejlécéért, láblécéért. Továbbá kezeli a sötét és világos mód közötti váltást is. Amennyiben meg van adva úgy beállítja a böngészőben megjelenő oldalcímet is.

Componensek

- **Button**
  Alapvető gomb komponens. Alkalmazkodik a megfelelő szín sémához és egységes megjelenést biztosít. Újrafelhasználó, könnyen személyre szabható. Opcionálisan biztosít kiemelt, nagyobb megjelenítési módot is.
- **Container**
  A tartalomnak meghatározott szélességű területet biztosít a különböző képernyőméreteken. Ez a komponens segít a weboldal tartalmának elrendezésében és igazításában, biztosítva, hogy az olvasható és esztétikus maradjon a különböző eszközökön.
- **DefaultCard**
  A weboldal különböző részeinek megjelenítésére lehet használni. Ez a komponens egy stílusozott keretet biztosít a benne lévő tartalom számára, opcionális címmel és egyedi háttérszínnel. A DefaultCard komponens segítségével a weboldalad egységes megjelenést kap, miközben a különböző tartalmak elkülönülnek és jól szervezettek maradnak.
- **Dropdown**
  Egy legördülő menüt hoz létre, amely helytakarékos és interaktív módon jeleníti meg a tartalmat vagy a menüpontokat. A komponens egy gombból és egy lenyíló listából áll. A gombra kattintva a lista megjelenik, és a felhasználó kiválaszthat egy elemet, vagy megtekintheti a további tartalmat. A Dropdown komponens testreszabható méretekkel és stílusokkal rendelkezik.
- **FaqBox**
  Gyakran ismételt kérdések (GYIK) megjelenítésére szolgáló elem. A komponens egy kérdésből és a hozzá tartozó válaszból áll. Alapértelmezés szerint a válasz el van rejtve, és a felhasználó a kérdésre kattintva tudja megjeleníteni vagy elrejteni azt.
- **FilterRow**
  A komponens két fő részből áll: egy "Műfaj szerinti szűrés" gombból és egy "Rendezés" gombból. Mindkét gomb lenyíló menüket nyit meg, amelyek lehetővé teszik a felhasználók számára, hogy a tartalmat a kívánt szempontok szerint szűrjék vagy rendezzék. A főoldalon jelenik meg, ahol a történetek felett kapott helyett.
- **Footer**
  A weboldal láblécét tartalmazza.
- **FormInput**
  Egy űrlapmező létrehozására szolgál, amely egy címkéből, egy beviteli mezőből és egy hibaüzenetből áll. A komponens egyszerűsíti az űrlapok készítését azáltal, hogy egységes megjelenést és működést biztosít az űrlapmezők számára.
- **Header**
  A weboldal fejlécének megjelenítésére szolgál. Tartalmazza a logót, a bejelentkezett felhasználók számára a keresési lehetőséget, az új tartalom létrehozásának gombját és a felhasználói menüt (profil linkkel, sötét mód kapcsolóval és kijelentkezéssel). A be nem jelentkezett felhasználók számára egy linket biztosít a bejelentkezési oldalra.

- **Input**
  Beviteli mezőt hoz létre, amely egy ikont és egy stílusozott input mezőt tartalmaz, amelyet űrlapokon lehet használni.
- **Like**
  Megjeleníti a likeok számát, lehetővé teszi a felhasználó számára, hogy likeoljon vagy dislikeoljon egy tartalmat, és kezeli ezeknek a műveleteknek a küldését a szerver felé.
- **MarkdownEditor**
  Felhasználóbarát felületet biztosít történetek írásához és közzétételéhez. Lehetővé teszi a felhasználó számára, hogy szöveget írjon és formázzon Markdown nyelven, valós idejű előnézettel, és tartalmaz egy űrlapot a cím és a tartalom beviteléhez. A komponens tartalmaz továbbá egy legördülő menüt a műfaj kiválasztásához, űrlapellenőrzést és hibaüzeneteket, valamint visszajelzést a sikeres közzététel esetén.
- **ProfileCard**
  Megmutatja a felhasználó nevét, profilképét, az általa írt történetek számát, és jelzi, ha a felhasználó adminisztrátor. Ha a megtekintett profil a sajátunk, akkor egy "Edit" gombot jelenít meg, egyébként pedig egy "Follow" vagy "Unfollow" gombot, attól függően, hogy követjük-e már az adott felhasználót.
- **Sidebar**
  Oldalsáv, amely navigációs linkeket tartalmaz. Ezek a linkek elvezetik a felhasználót a főbb oldalakhoz, mint például a "Stories", "Highlights" és "My Stories". Ezenkívül egy legördülő menü is található, amely a követett műfajokat és felhasználókat listázza
- **Story**
  Egy történet összefoglalóját jeleníti meg, beleértve a címet, a rövid leírást és a műfajt. Interaktív funkciókat is kínál. A felhasználók a címre kattintva megtekinthetik a teljes történetet, és vannak "Like" és "Share" gombok is. A "Share" gomb a történet linkjét a felhasználó vágólapjára másolja, és egy rövid üzenet jelzi, hogy a link másolása megtörtént.

Oldalak (pages mappa)

Ebben a mappában található React komponenseket jeleníti meg a Laravel az Inertia JS segítségével. Ezek a komponensek kapják meg az adatokat, majd jelenítik meg azokat. A layout és a components mappában lévő komponenseket felhasználva állnak össze az oldalak.

