ZÁRÓDOLGOZAT

Készítették:

Zsóka Marcell Tamás – Dobos Levente

Konzulens:

Kerényi Róbert Nándor

Tartalomjegyzék

Technológiák

Adatbázis

Az adatbázis-kezelés alapvető jelentőségű minden webalkalmazás, így a webshopok működésében is. Itt az adatok tárolása, lekérdezése, frissítése és törlése történik. Az adatbázisok lehetnek relációs vagy NoSQL típusúak, attól függően, hogy milyen adattípusokkal és mennyiséggel dolgozik a rendszer. A hatékony adatbázis-kezelés kulcsfontosságú a webshop gyors és megbízható működéséhez.

Backend

A backend a webshop "motorja", itt történik az adatfeldolgozás, üzleti logika megvalósítása és az adatbázis-kezelés. A következő technológiák szerepelnek:

Fejlesztői Környezet (Visual Studio Code)

A Visual Studio Code egy széles körben használt, könnyen kezelhető fejlesztői környezet, amely támogatja a C# nyelvet és számos más technológiát. Könnyen bővíthető plugin-rendszerével optimalizálható a fejlesztési folyamat.

WEB API

A Web API lehetővé teszi, hogy RESTful szolgáltatásokat hozzunk létre, amelyeken keresztül a frontend könnyedén kommunikálhat a backenddel. Ez a technológia az adatok HTTP protokollon keresztüli szabványosított kezelését teszi lehetővé.

ENTITY FRAMEWORK

Az Entity Framework egy ORM (Object-Relational Mapping) eszköz, amely segít a .NET alkalmazások adatbázis-kezelésében. Leegyszerűsíti az adatmodellek és az adatbázissémák közötti kapcsolat kezelését.

CORS (Cross-Origin Resource Sharing)

A CORS egy biztonsági mechanizmus, amely lehetővé teszi a különböző eredetű forrásokból származó erőforrások megosztását. Ez elengedhetetlen a modern, több eredetű webalkalmazások számára.

C#

A C# egy objektumorientált programozási nyelv, amely a .NET keretrendszerben gyakran használt a backend fejlesztéséhez. Robusztus szintaxisa és erőteljes típusrendszere hatékony eszközt biztosít a fejlesztők számára.

ORM (Object-Relational Mapping)

Az ORM technológia a táblázatos adatbázis-struktúrákat objektumorientált modellé alakítja, megkönnyítve ezzel a fejlesztői munkát és a kód karbantarthatóságát.

CRUD és Biztonság

A CRUD (Create, Read, Update, Delete) az adatbázis-műveletek alapvető műveleteit írja le. Ezen funkciók biztonságos implementációja kritikus, mivel védi az adatok integritását és a felhasználói információkat.

Frontend

A frontend az, amit a felhasználók látnak és használnak; ezért különösen fontos a gyors, responsive és vonzó felhasználói felület.

React

A React egy JavaScript könyvtár, amely a felhasználói interfészek építésére szolgál. Komponens-alapú megközelítése lehetővé teszi az újrafelhasználható UI elemek készítését.

Node.js

A Node.js egy JavaScript futási környezet, amely lehetővé teszi a JavaScript használatát a szerver oldalon. Ez a technológia különösen hasznos a webshopoknál, mivel egységesíti a frontend és backend fejlesztési nyelveit, így a fejlesztőknek nem kell külön nyelveket tanulniuk a különböző alkalmazásrétegekhez.

JWT Token

A JWT (JSON Web Token) egy kompakt, URL-biztonságos módszer a felhasználói állapot és a jogosultságok megbízható átvitelére a kliens és a szerver között. Ez lehetővé teszi a szolgáltatások biztonságos és hatékony hozzáférés-kezelését, nélkülözhetetlen eleme a modern webes autentikációi rendszereknek.

WPF (Windows Presentation Foundation)

A WPF a Microsoft platformon belül egy erős grafikus felhasználói interfész keretrendszer, amelyet különösen asztali alkalmazások fejlesztésére használnak. A webshop kontextusában a WPF alkalmazható a karbantartói és adminisztrációs eszközök fejlesztésére, amelyek segítenek a termékek és tranzakciók kezelésében, valamint a felhasználói interakciók elemzésében.

