***Szakképesítés neve:*** Szoftverfejlesztő és –tesztelő

***Szakmajegyzék száma:*** 506131203

**Záródolgozat**

**Állatorvosi rendelő adatkezelő program**

**Készítette:** Pákozdi Szimonetta Anikó

Békéscsaba, 2024.

**Felhasználói dokumentáció**

Az állatorvosi adatkezelő program egy összetett és rugalmas szoftver, melynek célja az állatorvosi rendelők hatékonyabbá tétele, az állatok egészségének és kezelésének optimalizálása, valamint az adminisztrációs feladatok egyszerűsítése. A program lehetővé teszi az állatorvosoknak és az asszisztenseknek a páciensek adatainak tárolását és kezelését. Ide tartoznak az állatok egészségügyi előzményei, kezelései, panaszai, adatai és egyéb fontos információk. A program rendelkezik fejlett keresési funkciókkal, amelyek lehetővé teszik az adatbázisban található információk gyors és hatékony elérését. Az állatorvosok könnyen megtalálhatják a szükséges ismereteket az adott állatokról.

**Rendszerkövetelmények**

1. **Hardverkövetelmények:**

* **Processzor (CPU):** Legalább 2 GHz-es vagy annál erősebb többmagos processzor javasolt.
* **Memória (RAM):** Minimum 8 GB RAM szükséges a program és a fejlesztőeszközök futtatásához.
* **Tárhely**: Legalább 1 GB szabad tárhely szükséges a Visual Studio Code, a Node.js, a React és az egyéb fejlesztői eszközök telepítéséhez és futtatásához.
* **Operációs Rendszer:** A program Windows rendszereken futtatható.
* **Grafikus Kártya:** Bár a program nem igényel nagy grafikus teljesítményt, az alapvető grafikus kártya javasolt a grafikus felhasználói felület megfelelő működéséhez.
* **Hálózati Kapcsolat:** Internet-hozzáférés szükséges a Node.js és a React függőségeinek letöltéséhez, valamint az adatbázis-kezelő rendszer eléréséhez és frissítéséhez.

1. **Szoftverkövetelmények:**

* **Operációs Rendszer:** A program Windows rendszereken futtatható.
* **Visual Studio Code:** A fejlesztéshez és szerkesztéshez szükséges a Visual Studio Code legfrissebb verziójának telepítése. A Visual Studio Code ingyenesen letölthető és telepíthető a hivatalos weboldalról.
* **Node.js és npm:** Az alkalmazás futtatásához és a projekt függőségeinek kezeléséhez szükséges a Node.js és a Node Package Manager (npm) telepítése.
* **React:** A program React alapú felhasználói felülettel rendelkezik, ezért szükséges a React könyvtár telepítése és konfigurálása a projektben.
* **XAMPP és phpMyAdmin:** Az adatbázis-kezeléshez és adatbázis-kapcsolat létrehozásához szükséges az XAMPP telepítése és konfigurálása a szerveroldali környezet biztosítása érdekében. A phpMyAdmin segítségével lehet adatbázisokat létrehozni, kezelni és módosítani.
* **Webböngésző:** A program teszteléséhez és futtatásához szükséges egy modern webböngésző, mint például a Google Chrome, Mozilla Firefox, vagy a Microsoft Edge.

**Telepítési útmutató**

**Visual Studio Code telepítése:**

* Látogasson el a Visual Studio Code hivatalos weboldalára.
* Töltse le a telepítőfájlt a megfelelő operációs rendszerhez.
* Indítsa el a telepítőfájlt, és kövesse az egyszerű telepítési lépéseket.

**Backend telepítése/elindítása:**

* Látogasson el a Node.js hivatalos weboldalára.
* Töltse le a telepítőfájlt a megfelelő operációs rendszerhez.
* Indítsa el a telepítőfájlt, és kövesse az egyszerű telepítési lépéseket.
* Ezt követően a terminálba a következőket kell beírni: npm i, npm start

**XAMPP telepítése és phpMyAdmin konfigurálása:**

* Látogasson el az XAMPP hivatalos weboldalára.
* Töltse le a telepítőfájlt a megfelelő operációs rendszerhez.
* Indítsa el a telepítőfájlt, és kövesse az egyszerű telepítési lépéseket.
* Nyissa meg az XAMPP vezérlőpultot, és indítsa el az Apache és MySQL szervereket.
* Nyisson meg egy webböngészőt, és látogasson el a phpMyAdmin címére (http://localhost/phpmyadmin).
* Bejelentkezés után importálja a *rendelo* adatbázist a fájlokból.

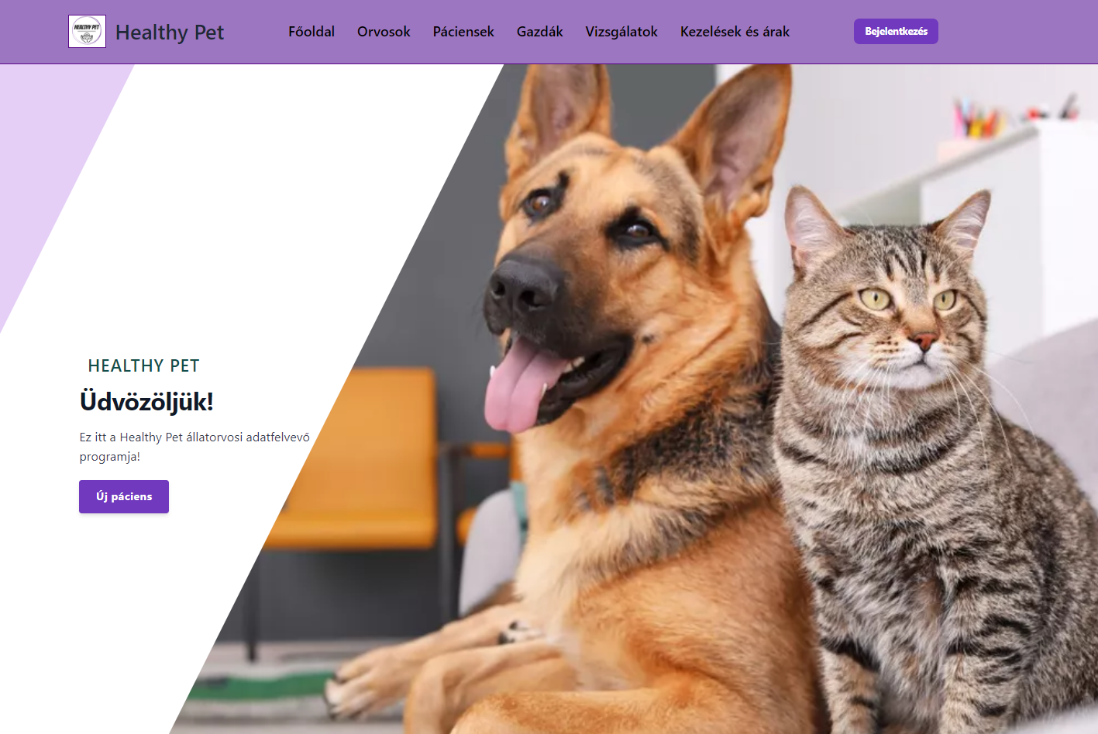
**Frontend elindítása:**

* Nyissa meg a parancssort a Visual Studio Code-ban a projekt gyökérkönyvtárában.
* Írja be a terminálba a következő parancsokat: npm i, npm run dev
* Ezt követően menjen a [http://localhost:1573-as](about:blank) oldalra, ahol megjelenik a frontend.