- **Auth**
  - **Login**: Bejelentkezési felület oldala *[/login]*
  - **Register**: Regisztrációs felület *[/register]*
- **Genres**
  - **Show**: Megjeleníti egy adott kategória összes történetét *[/genres/{kategória azonosítója}]*
- **Profile**
  - **Edit**: A bejelentkezett felhasználó szerkesztésének oldala *[/profile/edit]*
- **Stories**
  - **Create**: Űrlap új történet létrehozásához *[/stories/create]*
  - **Highlighted**: Kiemelt történetek listája *[/highlights]*
  - **Index**: Az összes történetet megjelenítő oldal, egyben főoldal a bejelentkezett felhasználóknak *[/stories]*
  - **MyStories**: A bejelentkezett felhasználó történeteit listázza ki *[/my-stories]*
  - **Show**: Egy adott történet oldala *[/stories/{történet azonosítója}]*

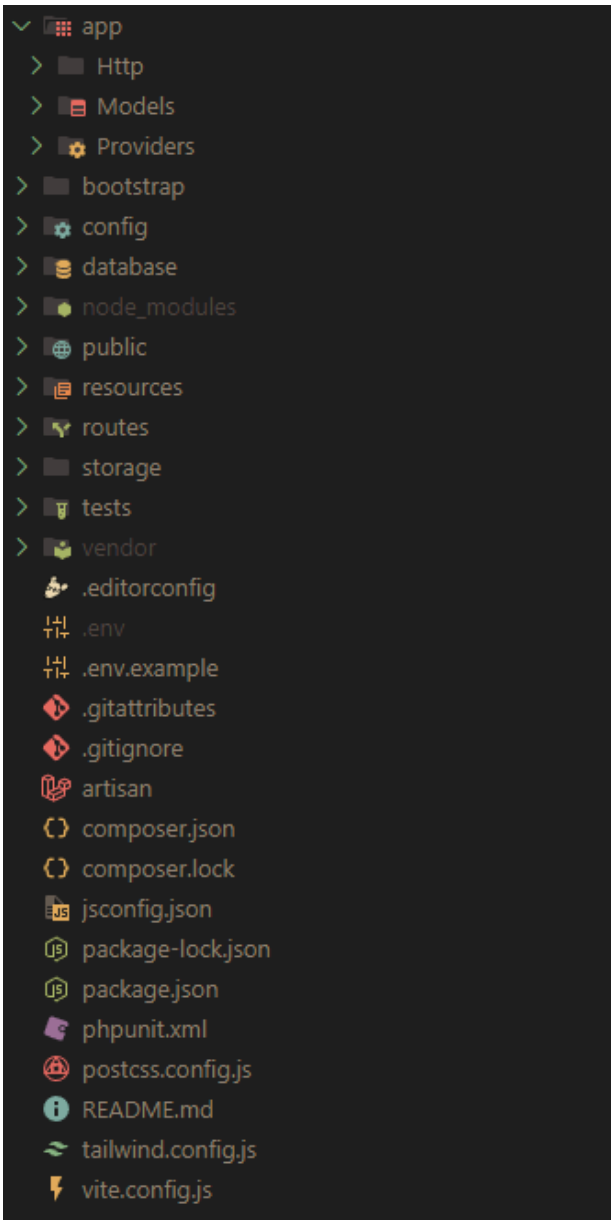
- ➔ **Aboutus**: A weboldal rövid ismertetője *[/aboutus]*
- ➔ **AdminPanel**: Adminisztrátori felület *[/adminpanel]*
- ➔ **Landing**: Az oldal főoldala *[/]*
- ➔ **Markdown**: Rövid útmutató a markdown nyelvhez *[/markdown]*
- ➔ **Policy**: Adatvédelmi tájékoztató oldala *[/privacy-policy]*
- ➔ **Profile**: Egy felhasználó oldalát jeleníti meg, lehet ez a saját profil is *[/profile][/user/{felhasználó azonosítója}]*
- ➔ **Search**: A keresés eredményeit listázó oldal *[/search]*

A backend

A projekt backendje PHP-ben, Laravel frameworkkel készült el. A fejlesztés során próbáltuk minél inkább a Laravel előre kialakított útmutatásait és struktúráját megtartani. Tovább próbáltuk a Clean kód elveket betartva és minél több kommentet használva könnyen olvasható és érthető kódot csinálni. Ennek köszönhetően a kód a gyorsan megérthető.

A backend struktúrája

A projekt teljesen a Laravel előre meghatározott mappa szerkezetén alapul, azt használja. A következőkben ezt fogjuk részletezni, illetve a saját erre létrehozott rendszerünket bemutatni.



Laravel Projekt mappái képen

Mappaszerkezet ismertetése

A Laravel sok alapértelmezetten létrehozott mappát tartalmaz, de most csak azokat soroljuk fel, amik a projekt szempontjából fontosak vagy mi módosítottunk rajta.

- **app** mappa: a Laravel itt tárolja el a weboldal controllereit, illetve az adatmodelleket.
    - Http mappa: ezen belül találhatóak a middlewerek, a controllerek, illetve az létrehozásért és frissítésért felelős request fájlok.
    - Models mappa: itt vannak az adatbázisban tárolt adatokat reprezentáló, objektum orientált modellek.
- **config** mappa: a konfigurációs fájlok helye.
- **database** mappa: az adatbázishoz tartozó összes fájl helye.
    - factories mappa: itt vannak a modellek "gyárai". Ezek segítségével gyorsan, generált adatokkal, kamu rekordokat gyárthatunk, ezekkel később tesztelhetjük a weboldalt.
    - migrations mappa: ebben a mappában kaptak helyet az adatbázis felépítésének módosítását leíró fájlok. Minden fájl egy-egy tábla létrehozását vagy módosítását írja le.
    - seeders mappa: az itt található fájlok lefutásakor a factoryk által generált kamu adatokat eltárolja az adatbázisban. Ebben a fájlban lehet megmondani például, hogy hány kamu adat legyen generálva vagy milyen kapcsolatban álljanak egymással.
- **public** mappa: a felhasználók számára is elérhető források helye. Ilyenek például a képek.
    - **img** mappa: nem alapértelmezetten létrehozott mappa. Ebben a mappában tároljuk el a weboldalon megjelenő képeket.
- **resources** mappa: a frontendet tartalmazza. Az Inertia JS segítségével Reactben íródott. A frontendről részletesen más bekezdésben írunk.
    - **css** mappa: a weboldalon megjelenő css fájlokat tároljuk
    - **js** mappa: a frontend legfontosabb mappája, itt található a React forráskódja
    - **views** mappa: a Laravel viewkat tárolja, mivel Reactben írtuk a projektet, így nem módosítottunk sokat rajta
- **routes** mappa: ezen belül található a web.php fájl, ez határozza meg, hogy milyen útvonalak vannak az oldalon.

