Dokumentáció

Készítette: Buha Milán – IY5AM2

Webtechnológiák II. beadandó

Tartalomjegyzék

Bevezető

A projekt célja és áttekintése.
Az online receptmegosztó projekt elkészítéséhez egy Angular alapú frontendet hoztam létre, amely lehetővé teszi a felhasználók számára, hogy recepteket osszanak meg egy MongoDB adatbázissal összekötött backend rendszeren keresztül. Az alkalmazásom célja, hogy egy közösségi platformot biztosítson, ahol a felhasználók megoszthatják kedvenc receptjeiket, beleértve az étel nevét, a hozzávalókat, valamint a receptet feltöltő személy nevét.

A projekt megvalósítása során első lépésként létrehoztam az Angular alkalmazás vázát, amely a felhasználói felületet szolgáltatja. Az Angular keretrendszer kiválasztásának fő oka az volt, hogy modern, komponens-alapú architektúrát kínál, ami elősegíti a fejlesztési folyamatot, és segít egy jól strukturált, karbantartható kódbázis létrehozásában.

A felhasználói felületen egy űrlap található, amelyen keresztül a felhasználók megadhatják az étel nevét, a hozzávalókat, valamint saját nevüket mint recept feltöltőjét. Az adatok beküldése után azok egy Node.js és Express.js alapú backend szerverre kerülnek, amely kezeli a kéréseket és kommunikál a MongoDB adatbázissal. A MongoDB-t választottam az adattárolási megoldásként, mivel NoSQL adatbázisként rugalmasságot biztosít a sémamentes adattároláshoz, ami ideális a receptekhez kapcsolódó változatos adatok kezelésére.

A backend rendszer felelős az adatok validálásáért, a receptek tárolásáért és lekérdezéséért, valamint a hozzáférési jogosultságok kezeléséért. Az adatbázisban tárolt recepteket a felhasználói felületen keresztül lehet megtekinteni, ahol a felhasználók listáját látják az összes rendelkezésre álló recepttel, beleértve a feltöltő nevét és az étel hozzávalóit. A felhasználói élmény javítása érdekében implementáltam egy törlési funkciót is, amely lehetővé teszi a felhasználók számára, hogy saját receptjeiket törölhessék az adatbázisból. Ezenkívül biztonsági intézkedéseket vezettem be a jogosulatlan hozzáférés és a receptek jogosulatlan törlése ellen.

A projekt kihívásai közé tartozott a felhasználói jogosultságok megfelelő kezelése, a frontend és backend közötti adatátvitel biztonságos és hatékony megvalósítása, valamint a felhasználói felület intuitív és felhasználóbarát kialakítása.

Weboldal felépítése és navigációs szerkezet

A weboldal struktúrájának leírása
Az Angular keretrendszer választását számos tényező motiválta a receptmegosztó webalkalmazás fejlesztése során. Az Angular egyike azoknak a modern frontend technológiáknak, amelyeket kifejezetten webalkalmazások fejlesztésére terveztek. Az Angularban találtam meg azt a rugalmasságot, teljesítményt és fejlesztői eszközkészletet, amelyre egy dinamikus és interaktív webes projekt megvalósításához szükségem volt.

Miért Az Angular?

Komponensalapú Szerkezet: Az Angular komponensalapú architektúrája különösen vonzó volt a projekt számára, mivel lehetővé teszi az újrafelhasználható UI elemek kialakítását. A komponensek segítségével az alkalmazás logikáját és felületét logikailag elszeparált egységekre bonthattam, ami elősegítette a kód karbantarthatóságát és tesztelhetőségét. A "FoodComponent" például egy olyan komponens, amely kifejezetten a receptek megjelenítéséért és kezeléséért felelős, biztosítva ezzel az alkalmazás strukturált és rendezett felépítését.

Two-way Data Binding: Az Angular kétirányú adatkötése (two-way data binding) rendkívül hasznos volt a felhasználói interakciók kezelésében. Ennek köszönhetően a felhasználói felület és az alkalmazás állapota közötti szinkronizáció zökkenőmentes és intuitív módon történik, ami csökkenti a szükséges boilerplate kódot és megkönnyíti a fejlesztést.

TypeScript: Az Angular TypeScript alapú, ami statikus típusellenőrzést biztosít, így a fejlesztés során korai szakaszban felfedezhetőek a típushibák. A TypeScript használata növeli az alkalmazás stabilitását és karbantarthatóságát, valamint javítja a fejlesztői élményt a jobb kód autokompleció és intelligens hibakeresés által.

Kiterjedt Eszközkészlet: Az Angular számos beépített funkciót és szolgáltatást kínál, mint például az HttpClient a backenddel való kommunikációra, formák kezelésére szolgáló modulok és routing megoldások. Ezek az eszközök lehetővé tették számomra, hogy koncentráljak a receptmegosztó alkalmazás üzleti logikájának

megvalósítására, anélkül, hogy minden alapfunkciót nulláról kellene felépítenem.