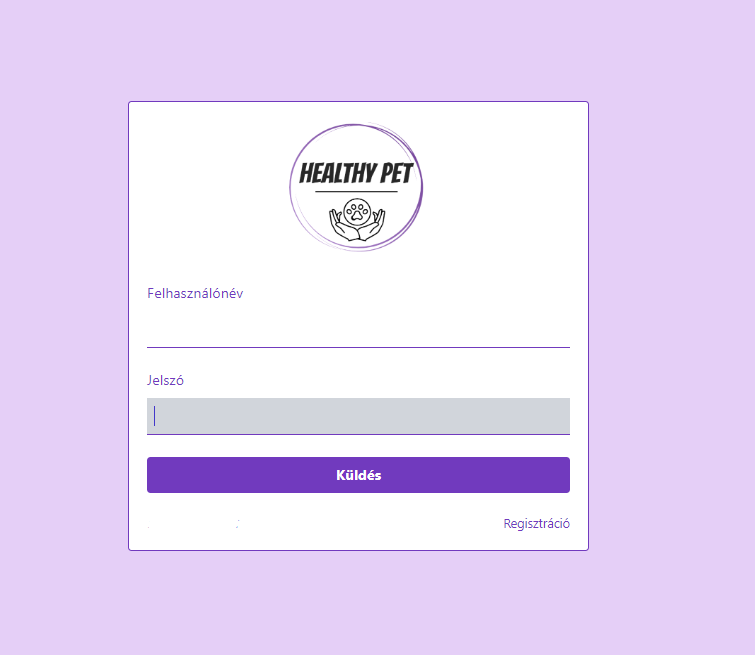
**Projekt klónozása és futtatása:**

* Klónozza le a projektet a GitHubról vagy más verziókezelő rendszerből.
* Nyissa meg a projektet a Visual Studio Code-ban.
* Nyissa meg a parancssort a projekt gyökérkönyvtárában.
* Telepítse a projekt függőségeit a következő paranccsal: npm install
* Indítsa el az alkalmazást a következő paranccsal: npm start (backend), npm run dev (frontend)

**A program használatának részletes leírása**

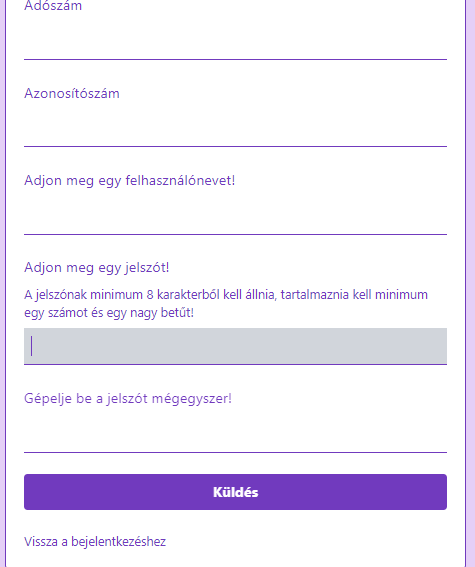
* **Főoldal:**

Ez a program fő oldala. A képen felül találhatóak a menüpontok (főoldal, orvosok, páciensek, gazdák, vizsgálatok, kezelések és árak) és egy bejelentkezési gomb (bejelentkezés), amivel az orvos tud bejelentkezni a rendszerbe. Addig nem tudunk semmit megnézni, amíg az orvos nincs belépve a programban. Ha ez megtörtént, akkor a gomb felirata kijelentkezés lesz, amivel pedig ki tud lépni az orvos. Középen láthatunk egy „Új páciens” feliratú gombot, amivel új állatot vehetünk fel a rendszerbe. Erre az oldalra fogunk visszatérni, ha új pácienst vettünk fel.

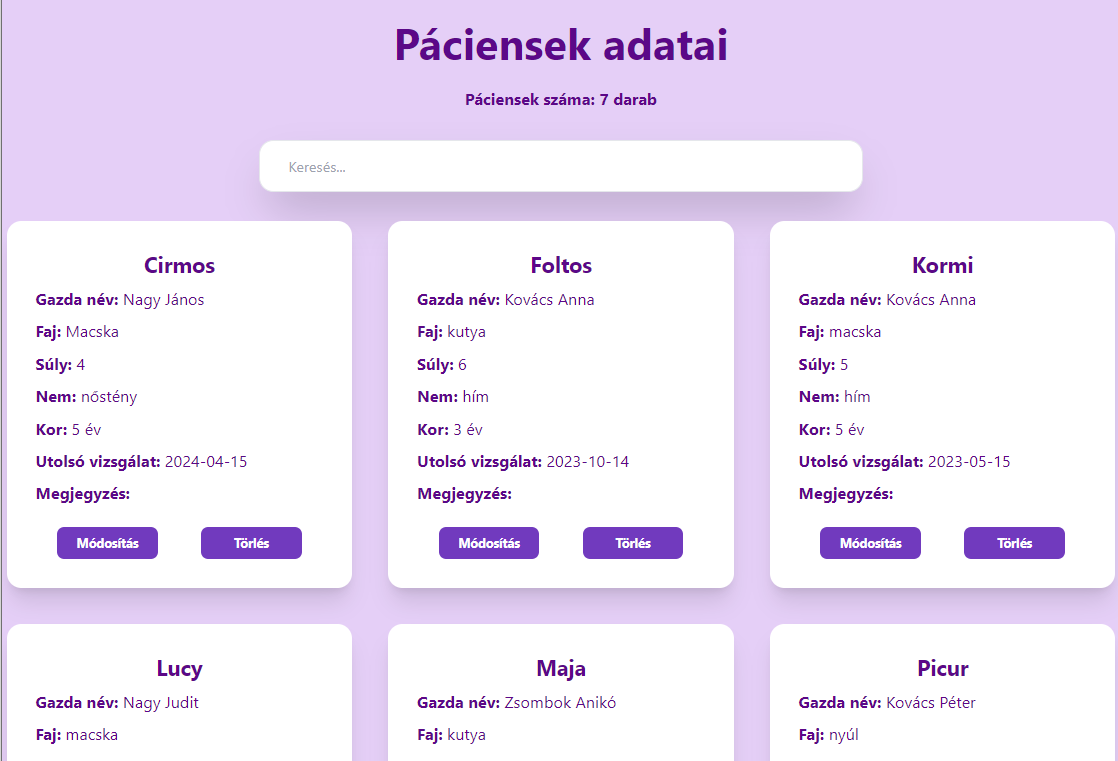
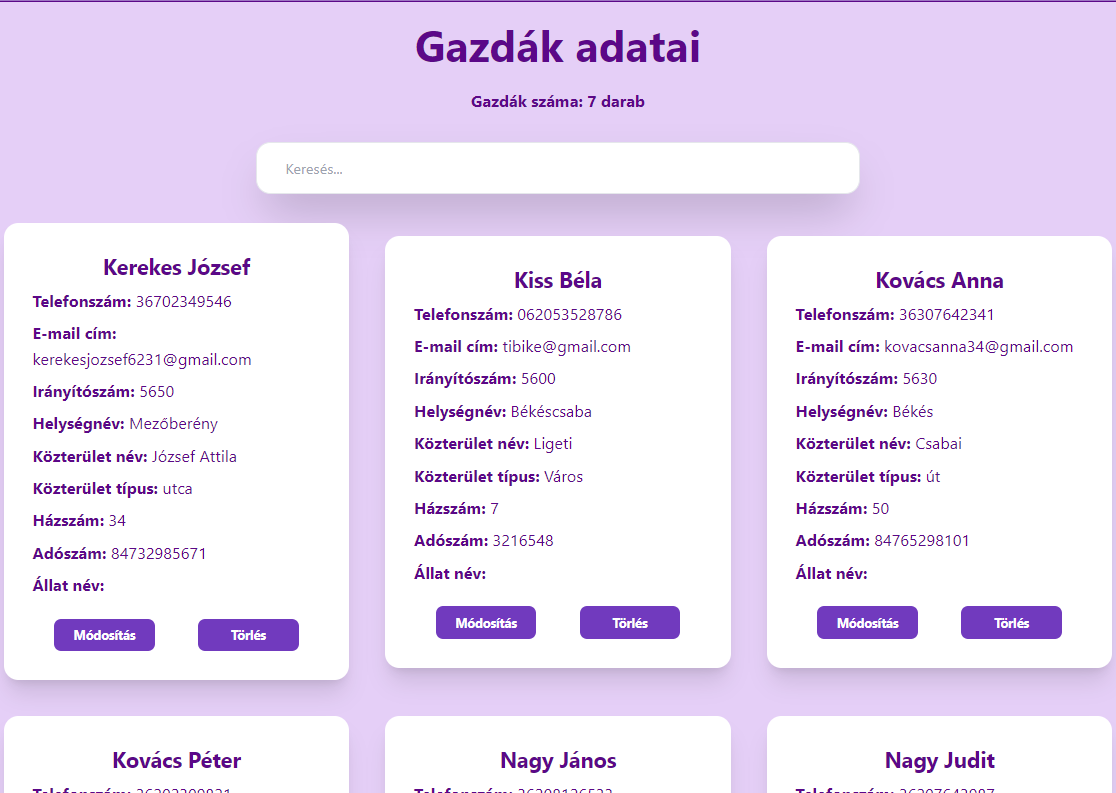
* **Bejelentkezés:**