Middlewarek

A weboldalon megtalálhatóak adminisztrátorok. A profilukon ez jelezve is van. A weboldal a users tábla egyik oszlopából tudja, hogy ki tölt be ilyen szerepet. Vannak olyan elérési utak az oldalon, amit csak nekik szabad elérni. Ehhez saját middlewaret készítettünk, így csak ők látogathatják meg azokat az oldalakat ami nekik van szánva. Amennyiben más próbálja meg, ők vissza lesznek irányítva a főoldalra.

Ezen felül a mappa a Reacthez szükséges Inertia JS middlewaret tartalmazza.

Requestek

Middlewarekhez hasonlóan saját request fájlt nem hoztunk létre. Minden validációt és authorizációt a controlleken belül oldottunk meg. Egyedül a bejelentkezéshez szükséges LoginRequest fájl található meg a requestek mappájában amit a Laravel Breeze starter kit hozott létre a beépített bejelentkezés funkció működéséhez.

Viewok

A Laravel MVC modellt alkalmaz a webfejlesztéshez. A modell szabja meg milyen módon tárolódik az adat. A view ezt megjeleníti, a kettőt a controller köti össze.

A Laravel a viewokat a saját php alapú megjelenítéséhez használja, így mivel mi Reactben fejlesztük le a frontendet, saját viewt nem hoztunk létre.

A modellek

A Laravel MVC modellt alkalmaz a webfejlesztéshez. A modell szabja meg milyen módon tárolódik az adat. A view ezt megjeleníti, a kettőt a controller köti össze.

A projektünkben majdnem minden adatbázis táblához kapcsoltunk modellt.

User modell

Alapból a Laravel által létrehozva. Ennek segítségével oldja meg a framework a felhasználó kezelést. A mi esetünkben kiegészítésre került több metódussal, hogy elérhessük a felhasználóhoz kötött egyéb adatokat.

- **stories** metódus
    - visszaadja a felhasználóhoz kötött, általa írt, összes történetet amit az adatbázisban tárolunk
- **likes** metódus
    - visszaadja a felhasználó által leadott összes szavazatot. Ez együtt tartalmazza a kedvelések és nem tetszéseket (dislike).
- **following** metódus
    - visszaadja az összes olyan felhasználót és azok adatait akit a felhasználó bekövetett.

Story modell

A stories táblában tárolt adatokat kezelő modell, ezek a történetek. Metódusokkal könnyű hozzáférést biztosít a stories tábla kapcsolataihoz. Továbbá a történeteket egyes adataihoz amelyek más táblában vannak tárolva.

- **author** metódus
    - visszaadja az írót/szerzőt (User-t) az adott történethez.
- **likes** metódus
    - visszaadja a történetre leadott szavazatokat (like/dislike). Ez az összes kedvelést visszaadja, függetlenül attól, hogy pozitív vagy negatív.
- **genre** metódus
    - visszaadja a történet kategóriáját.

Genre modell

A genres táblában tárolt adatokat kezelő modell. Ez a modell kezeli a kategóriákat amikben a felhasználók történeteket hozhatnak létre.

- **stories** metódus
    - visszaadja az összes olyan történetet amit az adott kategóriában hoztak létre

Like modell

A likes táblában elmentett adatokért felelős modell. Egyszerre kezeli a likeokat és dislikeokat. Az is_liked oszlop felelős az ezek közti megkülönböztetésért.

- **story** metódus
    - visszaadja a történetet amelyre az adott értékelést leadták.
- **user** metódus
    - visszaadja a felhasználót aki az értékelést leadta

A controllerek

A Laravel MVC modellt alkalmaz a webfejlesztéshez. A modell szabja meg milyen módon tárolódik az adat. A view ezt megjeleníti, a kettőt a controller köti össze.

Ügyeltünk rá, hogy minél érthetőbb, átláthatóbb rendszert alakítsunk ki a controllerekben. A Laravel nagy hangsúlyt fektet a resource controllerek használatára. Ezekben megadott nevű metódusok megadott művelethez vannak kötve (például: store - tárolás, destroy - törlés). Ezeket mi is használtuk. Ezen felül egyes controllerek plusz metódusokat kaptak plusz funkciók implementálásához.

RegisteredUserController

A Laravel Breeze által létrehozott controller, mi ehhez nem nyúltunk. A felhasználók létrehozásáért felelős.

- **create** metódus megjeleníti a regisztrációs felületet
- **store** metódus pedig eltárolja a megadott adatokkal a felhasználót

AuthenticatedSessionController

A Laravel Breeze által létrehozott controller, mi ehhez nem nyúltunk. A felhasználók be és kiléptetéséért felelős.

- **create** metódus megjeleníti a bejelentkezési felületet
- **store** metódus végrehajta a beléptetést
- **destroy** metódus pedig kilépteti a felhasználót meghívás esetén

AdminController

Egyetlen feladata az adminisztrációs oldal megjelenítése és az ahhoz kellő adatok összegyűjtése. Ezekért a *show* metódus felel.

