**„TrackIT” Fejlesztői Programterv**

A **TrackIT** egy modern, skálázható webalkalmazás, amelyet a legújabb fejlesztési technológiák és eszközök felhasználásával készítettünk. Az alkalmazás backend része **Node.js**-ben és **TypeORM**-mel készült, az adatkezeléshez **MySQL**-t használtunk, míg a frontend **Angular**-al lett megvalósítva. A fejlesztés során olyan eszközöket alkalmaztunk, mint a **Visual Studio Code**, **GitHub**, **Jira**, **Figma**, és **Katalon Studio**, amelyek mind hozzájárultak a projekt hatékony fejlesztéséhez és a magas szintű felhasználói élmény biztosításához. Az alkalmazás fejlesztését egy szoros együttműködésben dolgozó csapat végezte, akik mindegyike szakértője saját területének, és az összes munkaág összehangolt működésével biztosítottuk a magas szintű minőséget és az időben történő szállítást.

**1. Technológiai Stack és Programok**

**1.1 Frontend Fejlesztés**

A frontend fejlesztése során az **Angular** keretrendszert használtuk, amely lehetővé tette a dinamikus, reszponzív felhasználói felület gyors fejlesztését és karbantartását. Az **Angular** egy rendkívül fejlett keretrendszer, amely lehetőséget biztosít a komponens-alapú fejlesztésre és a reaktív adatkezelésre, ami lehetővé tette a gyors adatfeldolgozást és az egyszerű hibakezelést.

A fő weboldal mappa szerkezete így néz ki: A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A képen szöveg, képernyőkép, menü, tervezés látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Az oldalon használt komponenseket külön-külön hoztuk létre, ezzel sokkal átláthatóbbá, könnyebben kezelhetővé és szebbé téve a programot.

* **TypeScript:** A **TypeScript** a JavaScript erősebb típusellenőrzésének biztosításával hozzájárult a kód magasabb minőségéhez, valamint elősegítette a gyors hibafelismerést, amit a dinamikus fejlesztési folyamatok során nagyra értékeltek a fejlesztők. A TypeScript lehetővé tette, hogy a kód karbantartása és skálázása egyszerűbb legyen, különösen a nagyobb komponensek és modulok kezelésénél.
* **SCSS (Sass):** Az alkalmazás stílusait **SCSS**-szel (Sass) kezeljük, mivel ez sokkal hatékonyabb CSS kezelést biztosít, és lehetővé teszi a bonyolultabb stíluslogikák implementálását, például a változók, mixin-ek és beágyazott stílusok alkalmazását. Ezáltal könnyen testreszabható és újrahasználható stílusokat hozhattunk létre.
* **RxJS:** A **RxJS** (Reactive Extensions for JavaScript) segítségével kezeljük az aszinkron adatfolyamokat, például a HTTP-kéréseket és az eseményeket. Az **RxJS** lehetővé tette a hatékony adatkezelést a rendszerben, ahol az adatfolyamok kezelése aszinkron módon történt, így biztosítva a gyors válaszidőket és a gördülékeny felhasználói élményt.
* **NgRx:** A frontend állapotkezelést az **NgRx** segítségével oldottuk meg. Ez a könyvtár egyszerűsíti a globális állapot kezelését, így az alkalmazás különböző komponensei között könnyedén oszthatunk meg adatokat. Az **NgRx** lehetőséget biztosított az állapot kezelésére, amely fontos a nagyobb, komplexebb alkalmazásokban, mint a TrackIT.

**1.2 Backend Fejlesztés**

A backend fejlesztését **Node.js** alapú szerveren végeztük, mivel gyors, aszinkron működést biztosít, ami ideálissá teszi webalkalmazásokhoz. A backend fő feladata az adatok kezelése, REST API-k biztosítása, valamint az alkalmazás logikájának megvalósítása. A **Node.js** lehetővé tette, hogy egyetlen programozási nyelvet használjunk mind a frontend, mind a backend oldalon, ami egyszerűsítette a fejlesztési folyamatokat és a csapat számára könnyebbé tette a projekt kezelését.