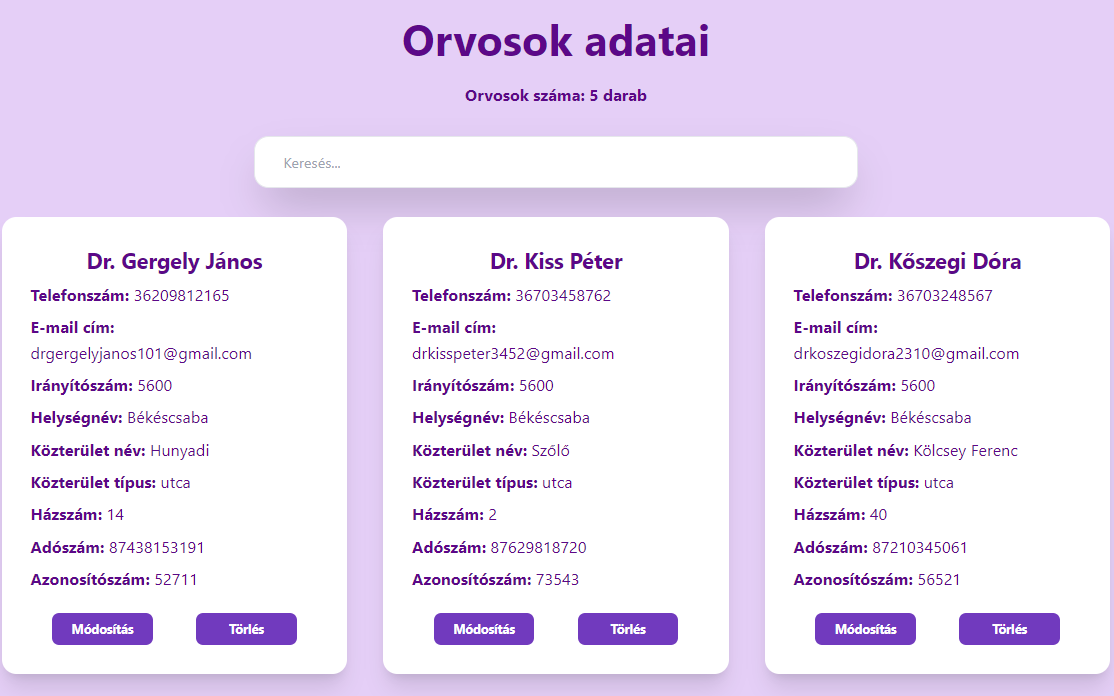
Ezt a felületet a bejelentkezés gombra kattintva láthatjuk. Meg kell adni az orvos nevét és jelszavát, majd a küldés gombra kattintani. Ha nincs regisztrálva az orvos, akkor a küldés gomb alatti regisztrációra kell kattintani.

* **Regisztráció:**



Az oldalon itt tudunk regisztrálni, ahol az orvos adatait kell megadni, majd a küldéssel elmenteni. A jelszót kétszer kell megadni, hogy biztos ne történjen az, hogy elírjuk a jelszót. Ha csak véletlen kattintottunk a regisztációs felületre, akkor a „vissza a bejelentkezéshez” feliratra kattintva visszajutunk az előző oldalhoz.

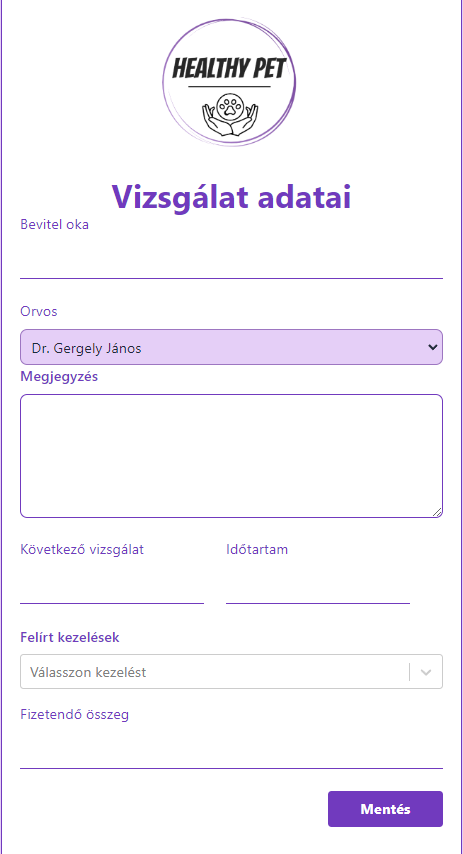
* **Menüpontok:**





Ezeket az oldalakat láthatjuk, ha rákattintottunk a menüpontokra. A pácienseknél a páciensek adatait látjuk, az orvosoknál az orvosok adatait, a gazdáknál a gazdák adatait, a vizsgálatoknál a vizsgálatok adatait, a kezelések és áraknál pedig a kezeléseket és a hozzájuk tartozó árakat. Az orvosok, gazdák és páciensek adatait tudjuk szerkeszteni és törölni is a módosítás és törlés gombokra nyomva. A keresés mezőbe az orvos nevét, a páciens nevét, vagy a gazda nevét kell beírni, hogy megjelenítse az adatokat. A címsor alatt pedig láthatjuk, hogy hány gazdát, orvost és pácienst tárol a program. A vizsgálatnál láthatjuk a kezelés dátumát, a kezelt állatot, az orvos nevét, a bevitel okát, a megjegyzést, a kezelés időtartamát, a kezelés nevét, a következő vizsgálatot és a használt kezelések árát.