ProfileController

A ProfileController a felhasználói profil kezelését végző controller. Alapvetően egy resource controller, tehát rendelkezik a Laravel sémája szerinti metódusokkal. Ezenfelül viszont pár extra funkciót is kezel, mely a profilok kezeléséhez, profilok közötti interakcióhoz tartozik.

- **show** metódus

    A bejelentkezett felhasználó profil oldalát jeleníti meg. Lekéri az összes adatot ami a

felhasználóhoz tartozik. Beletartozik ebbe az összes általa írt történet, illetve az összes leadott értékelés is. Itt kerül az is eldöntésre, hogy mely jelvényeket jelenítse meg a weboldal az oldalon. Ezeknek a számolása itt meg végbe.

- **edit** metódus

  Mindenfajta egyéb számolás nélkül megjeleníti a profil szerkesztése oldalt. Ez az oldal a jelenleg bejelentkezett felhasználót tudja szerkeszteni.

- **update** metódus

  Frissíti a bejelentkezett felhasználó adatait az adatbázisban. Validáció elvégzése után. A feltöltött profilképet base64 formátumba konvertálja, hogy így tárolható legyen az adatbázisban. Sikeres frissítés esetén az előző oldalra dob vissza (általában a szerkesztés oldal).

- **destroy** metódus

  Törli a jelenleg bejelentkezett felhasználó profilját és adatait. Törlés után kilépteti a felhasználót esetleges hibák elkerülése céljából. A adatbázis szerkezetében meg van határozva, hogy a felhasználó törlése esetén a hozzá kapcsolódó egyéb adatokat is törölje más táblákból. A szerkesztés oldalon található egy gomb amivel a profil törölhető.

- **deleteByAdmin** metódus

  Törli egy adott felhasználó profilját. Ezt csakis egy adminisztrátor kezdeményezheti az admin oldalról.

- **stories** metódus

  Egy oldalt jelenít meg ahol a jelenlegi felhasználó összes történetét listázza ki. Azokat a történeteket amiket a felhasználó maga hozott létre.

- **other** metódus

  A weboldalon lehetőség van más felhasználók megtekintésére. Ez a metódus megjeleníti ugyanazt az oldalt amit a felhasználó saját profilja megtekintése esetén is lát, viszont a *user* változóban tárolt másik felhasználó adatait jeleníti meg. A saját profil esetén is kiszámolt jelvényeket a weboldal feldolgozza, elküldi a frontendnek. Ezen felül a metódus megállapítja, hogy a jelenlegi felhasználó követi-e a megtekinteni kívánt felhasználót. Ez szükséges a megfelelő követés gomb megjelenítéséhez.

- **follow** metódus

  A metódus új rekordot rögzít a follows táblában. Ezzel a felhasználó aki éppen aktív beköveti a *user* változóban tárolt másik felhasználót.

- **unfollow** metódus

  A metódus rekordot töröl a follows táblából. Ezzel a felhasználó aki éppen aktív kiköveti a *user* változóban tárolt másik felhasználót.

- **grantAdmin** metódus

  Egy adott felhasználónak adminisztrátori jogosultságokat ad, majd menti ezt az adatbázisban.

- **removeAdmin** metódus

  Egy adott felhasználót megfogszt az adminiszitrátori jogosultságoktól, ezt menti az adatbázisban is.

StoryController

A StoryController felelős az összes, történettel kapcsolatos funkciók kezeléséért. A ProfileControllerhez hasonlóan resource controller, kiegészítve egyedi funkciók miatt létrehozott egyéb metódusokkal.

- **index** metódus

  Adatok összegyűjtése után megjeleníti az összes létrehozott történetet. Az url paraméterek alapján eldönti a rendszerezést. Ez lehet abc sorrend, like szám csökkenő sorrendben vagy létrehozás dátuma legújabbtól kezdve. Ezt követően kiszűri a történeteket amennyiben van meghatarázott kategória filter. Ezeket utána a frontendnek továbbítja.
- **create** metódus

  Megjeleníti az oldalt ahol a történeteket létre lehet hozni.
- **getLike** metódus

  Megkeresi és visszaadja json formátumba az adott történetre a jelenlegi felhasználó által leadott likeot vagy dislikeot. Amennyiben nincs like akkor is tér vissza értékkel.
- **like** metódus

  A jelenlegi felhasználóként pozitív értékelést ad le a történetre. A történet és felhasználó keresése után felveszi az adatbázisba a változást.
- **dislike** metódus

  A jelenlegi felhasználóként negatív értékelést ad le a történetre. A történet és felhasználó keresése után felveszi az adatbázisba a változást.
- **store** metódus

  Létrehoz egy történetet és eltárolja azt az adatbázisban. Először validálja az adatokat, majd slugot hoz létre és hozzáköti felhasználóhoz. Sikeresség esetén pedig vissza irányítja a felhasználót a létrehozás oldalra, egy sikeresség üzenettel.
- **show** metódus

  Egy kiválasztott történetet megjelenít. Lekéri a létrehozó felhasználó adatait, a kedvelések és negatív értékelések számát. Ezeket továbbítja az oldal felé. Az oldalon megtalálható zen mód kapcsolását is itt dolgozza fel.
- **edit, update** metódus

  Az oldalon nincs lehetőség a történetek szerkesztésére és törlésére. Ez szándékos tervezési döntés volt, ahogy az egyszeri értékelés leadás is. Ezzel szeretnénk hangsúlyozni, hogy fontos a megfelelő fogalmazás, tartalom és értékelés.