Ezek a technológiák együttesen alkotják a webshop technológiai alapjait. Minden egyes elem kulcsfontosságú szerepet tölt be a rendszer egészének stabilitásában, biztonságában és felhasználói élményében. Az adatbázistól kezdve a szerveroldali logikán át a kliensoldali prezentációig minden komponens összehangolt működése biztosítja, hogy a végfelhasználók gyors, biztonságos és kényelmes vásárlási élményben részesüljenek.

Miért Pont egy “Webshop”?

Amikor szembesültünk a feladattal, hogy egy webes alkalmazást szükséges fejlesztenünk, számos alternatíva közül választhattunk. A döntés, hogy egy webshopot hozzunk létre, több lényeges szempont alapján született meg. Egy webshop kifejlesztése nem pusztán hogy lehetőséget adott számunkra a React technológia mélyebb megismerésére, hanem számos kihívást is tartogatott, amik segítettek fejleszteni a programozási, valamint tervezési készségeinket.

Legfőbb oka, amiért a webshopot választottuk, az a felhasználói interaktivitás, valamint élmény priorált fontossága. Egy e-kereskedelmi platformon a felhasználói interakciók sokrétűek, valamint összetettek, legyen szó termékek böngészéséről, kosárba helyezésről, esetleg a fizetési processzusról. A React technológia dinamikus, válaszkészen reagáló felhasználói felületeket tesz lehetővé, amik javítják a vásárlói élményt, ezen kívül növelik a weboldal hatékonyságát. Amikor egy webshopot fejlesztettem, számos kihívással kellett szembenéznem, amelyek a projekt során felmerültek. Az egyik legnagyobb kihívás az volt, hogy a felhasználói élményt magas színvonalon tartsam, miközben a weboldal teljesítményét is optimalizálni kellett. Az elsődleges feladat az volt, hogy egy intuitív és felhasználóbarát design-t hozzak létre, amely egyszerű navigációt és könnyű vásárlási folyamatot biztosít a látogatók számára. Ehhez szükség volt a felhasználói kutatásra és a prototípusok tesztelésére, hogy biztosítsam az optimális felhasználói élményt. A másik nagy kihívás az volt, hogy a webshopot biztonságosan működtessük. Az online vásárlások során a biztonság kiemelten fontos tényező, így gondoskodni kellett a SSL tanúsítványról, a fizetési tranzakciók titkosításáról és az adatvédelmi szabályozások betartásáról. Az üzemeltetés és karbantartás terén is találkoztam kihívásokkal. A webshop folyamatosan fejlődik, új funkciók és termékek kerülnek bevezetésre, amelyeknek megfelelően frissíteni kellett a rendszert. Emellett fontos volt a weboldal teljesítményének monitorozása és optimalizálása, hogy gyors és megbízható maradjon még a növekvő forgalom mellett is.

Összességében a webshop fejlesztése során számos kihívással találkoztam, de mindegyiket sikerült sikeresen kezelni a megfelelő tervezéssel, stratégiával és csapatmunkával. A projekt végén egy jól működő, felhasználóbarát és biztonságos webshopot sikerült létrehoznunk, amely megfelelt az ügyfelek elvárásainak és igényeinek.

Front - end felépítése
(REACT)

A React, mint a modern webfejlesztés egyik legelterjedtebb, ezen kívül legnépszerűbb JavaScript könyvtára, számos előnnyel rendelkezik, amik okán a fejlesztők széles köre részesíti előnyben ezt a keretrendszert. Az alábbiakban összefoglaljuk a React főbb előnyeit, amik okán sokan preferálják a weboldalak, valamint alkalmazások fejlesztéséhez:

Komponens-alapú architektúra

A komponens-alapú architektúra(szerkezet) szintén kulcsfontosságú szerepet játszott a döntésünkben. Egy webshop számos funkciót, ezen kívül oldalt hordozhat magában, mint példának okáért a termékek listázása, a felhasználói fiókok kezelése, továbbá a rendelési mechanizmus. A React modularitása lehetővé teszi számunkra, hogy az alkalmazást jól strukturált, újrafelhasználható komponensekre bontsuk, ami nagymértékben leegyszerűsíti a fejlesztést, továbbá karbantartást.