* **Új páciens felvétele:**



Ezeken a felületeken tudjuk megadni azt új pácienssel kapcsolatos adatokat. Először a gazda adatait kell megadni, majd az állat adatait ezt követően pedig a vizsgálat adatait. Az állat adatainál a nemnél tudunk választani 3 opció közül: hím, nőstény vagy ismeretlen. Az utolsó vizsgálatnál pedig naptárral tudjuk beírni az adatot. Az orvosnál választhatunk a programba regisztrált orvosok közül. A felírt kezeléseknél pedig rugalmasan több kezelést ki tudunk választani, ami közben összeadja a kiválasztott kezelések összegeit. A felületek között gombokkal lehet tovább menni, majd a végén a mentés gombbal kell érvényesíteni a bevitt adatokat, amit a program hozzáad az adatbázishoz.

**Fejlesztői dokumentáció**

**Témaválasztás indoklása**

A saját macskáim ellátása során tapasztaltam, hogy az állatorvosi rendelőkben való adatkezelés és nyomon követés kulcsfontosságú tényező az állatok hatékony ellátásában. A személyes tapasztalataim alapján látom, hogy az állatorvosi rendelőkben rendszerint számos kihívás áll fenn az adatok kezelése terén, például a betegadatok pontatlansága, a dokumentáció elavultsága vagy az adminisztratív terhek túlzott mértéke. Ezek a tényezők megnehezítik az állatok egészségének megfelelő kezelését és a rendelők hatékony működését.

Az imént említettek ösztönöztek arra, hogy egy olyan projektet készítsek, amely segít megoldani ezeket az akadályokat. Az adatkezelő rendszer fejlesztésével lehetőséget láttam arra, hogy olyan eszközt hozzak létre, amely segíti az állatorvosokat és az asszisztenseket a hatékonyabb munkavégzésben.

Az adatkezelő rendszer fejlesztése során célom az is, hogy összehangoljam az állatorvosok és az asszisztensek munkáját, és a program lehetőséget teremtsen a hatékonyabb kommunikációra és együttműködésre a betegek ellátása során.

A személyes tapasztalataim mellett fontosnak tartottam, hogy a projekt valóban értéket képviseljen az állatorvosok, az állatorvosi rendelők és természetesen az állatok és gazdák számára egyaránt.

**Az alkalmazott fejlesztői eszközök**

**Adatbázis:**

* **Programozási nyelv:** MySQL
* **Fejlesztési felület:** XAMPP-PhpMyAdmin

A MySQL, ahogy a neve is mutatja, egy SQL-alapú adatbázis-kezelő rendszer vagy DBMS. A nyílt forráskódú szoftvert jelenleg az Oracle, a Java programozási nyelvet is kifejlesztő vállalat tartja karban. A MySQL képes az adatok táblázatokban történő tárolására, kezelésére és megjelenítésére. Míg az adatbázis szerverként működik, amelyen minden lényeges információ tárolódik, addig a szoftver kliensnek tekinthető. A szoftver segítségével a vonatkozásos adatbázis felhasználói különböző lekérdezéseket fogalmazhatnak meg az SQL lekérdezési nyelven, és küldhetik el azokat az adatbázis-rendszernek. Ezeket aztán a MySQL feldolgozza, ezért az adathozzáférés is fontos eleme a MySQL-nek.

**Backend:**

* **Fejlesztési környezet:** Visual Studio Code
* **Programozási nyelv:** JavaScript
* **Keretrendszer:** Express – Node.js

**Visual Studio Code:**

A Visual Studio Code a Microsoft cég által fejlesztett ingyenes és nyílt forráskódú szövegszerkesztő. Támogatja az automatikus kódkiegészítést, rendelkezik beépített hibakeresővel, és Git verziókezelő támogatással, és még számos egyéb szolgáltatással. Bővítmények segítségével további szolgáltatások és funkciók adhatóak hozzá.

**Javascript:**

A JavaScript (röviden JS) egy programozási nyelv, amelyet kifejezetten az internetre fejlesztettek ki. A legtöbb webböngésző szoftver, és a modern okostelefonok is mind támogatják a JavaScriptet.

A JavaScriptet első sorban arra használják, hogy gazdagabb, felhasználóbarát élményeket teremtsenek vele az internetet böngészők számára, például dinamikusan frissülő weboldalakat, intuitív felhasználói felületeket, menüket, párbeszédpaneleket, 2D-s és 3D-s grafikákat, interaktív térképeket, videólejátszókat, és számos egyéb elemet, illetve funkciót. A JavaScript ilyesfajta, webböngészőkben történő alkalmazását kliensoldalú JavaScriptnek szokás nevezni.

A JavaScript az internetes háromszög egyik eleme, a másik kettő pedig a CSS illetve a HTML. A HTML feladata a weboldal leírása (szövegek, grafikák, stb.), a CSS pedig a weboldal megjelenéséért felel. A JavaScript nem esszenciális, de a fent említett hasznos funkciók miatt nagyon fontos eleme a webdesignnak, hiszen dinamikus weboldalakat lehet létrehozni vele, és lehetővé teszi a programozható elemek elhelyezését is ezeken a weboldalakon.

**Node.js**

A Node.js egy nyíltforrású, többplatformos, szerveroldali JavaScript futtatókörnyezet (JavaScript runtime environment) a Google Chrome V8 JavaScript-motoron alapulva. Leginkább webes applikációk, dinamikus weboldalak fejlesztésére használják, de akár szerveroldali feldolgozó szkriptek készítésére is kiválóan alkalmas.

Segítségével megvalósítható a "JavaScript everywhere" paradigmája, azaz, hogy a fejlesztők mind a szerveroldali (back-end), mind a kliensoldali (front-end) programkódokat egyazon programozási nyelvvel alkothassák meg.

A fejlesztők többnyire a futtatókörnyezet eseményvezérelt arhitektúrájának lehetőségeit aknázzák ki, amellyel aszinkron I/O-ra képes webes alkalmazások készíthetőek. Ennek a lényege röviden összefoglalva annyi, hogy a programkód egyes blokkjai akár egyszerre is legenerálhatják a kért weboldalt, szolgáltatják a kliens oldal számára az adatokat egy-egy grafikai elem megrajzolásához, ezáltal valódi real-time alkalmazásokat fejleszthetünk vele.