- **destroy** metódus

  Törli a történetet az adatbázisból. Moderáció miatt az adminisztrátoroknak joga van töröli a történeteket.
- **showHighlighted** metódus

  Megkeresi az összes olyan történetet ami a szerkesztők/adminisztrátorok által kiemelve lettek. Ezeket összegyűjtve megjeleníti a felhasználó számára.
- **toggleHighlight** metódus

  Abban az esetben ha egy adminisztrátor meg akarja változtatni egy adott történet kiemelését, akkor ez a metódus fog cselekedni. Amennyiben már ki van emelve akkor elveszi a kiemelést, ellenkező esetben pedig kiemelést ad neki. Ezeket a változtatásokat pedig menti az adatbázisban is.

SearchController

A SearchController mindösszesen egy feladatért felelős. Egyetlen metódusa kikeresi a keresési feltételeknek megfelelően a találatokat és átadja azt a frontendnek. Nem resource controller.

- **search** metódus

  A *q* url paraméter alapján kikeresi az összes olyan felhasználót és történetet aminek a nevében szerepel a keresett szöveg. Ezeket továbbítja a keresés oldalnak ahol megjelennek ezek a találatok

GenreController

A kategóriákat kezeli. Az adatbázisban ez a *genres* tábla. Nem resource controller, de annak egy két metódusát megvalósítja.

- **show** metódus

  Megjelenít egy oldalt ahol az összes történetet megjeleníti ami az adott kategóriában van
- **store** metódus

  Validálást követően az adatbázisban eltárol egy új kategóriát. Csakis adminisztrátor hozhat létre.
- **destroy** metódus

  Töröl egy adott kategóriát. Csakis adminisztrátor törölhet kategóriát.
- **follow** metódus

  A metódus új rekordot rögzít a genre_follows táblában. Ezzel a felhasználó aki éppen aktív beköveti a *genre* változóban tárolt kategóriát.
- **unfollow** metódus

  A metódus rekordot töröl a genre_follows táblából. Ezzel a felhasználó aki éppen aktív kiköveti a *genre* változóban tárolt kategóriát.

A weboldal útvonalai

Az elérhető oldalakat a Laravelben a routes/web.php fájl határozza meg. Ebbe lehet megírni, mely utakon mely controller metódusa cselekedjen.

- ➔ / (főoldal, alapértelmezett amikor valaki felkeresi a webhelyet)
  - ◆ Leírás: Az oldal főoldala, megtalálható itt egy rövid gyakorikérdések rész, illetve gombok a regisztrálni kívánó felhasználóknak.
  - ◆ Korlátozások: nincs
  - ◆ Felelős: a web.php-n belül definiált névtelen függvény
  - ◆ Válasz típusa: weblap (Landing frontend komponens)
- ➔ /aboutus
  - ◆ Leírás: Egy rövid oldal, ahol leírást kaphatnak a látogatók az oldalról, illetve a készítőkről.
  - ◆ Korlátozások: nincs
  - ◆ Felelős: a web.php-n belül definiált Inertia útvonal
  - ◆ Válasz típusa: weblap (Aboutus frontend komponens)
- ➔ /privacy-policy
  - ◆ Leírás: Oldal, ahol a látogató megtudhatja, milyen típusú adatokat tárolunk róla és hogy miként rendelkezhet felette.
  - ◆ Korlátozások: nincs
  - ◆ Felelős: a web.php-n belül definiált Inertia útvonal
  - ◆ Válasz típusa: weblap (Policy frontend komponens)
- ➔ /markdown
  - ◆ Leírás: Segítség a történetek megírásához. Rövid útmutató a markdown működéséről. Bemutatja az összes, oldalon használható markdown funkciót és szintaxist.
  - ◆ Korlátozások: Nincs
  - ◆ Felelős: a web.php-n belül definiált Inertia útvonal
  - ◆ Válasz típusa: weblap (Markdown frontend komponens)
- ➔ /stories
  - ◆ Leírás: Az összes történet listájának megjelenítése.
  - ◆ Korlátozások: Bejelentkezett felhasználó szükséges
  - ◆ Felelős: *StoryController* osztály *index* metódusa
  - ◆ Válasz típusa: weblap
- ➔ /stories/create
  - ◆ Leírás: Űrlap megjelenítése új történet létrehozásához.
  - ◆ Korlátozások: Bejelentkezett felhasználó szükséges
  - ◆ Felelős: *StoryController* osztály *create* metódusa
  - ◆ Válasz típusa: weblap
- ➔ /stories (POST)
  - ◆ Leírás: Új történet létrehozása az űrlapon megadott adatokkal.
  - ◆ Korlátozások: Bejelentkezett felhasználó szükséges
  - ◆ Felelős: *StoryController* osztály *store* metódusa
  - ◆ Válasz típusa: átirányítás
- ➔ /stories/{történet azonosítója}
  - ◆ Leírás: Egy adott történet részleteinek megtekintése.
  - ◆ Korlátozások: Bejelentkezett felhasználó szükséges
  - ◆ Felelős: *StoryController* osztály *show* metódusa

- Válasz típusa: weblap

➔ /my-stories
- Leírás: Saját történetek megtekintése. Az aktuális felhasználó által létrehozott történetek listájának megjelenítése.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *ProfileController* osztály *stories* metódusa
- Válasz típusa: weblap

➔ /highlights
- Leírás: Kiemelt történetek megtekintése. Az adminisztrátor által kiemelt történetek listájának megjelenítése.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *StoryController* osztály *showHighlighted* metódusa
- Válasz típusa: weblap

➔ /stories/{történet azonosítója}/like
- Leírás: Történet kedvelésének lekérdezése. Megadja, hogy az aktuális felhasználó kedvelte-e a megadott történetet.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *StoryController* osztály *getLike* metódusa
- Válasz típusa: json (a like objektuma vagy null)