Skálázhatóság és Jövőbiztosság: Az Angular moduláris felépítése és a Google általi aktív támogatása biztosítja, hogy az alkalmazás könnyen bővíthető és frissíthető legyen a jövőben. Ez fontos szempont volt a projekt hosszú távú sikeressége szempontjából.

A projekt központi eleme a "food" komponens, amely az összes recept kezeléséért felelős. Itt kerülnek megjelenítésre a receptek adatai, itt történik a receptek hozzáadása és törlése. Az adatok beküldése egy egyszerű űrlapon keresztül történik, amelyet a komponens kezel, így a felhasználói interakciók közvetlenül itt zajlanak. A komponens felépítése és a benne rejlő logika tükrözi az Angular által nyújtotta előnyöket, például a komponensalapú szerkezetet, a kétirányú adatkötést és az adatok kezelésének egyszerűsített módjait.

A food komponens

A komponens funkcióinak és elemeinek ismertetése
Ez a komponens az Angular keretrendszer előnyeit kihasználva biztosítja az adatok sima áramlását a felhasználói felület és a szerver között, valamint az alkalmazás állapotának hatékony kezelését.

Komponens Felépítése és Funkciói
Űrlap a Receptek Feltöltéséhez: A komponens egy űrlapot tartalmaz, amelyen keresztül a felhasználók megadhatják a recept nevét, a hozzávalókat leíró szöveget, és saját nevüket mint feltöltőt. Az űrlap

adatait a submitFood() metódus dolgozza fel, amely egy POST kérést küld a szervernek, hogy tárolja az új receptet az adatbázisban.

Receptek Listázása: Az alkalmazás dinamikusan listázza a meglévő recepteket, amelyek az adatbázisból kerülnek lekérdezésre a loadFoods() metódus segítségével. A metódus egy GET kérést indít a szerver felé, ami visszaküldi az összes elérhető receptet, és a komponens megjeleníti azokat egy listában.

Receptek Törlése: Minden listázott recept mellett egy "X" gomb található, amely lehetővé teszi a felhasználók számára, hogy töröljék a saját receptjeiket. A törlés funkció a deleteFood() metóduson keresztül valósul meg, ami DELETE kérést küld a szervernek az adott recept azonosítójával. A jogosultság ellenőrzése miatt a törlési kérés tartalmazza a feltöltő nevét is, biztosítva, hogy csak a recept létrehozója törölhesse azt.

Technikai Megvalósítás
Two-way Data Binding: Az Angular kétirányú adatkötése (two-way data binding) a (ngModel) direktíván keresztül biztosítja, hogy az űrlapmezők és a komponens állapota között automatikus szinkronizáció történjen. Ez segíti a felhasználói interakciók egyszerű kezelését és az adatok valós idejű frissítését a felületen.

HTTP Kommunikáció: A HttpClient modul segítségével a komponens kommunikál a backenddel. Ez a modul biztosítja az HTTP kérések egyszerű kivitelezését és kezelését Angularban, lehetővé téve az adatok zökkenőmentes küldését és fogadását a szerverrel.

FormsModule és CommonModule: A FormsModule szükséges az űrlapkezeléshez, míg a CommonModule az Angularban gyakran használt direktívák, például az *ngFor a listák dinamikus generálásához és az *ngIf az állapotfüggő megjelenítés vezérléséhez.

Food service

Röviden a service-ről

A FoodService egy központi szerepet tölt be, mivel ez felelős a backenddel történő kommunikációért. Ez a szolgáltatás a HttpClient modult használja az adatok szerverről való lekérése és küldése céljából, így közvetlenül kapcsolódik a RESTful API-hoz, amely az adatokat kezeli.

A Szolgáltatás Szerepe és Működése

Injektálhatóság és Könnyű Hozzáférés: Az @Injectable() dekorátorral ellátott FoodService szolgáltatás az Angular Dependency Injection (DI) rendszerének köszönhetően könnyen injektálható bármely komponensbe vagy más szolgáltatásba. Ez biztosítja, hogy az alkalmazás különböző részei egyszerűen hozzáférjenek a szükséges adatokhoz és funkciókhoz, növelve ezzel a kód újrafelhasználhatóságát és a fejlesztési folyamat hatékonyságát.

Adatkommunikáció a Backenddel: A FoodService alapvető funkciója, hogy megkönnyítse a receptekkel kapcsolatos adatok lekérését és manipulálását a backend rendszeren keresztül. A getFoods() metódus például egy HTTP GET kérést indít a szerver felé, amely visszatér az

összes elérhető recept listájával. Ez lehetővé teszi a frontend számára, hogy dinamikusan jelenítse meg az adatokat az alkalmazásban.

Technikai Megvalósítás

HttpClient Modul: Az Angular HttpClient modulja a modern webalkalmazások egyik alapköve, amely egyszerűsíti az aszinkron HTTP kérések kezelését. Az HttpClient segítségével a FoodService képes JSON formátumban kommunikálni a backenddel, így a frontend és a szerver közötti adatcsere gyors és hatékony.