**Frontend**

* **Fejlesztési környezet:** Visual Studio Code
* **Programozási nyelv:** JavaScript
* **Keretrendszer:** React
* **Egyéb:** Tailwind

**React:**

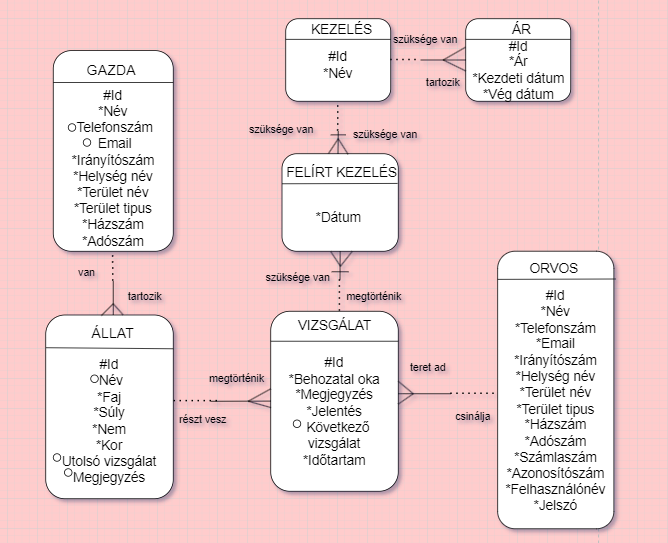
A React egy relatíve fiatal, objektumorientált nyelv, amit a Microsoft fejlesztett ki a Sun Microsystems-szel folytatott per következményeként, hogy legyen egy saját nyelvük a Java ellen/mellett. A React egy JavaScript keretrendszer, amely lehetővé teszi a felhasználói felületek készítését webalkalmazásokhoz. A React keretrendszer segítségével a JavaScript nyelvet használva lehet készíteni felhasználói felületeket és webalkalmazásokat. A ReactJS segítségével épített alkalmazások az egyszeri újrafelhasználhatóság elvét alkalmazzák. Ez arra az elvre épül, hogy a weboldalak és alkalmazások komponensekből, egyirányú folyamattal épüljenek fel. A Reactben létezik az állapotok és a megváltoztathatatlanság fogalma. A szülő és a gyermek komponensek tekintetében a komponensek szigorú hierarchiával rendelkeznek. A React esetében egy komponens egy meglévő kódrészlet, amely megírása a tiszta kód (Clean Code) elvén alapult.

**Tailwind:**

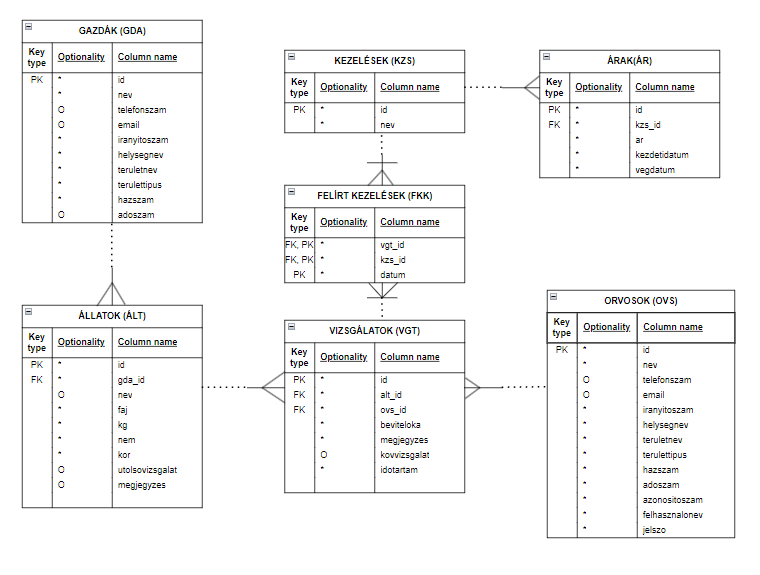
A Tailwind CSS lehetővé teszi a gyorsabb fejlesztést, mivel a kész stílusosztályokat lehet használni a HTML elemekhez. Ezáltal nem kell egyesével meghatározni saját CSS osztályokat, hanem egyszerűen alkalmazhatja a Tailwind osztályokat a megfelelő elemekre, megspórolva rengetek időt.

**Adatmodell leírása:**

**Logikai modell:**

A logikai adatmodell absztrakt nézetet nyújt az adatokról és azok kölcsönös kapcsolatairól. Ez a modell segít az üzleti logika és folyamatok pontosabb megértésében, anélkül hogy részletesen meg kellene ismerni az adatbázis fizikai részét. Ezáltal lehetővé teszi az üzleti igények pontosabb azonosítását és megértését, ami hozzájárulhat az adatbázis hatékonyabb tervezéséhez és kialakításához.

**Fizikai modell:**

A fizikai adatmodell a logikai adatmodell konkrét részét írja le, vagyis hogyan kerülnek az adatok ténylegesen tárolásra és strukturálásra az adatbázisban. Ez magában foglalja az adattípusokat, indexeket, kulcsokat és más technikai részleteket. A fizikai adatmodell segít optimalizálni az adatbázist a teljesítmény és hatékonyság szempontjából, valamint lehetővé teszi a rendszer terhelésének és skálázhatóságának kezelését.

**Tábla sorainak értelmezése:**

**gazdak:**

* **id:** Ez az oszlop az adott gazda egyedi azonosítóját tartalmazza. Minden gazda egyedi azonosítóval rendelkezik, ami segíti azonosításukat az adatbázisban.
* **nev**: Itt tároljuk a gazda nevét. Ez segít azonosítani és kapcsolódni a gazdához tartozó adatokhoz.
* **telefonszam:** A gazda telefonszámát tároljuk ebben az oszlopban, hogy elérhető legyen számunkra. Nem kötelező, mivel nem feltétlen van mindenkinek telefonszáma.
* **email:** Ebben az oszlopban tároljuk a gazda email címét, amely egy másik kapcsolattartási lehetőség. Nem kötelező, mivel nem feltétlen van mindenkinek email címe.
* **iranyitoszam:** A gazda lakóhelyének irányítószámát tartalmazza az oszlop. Ez segít a gazda fizikai helyének azonosításában.
* **helysegnev:** A gazda lakóhelyének városának vagy településének nevét tároljuk ebben az oszlopban.
* **teruletnev:** Ez az oszlop a gazda lakóhelyének területét tartalmazza, ami segíthet a különböző területek közötti megkülönböztetésben.
* **terulettipus:** A terület típusát tároljuk ebben az oszlopban.
* **hazszam:** A gazda lakóhelyének pontos címét, házszámát tároljuk ebben az oszlopban.
* **adoszam:** A gazda adószámát tároljuk ebben az oszlopban, ami fontos lehet a későbbi adminisztratív feladatokhoz. Ezt csak akkor szükséges megadni, ha számlát szeretnénk róla.

**allatok**:

* **id:** Ez az oszlop tartalmazza az adott állat egyedi azonosítóját. Minden állatnak egyedi azonosítóval kell rendelkeznie az adatbázisban.
* **gda\_id**: Ez az oszlop az állat gazdájának azonosítójára hivatkozik a gazdák táblából, így összekapcsolva az állatokat a gazdáikkal.
* **nev:** Az állat nevét tároljuk ebben az oszlopban, ami segíti az azonosítást és az említésüket a gazdájukhoz. Nem szükséges megadni, mert nem feltétlen tudjuk az állat nevét.
* **faj**: Az állat faját (pl. kutya, macska stb.) tartalmazza ez az oszlop.
* **kg:** Az állat súlyát tároljuk itt, ami fontos lehet az állatorvosi kezelések során.
* **nem:** Az állat nemét (pl. hím, nőstény) tároljuk ebben az oszlopban.
* **kor:** Az állat életkorát vagy születési időpontját tartalmazza ez az oszlop.
* **utolsovizsgalat:** Az állat utolsó vizsgálatának időpontját tárolja, ami fontos információ lehet az állapotuk megítéléséhez.
* **megjegyzes:** Ebben az oszlopban további megjegyzéseket vagy információkat lehet tárolni az állatokkal kapcsolatban.