➔ /stories/{történet azonosítója}/like (POST)
- Leírás: Történet kedvelése.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *StoryController* osztály *like* metódusa
- Válasz típusa: json

➔ /stories/{történet azonosítója}/dislike
- Leírás: Történet nem kedvelése.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *StoryController* osztály *dislike* metódusa
- Válasz típusa: json

➔ /genres/{műfaj azonosítója}
- Leírás: Műfaj megtekintése. Az adott műfajhoz tartozó történetek listájának megjelenítése.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *GenreController* osztály *show* metódusa
- Válasz típusa: weblap

➔ /genres/{műfaj azonosítója}/follow
- Leírás: Műfaj követése. Lehetővé teszi az aktuális felhasználó számára, hogy kövesse a megadott műfajt.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *GenreController* osztály *follow* metódusa
- Válasz típusa: átirányítás

➔ /genres/{műfaj azonosítója}/unfollow
- Leírás: Műfaj követésének megszüntetése. Lehetővé teszi az aktuális felhasználó számára, hogy megszüntesse a megadott műfaj követését.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *GenreController* osztály *unfollow* metódusa
- Válasz típusa: átirányítás

➔ /profile

- Leírás: Profil megtekintése. Az aktuális felhasználó profiljának adatainak megjelenítése.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *ProfileController* osztály *show* metódusa
- Válasz típusa: weblap

➔ /profile/edit
- Leírás: Profil szerkesztése. Lehetővé teszi az aktuális felhasználó számára a profiljának adatainak szerkesztését.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *ProfileController* osztály *edit* metódusa
- Válasz típusa: weblap

➔ /profile/edit (POST)
- Leírás: Profil frissítése. Feldolgozza az aktuális felhasználó által megadott új profiladatokat és frissíti azokat.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *ProfileController* osztály *update* metódusa
- Válasz típusa: átirányítás

➔ /profile/delete
- Leírás: Profil törlése. Lehetővé teszi az aktuális felhasználó számára a profiljának törlését.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *ProfileController* osztály *destroy* metódusa
- Válasz típusa: átirányítás

➔ /search
- Leírás: Keresés. Lehetővé teszi a felhasználók számára, hogy történetek és felhasználók között keressenek kulcsszavak alapján.
- Korlátozások: Nincs
- Felelős: *SearchController* osztály *search* metódusa
- Válasz típusa: weblap

➔ /user/{felhasználó azonosítója}
- Leírás: Felhasználó profiljának megtekintése. Egy adott felhasználó nyilvános profiljának adatainak megjelenítése.
- Korlátozások: Nincs
- Felelős: *ProfileController* osztály *other* metódusa
- Válasz típusa: weblap

➔ /user/{felhasználó azonosítója}/follow
- Leírás: Felhasználó követése. Lehetővé teszi az aktuális felhasználó számára, hogy kövesse a megadott felhasználót.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *ProfileController* osztály *follow* metódusa
- Válasz típusa: átirányítás

➔ /user/{felhasználó azonosítója}/unfollow
- Leírás: Felhasználó követésének megszüntetése. Lehetővé teszi az aktuális felhasználó számára, hogy megszüntesse a megadott felhasználó követését.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *ProfileController* osztály *unfollow* metódusa
- Válasz típusa: átirányítás

➔ /adminpanel

- Leírás: Adminisztrációs felület megjelenítése.
- Korlátozások: Adminisztrátori jogosultság szükséges
- Felelős: *AdminController* osztály *show* metódusa
- Válasz típusa: weblap

➔ /user/{user}/ban
- Leírás: Felhasználó kitiltása. Lehetővé teszi az adminisztrátor számára, hogy kitiltson egy felhasználót. Ez a metódus törli a felhasználó profilját
- Korlátozások: Adminisztrátori jogosultság szükséges
- Felelős: *ProfileController* osztály *deleteByAdmin* metódusa
- Válasz típusa: átirányítás

➔ /user/{user}/grant
- Leírás: Adminisztrátori jog megadása. Lehetővé teszi az adminisztrátor számára, hogy adminisztrátori jogot adjon egy felhasználónak.
- Korlátozások: Adminisztrátori jogosultság szükséges
- Felelős: *ProfileController* osztály *grantAdmin* metódusa
- Válasz típusa: átirányítás

➔ /user/{user}/remove
- Leírás: Adminisztrátori jog visszavonása. Lehetővé teszi az adminisztrátor számára, hogy visszavonja egy felhasználó adminisztrátori jogát.
- Korlátozások: Adminisztrátori jogosultság szükséges
- Felelős: *ProfileController* osztály *removeAdmin* metódusa
- Válasz típusa: átirányítás

➔ /genres/create
- Leírás: Új műfaj létrehozása.
- Korlátozások: Adminisztrátori jogosultság szükséges
- Felelős: *GenreController* osztály *store* metódusa
- Válasz típusa: átirányítás

➔ /genres/{genre}/delete
- Leírás: Műfaj törlése.
- Korlátozások: Adminisztrátori jogosultság szükséges
- Felelős: *GenreController* osztály *destroy* metódusa
- Válasz típusa: átirányítás

➔ /stories/{story}/highlight-toggle
- Leírás: Történet kiemelésének/eltávolításának váltása. Lehetővé teszi az adminisztrátor számára, hogy egy történetet kiemeljen vagy eltávolítsa a kiemeltek közül.
- Korlátozások: Adminisztrátori jogosultság szükséges
- Felelős: *StoryController* osztály *toggleHighlight* metódusa
- Válasz típusa: átirányítás

➔ /stories/{story}/remove
- Leírás: Történet eltávolítása.
- Korlátozások: Adminisztrátori jogosultság szükséges
- Felelős: *StoryController* osztály *destroy* metódusa
- Válasz típusa: Átirányítás