Virtuális DOM

A React használatának egyik kulcsfontosságú előnye a virtuális DOM (Document Object Model) megközelítése, mely nagyban javítja az alkalmazások teljesítményét, továbbá reaktivitását. A React egy virtuális DOM-ot alkalmaz a valós DOM helyett, ami azt jelenti, hogy az alkalmazás a böngészővel egy virtuális rétegen keresztül kommunikál. Ez a megközelítés lehetővé teszi, hogy csupán azokat a DOM elemeket frissítse, amik valóban megváltoztak, ezzel minimalizálva a valós DOM-ra gyakorolt terhelést. Ez különösen előnyös a webshopok esetén, ahol a hatalmas mennyiségű adatfrissítés, továbbá a gyakran előforduló felhasználói interakciók kezelése nélkülözhetetlen. A virtuális DOM effektív frissítései, ezen kívül a rapid válaszidők létfontosságúak egy interaktív weboldal, mint egy webshop zökkenőmentes működéséhez.

Egyszerű, valamint deklaratív kódolás

A React lehetővé teszi a fejlesztők számára, hogy a felhasználói felületet deklaratív módon definiálják. Ennek eredményeként az alkalmazás kódja könnyen érthető, továbbá karbantartható marad. A JSX (JavaScript XML) segítségével a fejlesztők HTML-szerű szintaxisban tudnak JSX elemeket definiálni, amelyek magukban hordozzák a JavaScript kódot. Ez a megközelítés csökkenti a hibák lehetőségét, továbbá növeli a fejlesztési processzus hatékonyságát, mivel a kód strukturáltabb, ezen kívül könnyebben áttekinthető. Továbbá, a deklaratív jelleg segíti az újabb fejlesztőket is a projektbe való gyorsabb becsatlakozásban, hiszen intuitívabb az összetevők funkcióinak megértése.

Elosztott kisközösség, továbbá ökoszisztéma

A React rendkívül széles körben elfogadott a fejlesztői közösségben, így rengeteg támogató dokumentáció, könyvtár, ezen kívül kész megoldás érhető el hozzá. Ez a gazdag ökoszisztéma jelentős mértékben megkönnyíti az alkalmazások fejlesztését, valamint a problémák megoldását. A közösségi segítség lehetővé teszi, hogy a fejlesztők rapid módon megtalálják a szükséges erőforrásokat, ezen kívül megoldásokat a problémákra. A permanensen növekvő pluginok, valamint kiegészítő könyvtárak gazdag választéka tovább bővíti a React alkalmazhatóságát. Ez a sokszínűség segít a fejlesztőknek abban, hogy alkalmazásaikat a lehető legjobban kihasználják, optimalizálják, ezen kívül egyedi igényeik szerint szabják személyre.

Keresztplatformos kompatibilitás

A React nem korlátozódik csak webalkalmazásokra; az Electron, React Native, ezen kívül más keretrendszerek segítségével lehetőség van asztali alkalmazások, ezen kívül mobilalkalmazások fejlesztésére is, React alapokon. Ez a keresztplatformos kompatibilitás lehetővé teszi a fejlesztők számára, hogy egyedüli kódbázist használjanak több platformon, így csökkentve az ismételt munkát, ezen kívül növelve a produktivitást. A React Native példának okáért lehetővé teszi, hogy natív mobilalkalmazásokat fejlesszenek JavaScript támogatásával, ami jelentős költségmegtakarítást jelent a szolgáltatók részére.

Teljesítmény, továbbá skálázhatóság

A React rendkívül hatékony és skálázható, ami lehetővé teszi nagyobb projekt méretek kezelését is. A komponens-alapú megközelítés, továbbá a virtuális DOM segít fenntartani az alkalmazás hatékonyságát még hatalmas terhelés esetekor is. Az magas intelligenciájú frissítési logika minimalizálja a szükségtelen műveletek számát, így az alkalmazás gyorsabb, valamint válaszkészebb marad abban az esetben is, ha összetett felhasználói interakciókat, továbbá adatfrissítéseket szükséges kezelnie. Ez a magas szintű teljesítőképesség, valamint skálázhatóság

teszi a Reactot ideális választássá nagyméretű látogatottságú weboldalak, ezen kívül hatalmas adatigényű alkalmazások részére.

Támogatottság, valamint frissítések

A React aktív fejlesztés alatt áll, valamint folyamatosan frissítik, így a fejlesztők biztosak lehetnek benne, hogy az alkalmazásuk mindig a legújabb technológiákat használja. Az újabb funkciók, ezen kívül javítások rendszeres időközönként történő bevezetése biztosítja, hogy a React alkalmazások optimalizáltak, továbbá naprakészek maradnak a modern webfejlesztési trendekkel. Ez a permanens segítség, valamint frissítés növeli a React alkalmazások biztonságát, valamint megbízhatóságát, ami kulcsfontosságú a vállalati szintű projekteknél.