**orvosok:**

* **id:** Ez az oszlop tartalmazza az adott állatorvos egyedi azonosítóját. Minden orvosnak egyedi azonosítóval kell rendelkeznie az adatbázisban.
* **nev:** Az orvos nevét tároljuk ebben az oszlopban, ami segíti az azonosítást és a kapcsolódást az orvoshoz.
* **telefonszam:** Az orvos telefonszámát tároljuk ebben az oszlopban, hogy elérhető legyen számunkra.
* **email:** Az orvos email címét tároljuk ebben az oszlopban, ami egy másik kapcsolattartási lehetőség.
* **iranyitoszam:** Az orvos lakóhelyének irányítószámát tartalmazza az oszlop. Ez segíti az orvos fizikai helyének azonosítását.
* **helysegnev:** Az orvos lakóhelyének városának vagy településének nevét tároljuk ebben az oszlopban.
* **teruletnev:** Ez az oszlop az orvos lakóhelyének területét tartalmazza, ami segíthet a különböző területek közötti megkülönböztetésben.
* **terulettipus:** A terület típusát tároljuk ebben az oszlopban.
* **hazszam:** Az orvos lakóhelyének pontos címét, házszámát tároljuk ebben az oszlopban.
* **adoszam:** Az orvos adószámát tároljuk ebben az oszlopban, ami fontos lehet a későbbi adminisztratív feladatokhoz.
* **azonositoszam:** Az orvos egyedi azonosító számát tároljuk ebben az oszlopban.
* **felhasznalonev:** Az orvos felhasználónevét tároljuk ebben az oszlopban, amely a rendszerbe való bejelentkezéshez szükséges.
* **jelszo:** Az orvos jelszavát tároljuk ebben az oszlopban, biztonsági okokból általában titkosítva.

**vizsgalatok:**

* **id:** Ez az oszlop tartalmazza az adott vizsgálat egyedi azonosítóját. Minden vizsgálatnak egyedi azonosítóval kell rendelkeznie az adatbázisban.
* **alt\_id:** Ez az oszlop az adott vizsgálat alatt álló állat azonosítójára hivatkozik az állatok táblából.
* **ovs\_id:** Ez az oszlop az adott vizsgálatot elvégző orvos azonosítójára hivatkozik az orvosok táblából.
* **beviteloka:** Az oszlopban tároljuk, hogy miért szorul kezelésre az állat.
* **megjegyzes:** Az oszlopban további megjegyzéseket vagy információkat lehet tárolni pl. az állat magatartásával kapcsolatban.
* **kovvizsgalat:** Az oszlopban tároljuk a következő vizsgálat tervezett időpontját vagy dátumát.
* **idotartam:** Az oszlopban tároljuk, hogy a vizsgálat mennyi ideig tartott.

**kezelesek:**

* **id:** Ez az oszlop tartalmazza az adott kezelés egyedi azonosítóját. Minden kezelésnek egyedi azonosítóval kell rendelkeznie az adatbázisban.
* **nev:** Az oszlopban tároljuk a kezelés nevét.

**felirt\_kezelesek:**

* **vgt\_id:** Ez az oszlop az adott vizsgálat egyedi azonosítójára hivatkozik a vizsgálatok táblából.
* **kzs\_id:** Ez az oszlop az adott kezelés egyedi azonosítójára hivatkozik a kezelesek táblából.
* **datum:** Az oszlopban tároljuk a kezelés felírásának dátumát.

**arak:**

* **id:** Ez az oszlop tartalmazza az adott ár egyedi azonosítóját. Minden árnak egyedi azonosítóval kell rendelkeznie az adatbázisban.
* **kzs\_id:** Ez az oszlop az adott kezelés egyedi azonosítójára hivatkozik a kezelesek táblából.
* **ar:** Az oszlopban tároljuk az adott kezeléshez tartozó árat.
* **kezdetidatum:** Az oszlopban tároljuk az ár érvényességének kezdeti dátumát.
* **vegdatum:** Az oszlopban tároljuk az ár érvényességének végdátumát.

**Táblakapcsolatok:**

**gazdak és allatok táblák közötti kapcsolat:**

Az allatok tábla egy idegen kulcsot tartalmaz, amely a gazdák táblában lévő egyedi azonosítóra hivatkozik (gda\_id az allatok táblában). Ez a kapcsolat jelzi, hogy az adott állatokhoz tartozik egy gazda a gazdak táblából.

**orvosok és vizsgalatok táblák közötti kapcsolat:**

A vizsgalatok tábla egy idegen kulcsot tartalmaz, amely az orvosok táblában lévő egyedi azonosítóra hivatkozik (ovs\_id a vizsgalatok táblában). Ez a kapcsolat jelzi, hogy az adott vizsgálatot elvégezte egy orvos az orvosok táblából.

**allatok és vizsgalatok táblák közötti kapcsolat:**

A vizsgalatok tábla egy idegen kulcsot tartalmaz, amely az állatok táblában lévő egyedi azonosítóra hivatkozik (alt\_id a vizsgalatok táblában). Ez a kapcsolat jelzi, hogy az adott vizsgálat azonosított állathoz tartozik az allatok táblából.

**vizsgalatok és felirt\_kezelesek táblák közötti kapcsolat:**

A felirt\_kezelesek tábla egy idegen kulcsot tartalmaz, amely az adott vizsgálatok egyedi azonosítójára hivatkozik (vgt\_id a felirt\_kezelesek táblában). Ez a kapcsolat jelzi, hogy az adott kezeléseket felírták egy vizsgálat során a vizsgalatok táblából.

**kezelesek és felirt\_kezelesek táblák közötti kapcsolat:**

A felirt\_kezelesek tábla egy idegen kulcsot tartalmaz, amely az adott kezelések egyedi azonosítójára hivatkozik (kzs\_id a felirt\_kezelesek táblában). Ez a kapcsolat jelzi, hogy az adott felírt kezelések egy meglévő kezelést jelentenek a kezelesek táblából.

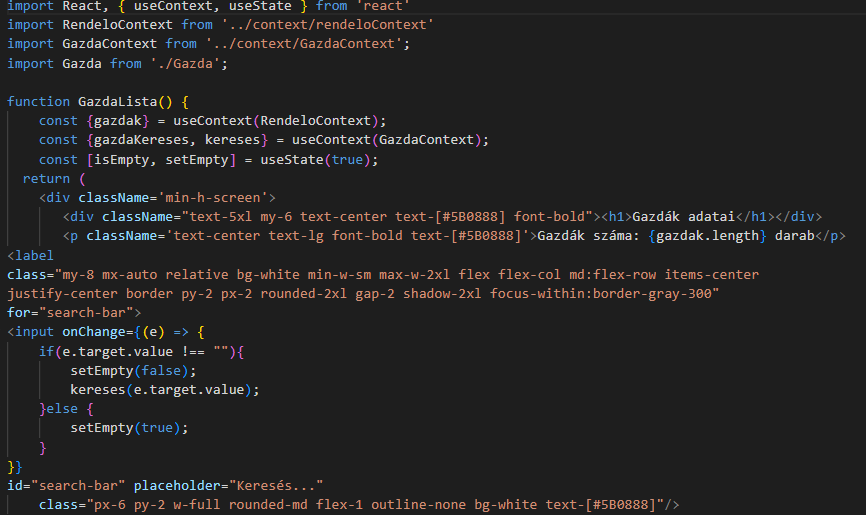
**kezelesek és arak táblák közötti kapcsolat:**

Az arak tábla egy idegen kulcsot tartalmaz, amely az adott kezelések egyedi azonosítójára hivatkozik (kzs\_id a arak táblában). Ez a kapcsolat jelzi, hogy az arak egy kezeléshez tartozó árat jelentenek a kezelesek táblaból.