➔ /register
- Leírás: Regisztrációs oldal megjelenítése.
- Korlátozások: Csak nem bejelentkezett felhasználók számára elérhető
- Felelős: *RegisteredUserController* osztály *create* metódusa

- Válasz típusa: weblap

➔ /register (POST)
- Leírás: Új felhasználó regisztrálása.
- Korlátozások: Csak nem bejelentkezett felhasználók számára elérhető
- Felelős: *RegisteredUserController* osztály *store* metódusa
- Válasz típusa: átirányítás

➔ /login
- Leírás: Bejelentkezési oldal megjelenítése.
- Korlátozások: Csak nem bejelentkezett felhasználók számára elérhető
- Felelős: *AuthenticatedSessionController* osztály *create* metódusa
- Válasz típusa: weblap

➔ /login (POST)
- Leírás: Felhasználó bejelentkezése.
- Korlátozások: Csak nem bejelentkezett felhasználók számára elérhető
- Felelős: *AuthenticatedSessionController* osztály *store* metódusa
- Válasz típusa: átirányítás

➔ /logout
- Leírás: Felhasználó kijelentkezése.
- Korlátozások: Bejelentkezett felhasználó szükséges
- Felelős: *AuthenticatedSessionController* osztály *destroy* metódusa
- Válasz típusa: átirányítás

Asztali alkalmazás

Könyvtárstruktúra

- **Assets:** A projekt által használt statikus erőforrást tartalmazza. A relaxáló hangot.
- **models:** A projekt adatmodelljeit tartalmazó C# osztályfájlokat foglalja magában.
- **models/Author.cs:** Az API-ból lekért történethez kapcsolódó szerző tároló osztály fájlja.
- **models/Genre.cs:** Az API-ból lekért történethez kapcsolódó kategória tároló osztály fájlja.
- **models/Story.cs:** Az API-ból lekért történetet tároló osztály fájlja.
- **App.xaml:** Ez egy XAML (Extensible Application Markup Language) fájl, amely az alkalmazás felhasználói felületének deklaratív leírására szolgál WPF (Windows Presentation Foundation) projektben
- **MainWindow.xaml:**Alkalmazás főablakának felhasználói felületét definiálja.
- **StoryReader.xaml:** Felhasználói felületet definiál, amely történetek olvasására szolgál.

Code-behind

MainWindow.xaml.cs

A fő célja, hogy történeteket jelenítsen meg a külső API-ból. Betölti és megjeleníti a történeteket az API-ból egy listanézetben. Kezeli az API-val való kommunikáció során fellépő esetleges hibákat.

- **LoadInkliStories** metódus: Ez egy async void metódus, ami azt jelenti, hogy aszinkron módon fut, de nem ad vissza értéket. Meghívja a ReadInkliAPI() metódust a történetek lekéréséhez az API-ból. Ha az API válasza null, megjeleníti a hibaüzenet-konténert. Deserializálja az API választ egy List<Story> objektumba a JsonConvert.DeserializeObject<List<Story>>(apiResponse) segítségével. Hozzáadja a történeteket a Stories ObservableCollection-höz, ami frissíti a felhasználói felületet.
- **ReadInkliAPI** metódus: Ez egy async Task<string> metódus, ami azt jelenti, hogy aszinkron módon fut, és egy stringet ad vissza, ami az API válasza. Létrehoz egy HttpClient objektumot a HTTP kérések küldéséhez. Megpróbálja lekérni a történeteket az API-tól a client.GetAsync(API_ADDRESS) segítségével. Ha a válasz sikeres, visszaadja a válasz tartalmát stringként. Ha kivétel lép fel (pl. hiba az API elérésében), visszaadja a null értéket.
- **OnStoryDoubleClick** metódus: Ez a metódus akkor hívódik meg, amikor a felhasználó duplán kattint egy történetre a listanézetben. Megkapja a kiválasztott Story objektumot a listanézetből. Létrehoz egy StoryReader ablakot a kiválasztott történet részleteinek megjelenítéséhez. Beállítja a MainWindow ablakot a StoryReader ablak tulajdonosának. Megjeleníti a StoryReader ablakot.

StoryReader.xaml.cs

A StoryReader osztály egy külön ablakot biztosít a felhasználó számára, ahol elolvashatja a kiválasztott történet teljes tartalmát. Lehetővé teszi a felhasználó számára a háttérzaj lejátszását/szüneteltetését és a hangerő szabályozását.

- **PlaySound_Click** metódus: Ez a metódus akkor hívódik meg, amikor a felhasználó a "Lejátszás" gombra kattint. Elindítja a RelaxingSoundPlayer lejátszását.
- **PauseSound_Click** metódus: Ez a metódus akkor hívódik meg, amikor a felhasználó a "Szünet" gombra kattint. Megállítja a RelaxingSoundPlayer lejátszását.

- **VolumeSlider_ValueChanged** metódus: Ez a metódus akkor hívódik meg, amikor a felhasználó megváltoztatja a hangerő csúszka értékét. Beállítja a RelaxingSoundPlayer hangerejét a csúszka új értékére.

Elvégzett tesztek

Áttekintés

Ez a dokumentáció az Inkli projekt tesztjeit mutatja be, amelyek két fő csoportba rendezve ellenőrzik az alkalmazás adatbázisának és webes elérhetőségének állapotát. A leírásból megtudható, hogy milyen feltételeket vizsgálnak a tesztek, miért fontosak, és hogyan futtathatók.