* **Express.js:** Az **Express.js** segítségével gyorsan és rugalmasan hoztuk létre a REST API-kat. Az **Express.js** egy minimalista, de erőteljes keretrendszer, amely lehetővé teszi a HTTP kérések hatékony kezelését és middleware-ek alkalmazását a különböző funkciók, mint a jogosultságkezelés és a hibakezelés megvalósítására.
* **JWT (JSON Web Tokens):** A felhasználói autentikáció és az autorizáció biztosítása érdekében **JWT**-t alkalmaztunk, amely biztonságos módot biztosít a felhasználók hitelesítésére, és segít megakadályozni a jogosulatlan hozzáférést.

**1.3 Adatbázis és ORM**

A **MySQL** adatbázis biztosítja az alkalmazás adatainak biztonságos tárolását, míg a **TypeORM** **ORM** (Object-Relational Mapping) biztosítja az egyszerű adatkezelést. Az adatbázisban tárolt információk gyors keresésére és manipulálására koncentráltunk, hogy a TrackIT alkalmazás hatékonyan működjön a nagyobb adatforgalom esetén is.

* **MySQL**: A MySQL adatbázis kezelő rendszer biztosítja az adatok **gyors** és **robusztus** tárolását. Az adatbázis segít az alkalmazás adatainak biztos kezelésében, amely elengedhetetlen volt az alkalmazás megbízhatósága és teljesítménye szempontjából.
* **TypeORM**: A TypeORM ORM biztosítja az objektum-relációs leképezést (Object-Relational Mapping), amely lehetővé tette az adatbázis műveletek egyszerű kezelését és az entitások, mint a felhasználók, célok és kihívások objektum-orientált módon történő kezelését.

Entitások

A TypeORM lehetőséget biztosít a komplex adatbázis struktúrák egyszerű kezelésére úgy, hogy az adatokat osztályokban (ún. entitások) tárolja. Az entitások olyan objektumok, amelyek az adatbázis táblák megfelelőit képviselik a kódban. Minden entitás egy osztály, amely a táblát reprezentálja, és a benne lévő mezők a táblázat oszlopaihoz tartoznak.

**Miért jók az entitások?**

1. **Objektum-orientált megközelítés:** Az entitások lehetővé teszik az adatbázis objektum-orientált módon történő kezelését, így a fejlesztők kényelmesen dolgozhatnak az adatbázis sorokkal, mintha azok sima objektumok lennének. Ez megkönnyíti a kód karbantartását és javítja a kód olvashatóságát.
2. **Egységes adatkezelés:** Az entitásoknak köszönhetően az adatbázis lekérdezéseit, frissítéseit és törlését JavaScript (vagy TypeScript) osztályok szintjén kezelhetjük, amely egységesíti a különböző adatkezelési folyamatokat az alkalmazásban. Ez csökkenti az adatbázis-specifikus SQL kódok használatát, így könnyebben átlátható és fenntartható lesz a projekt.
3. **Adatbázis-migrációk:** A TypeORM migrációs mechanizmusa lehetővé teszi az adatbázis sémájának változtatását és nyomon követését. Az entitások definíciója alapján a migrációk automatikusan generálhatók, így a fejlesztőkkönnyen frissíthetik az adatbázist a legújabb változásokkal, anélkül, hogy manuálisan kellene SQL parancsokat írniuk.
4. **Kapcsolatok kezelése:** Az entitások segítségével könnyedén modellezhetők az adatbázisban lévő kapcsolatok, mint például egy-több vagy sok-több kapcsolat. A TypeORM a kapcsolatok kezelését automatikusan végzi, így a fejlesztőknek nem kell manuálisan létrehozniuk és karbantartaniuk az összes SQL lekérdezést a kapcsolatokkal.
5. **Kód és adat szinkronizálása:** Mivel az entitások szoros kapcsolatban állnak az adatbázis táblákkal, minden változtatás, amit a kódban végzünk (például egy osztály új mezőt kap), automatikusan szinkronizálódik az adatbázissal. Így garantált, hogy az alkalmazásban lévő logika mindig tükrözni fogja az adatbázis aktuális állapotát.