**Fő kódrészek bemutatása:**

**Frontend:   
App.jsx:**

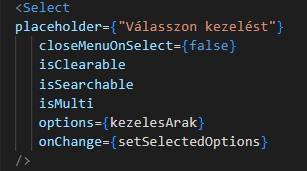
Az App.jsx az alkalmazás fő keretrendszerét tartalmazza, amely a különböző alkomponensek összehangolását és megjelenítését végzi. Ez az alkalmazás fő ablaka és fő navigációs keret is.

**Componens:** 

A React-ben a komponens egy olyan rész, ami megjelenít egy darabot a weboldalról vagy alkalmazásról. Ez lehet egy fejléc, egy gomb vagy akár egy teljes oldal is. A komponensek segítenek a kód tisztán tartásában és újrafelhasználásban, és lehetnek egymásba ágyazva. A programban több componens van, mivel több különböző adatot kell a programnak megjelenítenie. Ezeket a komponenseket a fentebb említett App.jsx-ben kellett meghívni. Itt lehet az oldal kinézetét állítani, Tailwind komponenseket beilleszteni, de lehet függvényeket is írni a különböző funkciókhoz.

**Context:**

Ha olyan adatok vannak, amelyeket számos komponensnek meg kell osztania, a Context segítségével könnyen megoszthatók az adatok az alkalmazás különböző részei között. A programban a localhost elérése itt lett megvalósítva fetch segítségével.

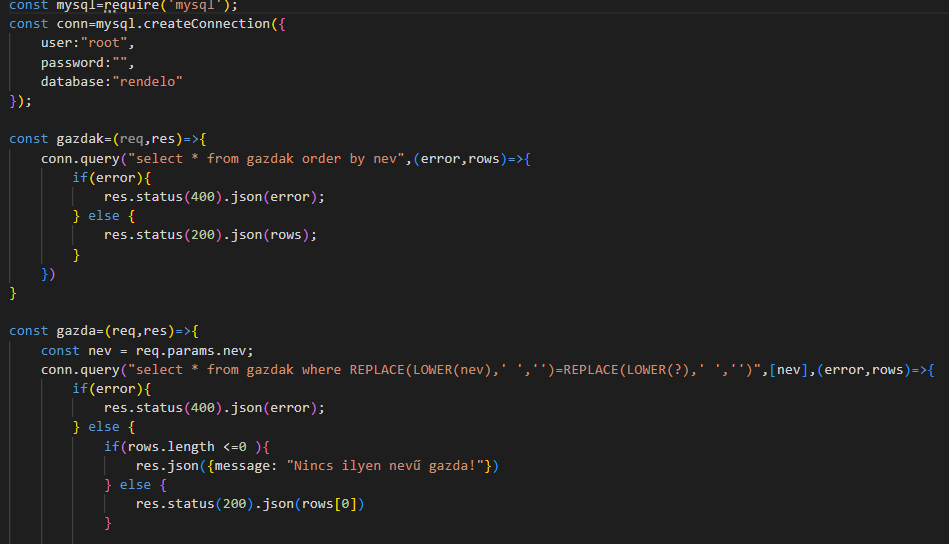
**Példa egy komponensre:**

Ez egy select komponens, ahol egy legördülő menüben tudjuk kiválasztani az adatbázisban tárolt kezeléseket. Ezekre kereséssel is rá tudunk keresni a listában, de görgetéssel is egyaránt megoldható. Ezt követően pedig a kiválasztott értéket hozzá adja egy listához.

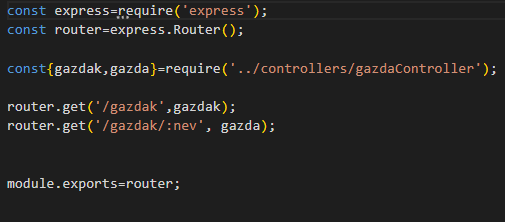
**Backend:**

**Index.js:**

Az index.js a Node.js backend programrésznek a fő fájlja, Beállítja az alkalmazás konfigurációját, beállítja a modulokat, definiálja az útvonalakat és elindítja az alkalmazást. Itt lehet megadni, hogy a backend milyen porton fusson, ami jelen esetben a localhost:8000-as portja. Mind ezek mellett beállíthatjuk a főoldalon megjelenő üzenetet/címet és a consol.log üzenetét, amit hiba nélküli elindításnál láthatunk.

**Controller:**

A controller fogadja a kéréseket a kliensektől, és meghatározza, hogy melyik funkciót kell végrehajtani az adott kérésre. Ezt követően egy adatbázist lekérdezést kell írni, attól függően, hogy milyen adatokat szeretnék visszakapni. Majd meg kell adni, hogy milyen választ adjon a backend az adott lekérdezésre hiba és hiba nélküli esetre is. A projektben itt van meghívva az adatbázis is.

**Routes:**

Az útvonalak szerepe az, hogy azonosítsák a végpontokat, amelyekhez a kliensek hozzáférhetnek. Amikor kérést külünk egy adott útvonalra, a szerver meghatározza az útvonalhoz tartozó kontrollert és végrehajtja a kérés feldolgozását.

**User\_login:**

A bejelentkezés JWT tokenes hitelesítéssel van megvalósítva. Ez egy nyílt szabvány által definiált token formátum, amelyet az információk biztonságos átvitelére és hitelesítésére használnak. A felhasználó beírja a felhasználónevét és jelszavát a bejelentkezési űrlapba, majd elküldi azt a backendnek. A backend fogadja a bejelentkezési kérelmet, és ellenőrzi a megadott felhasználói azonosítót és jelszót. Ha az adatok megfelelnek a backend adatbázisában tárolt adatoknak, akkor a bejelentkezés sikeresnek tekinthető. Amint a felhasználó sikeresen bejelentkezett, a backend létrehoz egy JWT tokent. Ha a token érvényes és a felhasználónak megfelelő jogosultságai vannak, akkor a backend feldolgozza a kérést és válaszol a kliensnek a kért adatokkal vagy művelettel. Emellett a program mysql2-őt is használ. Ez különösen előnyös a program számára, amelyek aszinkron működésre épülnek, mivel támogatja a Promise alapú interfészt, ami lehetővé teszi a könnyebb és hatékonyabb aszinkron műveleteket az adatbázissal való kommunikáció során.

**Kódrészlet magyarázata:**

* **Hiányzó adatok:** Ezt az üzenetet kapjuk, ha a nincs se a felhasználó megadva, se a jelszó megadva.
* **Felhasználó nem található:** Ha az adatbázisban nem létezik ilyen email-cím.
* **Nem megfelelő jelszó:** Ha a jelszó nem megfelelő a regisztrált jelszóval.
* **Sikeres bejelentkezés:** A bejelentkezési kérés végrehajtásának része, ahol a sikeres bejelentkezés után a felhasználónak visszaküldik a generált tokent, amelyet aztán a kliens később használhat más kérések autentikálásához. A token azonosítja a felhasználót, és lehetővé teszi számára, hogy hozzáférjen a program további részeihez, amit bejelentkezés nélkül nem láthat.