Observable és RxJS: A szolgáltatás az Observable típust használja az aszinkron adatáramlás kezelésére. Az Observableek segítségével a FoodService reaktív módon tudja kezelni az adatokat, ami lehetővé teszi, hogy az alkalmazás frissítse a felületet az adatok változásával szinkronban. Az RxJS könyvtár funkcióit felhasználva a szolgáltatás további operátorokat és feldolgozási lépéseket is alkalmazhat az adatok manipulálására, mielőtt azok elérnék a komponenseket.

Backend

Mi található a backenden

Ez egy Node.js és Express alapú REST API, amely a MongoDB-t használja adattárolásra. Ez a szervertömb teszi lehetővé, hogy a receptmegosztó alkalmazásunk adatokat küldjön és fogadjon a frontend és az adatbázis között. A MongoDB Atlas szolgáltatáson futó clusterhez való kapcsolódás biztosítja az adatok skálázható és biztonságos tárolását.

Alapvető Funkciók és Implementáció

Express Szerver: Az Express egy minimális és rugalmas Node.js webalkalmazás keretrendszer, amely könnyűszerrel kezeli a kéréseket és válaszokat. A szerverünk az Express-t használja az útvonalak kezelésére, amely lehetővé teszi a különböző HTTP kérések (GET, POST, DELETE stb.) fogadását és kezelését.

Mongoose ODM: A Mongoose egy objektum-dokumentum leképező (ODM) könyvtár a MongoDB és Node.js számára. A Food modellünk definíciójával és a kapcsolódó sémával (schema) a Mongoose segít a MongoDB adatbázissal való interakcióban, például adatok mentésében, lekérdezésében és törlésében.

CORS és Body Parser Middleware: A CORS (Cross-Origin Resource Sharing) middleware biztosítja, hogy a frontend alkalmazásunk más domainről is hozzáférjen a backend által szolgáltatott erőforrásokhoz. A Body Parser middleware pedig az érkező kérések testét dolgozza fel, lehetővé téve, hogy a szerver JSON formátumban fogadja és értelmezze az adatokat.

Az API Endpontjai és Funkcióik

POST /foods: Új recept hozzáadása az adatbázishoz. A kérés törzsében szereplő adatok alapján létrehoz egy új Food dokumentumot, majd menti azt az adatbázisba.

GET /foods: Az összes recept lekérése az adatbázisból. Visszaadja a megtalált dokumentumok JSON formátumú listáját.

DELETE /foods/:foodId: Egy meghatározott azonosítójú recept törlése az adatbázisból. Ellenőrzi a jogosultságot a feltöltő nevével, és csak akkor engedélyezi a törlést, ha a kérésben szereplő feltöltő neve megegyezik a dokumentum feltöltőjével.

Adatbázis

MongoDB Cluster

A MongoDB Atlas egy teljesen kezelt, felhőalapú adatbázis szolgáltatás, amely lehetővé teszi a MongoDB adatbázisok egyszerű létrehozását, karbantartását és skálázását a felhőben. A projektünkben a MongoDB Atlas szolgáltatást használjuk, amely számos előnyt kínál, beleértve a magas rendelkezésre állást, a biztonságot, a rugalmasságot és a könnyű skálázhatóságot.

Fő Jellemzők és Előnyök

Magas Rendelkezésre Állás: A MongoDB Atlas automatikusan kezeli a replikációkat és az adatbázis biztonsági mentéseit, biztosítva ezzel az adatok magas rendelkezésre állását és tartósságát. Ez kritikus fontosságú a vállalkozások és az alkalmazások számára, ahol az adatvesztés elfogadhatatlan.

Automatikus Skálázás: Az Atlas lehetővé teszi a tároló és a számítási erőforrások automatikus vagy kézi skálázását. Ez azt jelenti, hogy az adatbázis könnyen alkalmazkodik az alkalmazás növekvő igényeihez, optimalizálva ezzel a költségeket és a teljesítményt.

Globális Adatközpontok: A MongoDB Atlas globális szinten több adatközpontot kínál, lehetővé téve a felhasználók számára, hogy az adatbázisaikat fizikailag közel helyezzék az alkalmazásuk felhasználóihoz. Ez jelentősen csökkentheti a késleltetést és javíthatja a felhasználói élményt.

Biztonság: Az Atlas számos biztonsági funkciót kínál, mint például a VPC peering, az adat titkosítás pihenő állapotban és tranzit közben, valamint a finom szemcsézettségű hozzáférés-szabályozás. Ezek a funkciók segítenek megvédeni az adatokat a nem kívánt hozzáféréstől és a biztonsági fenyegetésektől.

Használata a projektben

A MongoDB Atlas szolgáltatást használom azért, mert egy felhőalapú megoldás, amely könnyen integrálható a modern webes alkalmazásokka. A cluster konfigurációjának köszönhetően képes vagyok hatékonyan tárolni a receptekkel kapcsolatos adatokat, beleértve a neveket, leírásokat és egyéb információkat. Az Atlas kezelőfelülete könnyen használható, ami egyszerűvé teszi az adatbázisok kezelését, monitorozását és karbantartását.

Az Atlas felhőalapú természete azt is jelenti, hogy nem kell saját szerverinfrastruktúrát fenntartani és karbantartani, ami időt és erőforrásokat takarít meg számomra.