**Példa Entity: A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.**

**1.4 Egyéb Használt Programok és Könyvtárak**

A fejlesztéshez használt egyéb eszközök segítettek a projekt gyors és hatékony fejlesztésében:

* **GitHub:** A verziókezeléshez **GitHub**-ot használunk, amely lehetővé teszi a fejlesztési folyamatok nyomon követését és a csapat munkájának összehangolását. A kódot rendszeresen push-oljuk a repository-ba, és biztosítjuk, hogy minden fejlesztési változtatás követhető és verzionált legyen.

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* A képen szöveg, szoftver, Számítógépes ikon, Weblap látható

  Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.**Jira:** A fejlesztési folyamatokat és feladatokat a **Jira** segítségével menedzseljük. A **Jira** segít az agilis fejlesztési módszertan követésében, lehetővé téve, hogy könnyen kezeljük a backlogot, a sprinteket és a feladatokat. Ez lehetővé tette, hogy a csapatunk hatékonyan priorizálja és nyomon kövesse a projekt előrehaladását.
* **Visual Studio Code (VS Code):** A fejlesztési környezetként **Visual Studio Code**-ot használtunk, mivel gyors, könnyen konfigurálható és számos hasznos bővítménnyel rendelkezik, amelyek segítik a fejlesztési folyamatokat. A **VS Code** biztosította a hatékony kódírást, hibakeresést és kódformázást.

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* **Figma:** A **Figma** a dizájn és prototípus készítéséhez használt eszközként segít a frontend fejlesztőknek és UI/UX dizájnereknek az alkalmazás vizuális megjelenésének kialakításában.

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* **Katalon Studio:** A **Katalon Studio**-t is használtuk automatizált teszteléshez. Ez az eszköz segített a rendszeres, automatikus regressziós tesztek végrehajtásában, amelyek biztosítják, hogy minden új funkció integrálása után az alkalmazás továbbra is hibamentesen működjön. A **Katalon Studio** hatékony eszközként segítette a teljes tesztelési folyamatot, beleértve az automatizált funkcionális és integrációs teszteket.

**2. Fejlesztési Módszertan**

A **TrackIT** fejlesztése során az **agilis fejlesztési módszertant** alkalmazzuk. Az agilis elvek szerint rövid fejlesztési ciklusokban dolgozunk, ahol minden sprint végén egy tesztelhető verziót készítünk. Az agilis módszertan lehetővé teszi számunkra, hogy gyorsan reagáljunk a változó követelményekre és biztosítsuk, hogy a végtermék a felhasználók igényeihez igazodjon.

* **CI/CD (Continuous Integration/Continuous Deployment):** A folyamatos integráció és telepítés érdekében minden kódváltoztatást rendszeresen integrálunk és automatikusan tesztelünk. Az automatikus tesztelés biztosítja, hogy a kód minden új változtatása előtt minőségellenőrzést végezzünk.

**3. Biztonság és Adatvédelem**

A **TrackIT** kiemelt figyelmet fordít az adatbiztonságra. Az alkalmazás biztosítja a felhasználói adatok védelmét.

* **JWT (JSON Web Tokens):** A felhasználói autentikáció és autorizáció **JWT** használatával történik, amely biztonságos módot biztosít a felhasználók hitelesítésére és az engedélyezett hozzáférés biztosítására.

**4. Összegzés**

A **TrackIT** fejlesztése során modern, skálázható és biztonságos technológiai stack-et alkalmaztunk, amely biztosítja a gyors fejlesztést, az egyszerű karbantartást és a magas szintű felhasználói élményt. Az **Angular**, **Node.js**, **MySQL**, és **TypeORM** kombinációja garantálja a rendszer stabilitását, míg a **GitHub**, **Jira**, **Visual Studio Code**, **Figma**, és **Katalon Studio** eszközök lehetővé teszik a fejlesztési folyamat hatékony menedzselését és a csapat közötti zökkenőmentes együttműködést. Az alkalmazás minden szempontból a legmagasabb iparági szabványoknak megfelelően készült, biztosítva ezzel a projekt hosszú távú sikerét és fenntarthatóságát.