**Tesztelési dokumentáció**

**Cypress:**

A Cypress egy népszerű, modern és kifinomult tesztelési keretrendszer, amelyet elsősorban webes alkalmazások automatizált tesztelésére terveztek. A Cypress segítségével lehetőség van a webalkalmazások működésének ellenőrzésére, a felhasználói élmény tesztelésére és a hibák felderítésére az alkalmazás különböző részein.

**Cypress telepítése és használata:**

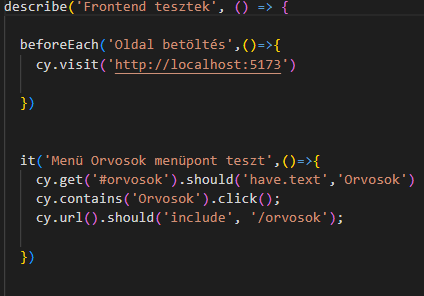
* Be kell írni egy külön terminálba a következőt: npm cypress –save-dev
* A megnyitás pedig a következő paranccsal megnyílik: npx cypress open
* Ezután megnyílik egy külön ablakban a Cypress, ahol ki kell választani milyen tesztelési típust szeretnénk. Itt két félét lehet választani: E2E Testing, Component Testing. Mi az elsőt választottuk.
* Majd ki kell választani, hogy milyen böngészőn szeretnénk tesztelni, ahol mi a Chrome mellett döntöttünk.
* E2E tesztnél a megnyílt felületen a Create new empty spec-re kell kattintani, majd bejön egy ablak be lehet állítani a nevet.
* Create Spec, majd az Okay, run the spec gombokra kell nyomni és létrehozhatjuk az új tesztet.
* A program fájljaiban létrejön egy cypress mappa, ahova a tesztet kell írni. A teszt eredményét pedig a behozott Chrome oldalon láthatjuk.

**End-2-end tesztelés:**

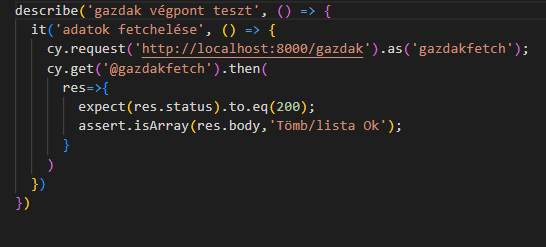
Az "end-to-end" (végponttól-végpontig) tesztelés egy olyan tesztelési módszer, amely során az alkalmazás teljes folyamatát teszteljük, az alkalmazás felhasználói felületétől (UI) a háttérlogikáig és adatbázisig. Az end-to-end tesztek szimulálják a valódi felhasználói műveleteket, például kattintásokat, adatbevitelt és navigációt, majd ellenőrzik az alkalmazás állapotát és válaszait.

**Előnyei:**

* Az E2E tesztek valódi böngészőben futnak, és szimulálják a felhasználó tényleges műveleteit (pl. kattintások, űrlapok kitöltése stb.).
* Lehetővé teszi az alkalmazás felhasználói felületének (UI) és a felhasználói interakcióknak (pl. animációk, megjelenítési változások) az ellenőrzését valós idejű környezetben.
* Valódi böngészőkben futnak, ami lehetővé teszi az alkalmazás viselkedésének ellenőrzését a valós környezetben. Ez segít a böngésző kompatibilitás és a platformfüggetlen működés ellenőrzésében.

**Frontend:** 

Ez a teszt először megvizsgálja, hogy a localhost:5173-as oldal betöltődik e. Ezt követően megnézi, hogy a menüpontok helyesen vannak e leírva és a megfelelő fetch-re irányít minket. Az alábbi példa az orvosok menüpontot ellenőrzi.

**Backend:** 

Ez a teszt a gazdak végpontot kezeli le, hogy megfelelően töltődik-e be a szükséges adat.

**Tesztelési eredmény:**

Frontenden az oldal betöltés sikeresen működött a localhost:5173-on. Ezt követően leteszteltük az összes menüpontot egyesével, hogy a megfelelő szó van-e oda írva, működik-e a kattintás és hogy a link jól funkciónál-e. Mindegyik menüpontnál sikeresen végrehajtódott a teszt hiba nélkül. A backenden minden útvonalat megnéztünk, hogy a megfelelő oldalra irányít- e minket. Ez a teszt is sikeresen lefutott.

**Továbbfejlesztési lehetőség**

A jelenlegi program a feladatának megfelel, azonban mindig van igény a fejlődésre és javításra. Fontos célunk, hogy a program további fejlesztésekkel minél jobb és személyre szabottabb legyen a leendő felhasználók részére. Fontos a felhasználók igényeinek a figyelembevétele is, ugyanis a program ezzel még fejleszthetőbbé válhat.

**Ötletek:**

* **Gyógyszerelérhetőségi adatbázis:** A fejlesztés célja az lenne, hogy az orvosok könnyen utána tudjanak nézni, hogy milyen gyógyszer elérhető náluk és hogy nem fogyott-e el, anélkül, hogy keresgélniük kellene a polcokon.
* **Integráció más szoftverekkel:** Lehetőséget szeretnénk arra, hogy a program más programmal is kompatibilis legyen.
* **Emlékeztetők, naptár:** A napi fontosabb teendők számon tartása a program naptárában, ahol emlékeztetőt is lehet majd beállítani külön konfigurációval.
* **Többnyelvűség:** Támogatni szeretnénk a többnyelvű felhasználói felületet, hogy a programot el lehessen juttatni más országok és kultúrák felé, növelve ezzel a program potenciális felhasználói bázisát.
* **Szűrők:** Összetettebb szűrők beállítása, amelynél nem csak az állat nevére tudunk rákeresni, hanem bármely más állat adatára, ezzel leszűkítve a keresést. Ez vonatkozik a többi táblára is (gazdák, orvosok, vizsgálatok).
* **Online üzenetküldés:** Üzenetküldési funkciót szeretnénk létrehozni, hogy az orvosok, asszisztensek, esetleg rendelők könnyebben tudjanak kommunikálni egymással vagy fájlokat átküldeni.
* **Bővíthetőség és testreszabhatóság**: Az alkalmazás bővíthető és testre szabható legyen, hogy az egyedi igényeknek és elvárásoknak megfeleljen**.**

**Irodalomjegyzék, forrásmegjelölés**

<https://visualstudio.microsoft.com/downloads/>

<https://nodejs.org/en>

<https://www.apachefriends.org/hu/download.html>

<https://react.dev/>

<https://tailwindcss.com/docs/installation>

<https://www.awh.hu/kb/webtarhely/mi-az-a-mysql>

<https://maas.hu/learning/it/prog/javascript/nodejs/#gsc.tab=0>

<https://digikiad.gitbook.io/digitalis-kiadvanyok/vs-code/mi-a-vs-code>

<https://maas.hu/learning/it/prog/javascript/nodejs/#gsc.tab=0>

<https://www.prooktatas.hu/hirek/front-end-webfejlesztes-react-tel-miert-jo-valasztas>

<https://www.cypress.io/>