Flexibilis konfiguráció

A React lehetőséget ad a fejlesztőknek arra, hogy a projektjeiket személyre szabják az igényeiknek jó módon. Számos kiegészítő, továbbá konfigurációs beállítás érhető el, amelyek segítségével az alkalmazás még hatékonyabban és rugalmasabban fejleszthető. Ez a flexibilitás lehetővé teszi, hogy a fejlesztők rapid módon reagáljanak a változó üzleti követelményekre, valamint integrálják a legújabb innovációkat. A széleskörű testreszabhatóság elősegíti az alkalmazások távlati rentábilisságát, ezen kívül skálázhatóságát, amik alapvetőek a technológia rapid módon változó világában.

A webshop fejlesztése egy kiváló precedens a React használatára számos dimenzió tekintetében. Íme egy-két ok, amiért ezt a projektet választottuk:

Felhasználói interaktivitás: Egy webshop esetekor magas prioritású a felhasználói élmény, ezen kívül az interaktivitás. A React lehetővé teszi dinamikus, rapid módon válaszadó felhasználói felületek kialakítását, amik jobb vásárlói élményt nyújtanak.
Komponens-alapú struktúra: A webshopok sok esetben bírnak sok féle oldallal, funkcióval, ezen kívül komponenssel. A React komponens-alapú szerkezete lehetővé teszi az alkalmazás könnyű bontását, valamint újrafelhasználhatóságát, ami különösen előnyös egy ilyen komplex program esetekor.

Virtuális DOM: A webshopok sok esetben nagyméretű mennyiségű adatot, valamint gyakran előforduló felhasználói interakciót igényelnek. A React virtuális DOM-ja lehetővé teszi az effektív frissítéseket, valamint a rapid válaszidőt, ami kritikus fontosságú egy webshop esetén.
Keresőmotor optimalizálás: A React lehetővé teszi a szerveroldali renderelést (SSR), mely lehetővé teszi a webshop tartalmának rapid betöltését a keresőmotorok, továbbá a felhasználók részére. Ez kulcsfontosságú lehet a webshop SEO (keresőmotor optimalizálás) szempontjából.
Flexibilis fejlesztés: A React egy flexibilis, továbbá testre szabható keretrendszer, mely lehetővé teszi a fejlesztők részére, hogy alkalmazásukat a kívánt módon formázzák, ezen kívül skálázzák. Ez különösen releváns egy webshop esetén, ahol sok esetben változó követelményekkel szükséges szembenézni.
Támogatott társadalmi alcsoport, továbbá ökoszisztéma: A React rendelkezik egy nagyméretű, továbbá aktív közösséggel, továbbá számos kiegészítővel, valamint harmadik féltől származó könyvtárral, amik megkönnyítik a webshop fejlesztését, valamint bővítését.

Mindezek a faktorok együttesen teszik a Reactot nagyszerű választássá egy webshop fejlesztéséhez, valamint mi is emiatt választottuk ezt a projektet példaként.

A Back - end Felépítése

Backend

A backend,vagy szerveroldali bővítés az alkalmazásoknak azt a részét jelenti, mely a felhasználói interakciókat feldolgozza, az adatokat kezeli, valamint tárolja és biztosítja a frontend alkalmazások részére a szükséges adatokat, továbbá funkciókat. Egy kétfős diákcsapat esetekor mint a miénk, a backend fejlesztése magában foglalja több kulcsfontosságú technológia, ezen kívül programozási viszonyrendszer használatát, amik lehetővé teszik az effektív, valamint biztonságos adatkezelést, továbbá kommunikációt. A WEB API-k használata kritikus szerepet játszik a mikroszolgáltatások architektúrájában, ahol különálló funkciók felelnek meg különféle API végpontoknak, ezáltal modularizálva az alkalmazás fejlesztését. Az Entity Framework alkalmazásával csapatunk képes az adatmodellünk egyszerű skálázására, ezen kívül az adatbázisséma változásainak rapid alkalmazkodására, ami különösen releváns rapid módon változó program követelmények esetekor. A CORS beállítások finomhangolása lehetővé teszi, hogy biztonságosan engedélyezzék a különféle eredetű kéréseket, miközben megőrzik az alkalmazás védelmét a nem kívánt hozzáférés ellen. A C# programozási nyelv használata egy további előnyt jelent a típusbiztonság, valamint az objektum-orientált programozási paradigmák okán, amik segítik a kód karbantarthatóságát, ezen kívül a fejlesztési processzus hatékonyságának növelését. Végül az ORM, valamint CRUD operációk kombinációja lehetővé teszi a diákcsapat részére, hogy flexibilisen kezeljék az adatokat, miközben fenntartják a szigorú biztonsági protokollokat.