Tesztcsoportok összefoglalása

Tesztcsomag	Célja
DatabaseTest	Ellenőrzi az adatbázis tábláinak meglétét, adatműveletek sikerességét
WebsiteTest	Ellenőrzi a webes útvonalak (route-ok) elérhetőségét és válaszkódját

Tesztek futtatása

A tesztek hibátlan futásához szükséges futtani egy másik terminálban a *composer run dev* parancsot

A teljes tesztcsomag futtatása:

- php artisan test

Csak az adatbázis-tesztek:

- php artisan test --filter DatabaseTest

Csak a webes elérhetőség-tesztek:

- php artisan test --filter WebsiteTest

Részletes tesztleírások

DatabaseTest.php

1. **checks if there is data in the users table**
    - **Mit vizsgál?**: Megnézi, hogy létezik-e legalább egy rekord a users táblában.
    - **Miért fontos?**: Biztosítja, hogy a felhasználói adatok inicializálva legyenek, és a migrációk, seedelés helyesen futottak.

2. **checks if there is data in the genres table**

- **Mit vizsgál?**: Ellenőrzi, hogy a genres tábla nem üres.
- **Miért fontos?**: A történetek besorolásához alapértelmezett műfajok szükségesek.

3. **checks if there is data in the stories table**
   - **Mit vizsgál?**: Ellenőrzi, hogy a stories tábla tartalmaz-e rekordot.
     Miért fontos?: Biztosítja, hogy legalább egy történet elérhető legyen az alkalmazásban.

4. **checks if data can be added to the stories table**
   - **Mit vizsgál?**: Tesztelendő új történet létrehozása, majd keresés a title alapján.
   - **Miért fontos?**: Validálja a create művelet működését és az adatbázis-írást.

5. **checks if data can be added to the users table**
   - **Mit vizsgál?**: Új felhasználó létrehozása és ellenőrzése email alapján.
   - **Miért fontos?**: A felhasználói regisztrációs folyamat alapfunkciójának tesztelése.

6. **checks if data can be added to the genres table**
   - **Mit vizsgál?**: Új műfaj létrehozása és ellenőrzése név alapján.
   - **Miért fontos?**: Biztosítja, hogy a műfajok CRUD-műveletei helyesen működjenek.

WebsiteTest.php

1. **checks if the landing page is accessible**
   - **Mit vizsgál?**: GET / kérés válaszkódja 200.
   - **Miért fontos?**: Az alapoldal elérhetőségének ellenőrzése.

2. **checks if the about us page is accessible**
   - **Mit vizsgál?**: GET /aboutus válaszkód 200.
   - **Miért fontos?**: A bemutatkozó oldal működőképességének tesztelése.

3. **checks if the privacy policy page is accessible**
   - **Mit vizsgál?**: GET /privacy-policy válaszkód 200.
   - **Miért fontos?**: Adatvédelmi tájékoztató oldal elérhetőségének ellenőrzése.

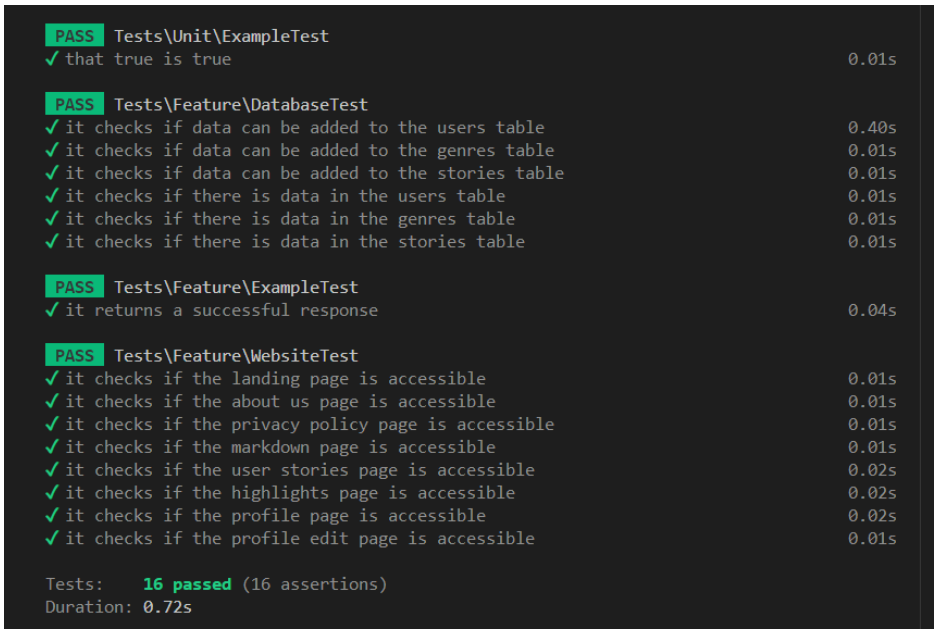
4. **checks if the markdown page is accessible**
   - **Mit vizsgál?**: GET /markdown válaszkód 200.
   - **Miért fontos?**: Markdown-szerkesztő oldal tesztelése.

5. **checks if the user stories page is accessible**
   - **Mit vizsgál?**: Bejelentkezett felhasználó GET /my-stories kérésének ellenőrzése.
   - **Miért fontos?**: Saját történetek oldal hozzáférési jogosultságának ellenőrzése.

6. **checks if the highlights page is accessible**
   - **Mit vizsgál?**: Bejelentkezett felhasználó GET /highlights sikeressége.
   - **Miért fontos?**: Kiemelt tartalmak oldalának hozzáférhetősége.

7. **checks if the profile page is accessible**
    - **Mit vizsgál?**: Bejelentkezett felhasználó GET /profile válasza 200.
    - **Miért fontos?**: Felhasználói profil oldala elérhetőségének tesztelése.

8. **checks if the profile edit page is accessible**
    - **Mit vizsgál?**: Bejelentkezett felhasználó GET /profile/edit válasza 200.
    - **Miért fontos?**: Profil szerkesztő felület működőképességének biztosítása.



A lefuttatott tesztek