Fejlesztői környezet és Eszközök

WEB API

A WEB API egy olyan interfész, mely lehetővé teszi az alkalmazások részére, hogy különféle programozási nyelveken keresztül kommunikáljanak a backend szolgáltatásokkal. Ezáltal könnyen integrálhattuk a frontend, továbbá backend részeket, lehetővé téve az adatok zökkenőmentes áramlását az alkalmazás

különféle részei között. Használatuk rendkívül effektív metódus az alkalmazások moduláris felépítésére, mivel lehetővé teszik a fejlesztők részére, hogy független módon fejlesszék a különféle alkalmazásrétegeket. Az API-kon keresztüli adatkommunikáció standardizálása csökkenti a fejlesztési időt, továbbá növeli a kód újrafelhasználhatóságát. Az API-k támogatásával kísérletezhettünk különféle frontend technológiákkal, anélkül, hogy a backend logikáját újra kellene írnunk, ami rugalmasságot biztosított a program különféle szakaszaiban. A jól megtervezett WEB API biztosítja az adatok integritását, valamint biztonságát, mivel alternatíva van az adatátviteli protokollok, mint az HTTPS, valamint az adatok titkosításának használatára. Emellett, a WEB API-k skálázhatósága lehetővé teszi, hogy egyszerűen kezeljük a felhasználói bázis növekedését, esetleg a rendszer terhelésének változásait, mivel a szervererőforrásokat dinamikusan lehet hozzárendelni az igényekhez.

ENTITY FRAMEWORK

Az Entity Framework (EF) egy objektum-relációs leképező (ORM) keretrendszer, melyet a .NET platform támogat. Támogatásával objektum-orientált módon kezelhettük az adatbázis-műveleteket, anélkül, hogy direkt módon SQL parancsokat kellene írnunk. Ez nagymértékben csökkenti a fejlesztési időt, továbbá növeli a kód olvashatóságát, ezen kívül karbantarthatóságát. Az Entity Framework használata lehetővé teszi hogy magasabb szintű abstrakcióval dolgozzunk, ami csökkenti a hibák esélyét az adatbázis-műveletek alkalmával. Az EF támogatja a LINQ (Language Integrated Query) lekérdezéseket, amik támogatásával a C# nyelv szintaxisát alkalmazva írhatunk lekérdezéseket, ezáltal a kód egyszerűbb, valamint könnyebben átlátható lesz. Az EF automatikusan kezeli a kapcsolatokat az adatbázis táblák, valamint az alkalmazásban használt objektumok között, így könnyedén implementálhatjuk az összetett adatmodelleket is. Az Entity Framework "Code First", továbbá "Database First" módszertanokat is kínál, amik lehetővé teszik, hogy az adatmodellünket a kódból generáljuk, esetleg a meglévő adatbázisunk alapján hozzuk létre a szükséges kódstruktúrát. Végül, az EF integrálása a .NET keretrendszerrel, továbbá annak eszközeivel, mint a Visual Studio, tovább egyszerűsíti a fejlesztési processzust, mivel ezek az instrumentumok számos automatizált funkciót, valamint vizuális segédletet kínálnak az ORM kezeléséhez.

CORS

A Cross-Origin Resource Sharing (CORS) egy biztonsági folyamat, mely lehetővé teszi a webalkalmazások részére, hogy korlátozott erőforrásokat használjanak fel egy másik domainről. Ez nélkülözhetetlen azokban a projektben, ahol a frontend, ezen kívül a backend különféle szervereken fut, megakadályozva az esetleg előforduló biztonsági rések kihasználását. A CORS beállításokat megfelelőképpen konfigurálva a meghatározhatjuk, hogy mely források, mint példának okáért képek,

stíluslapok, esetleg szkriptek, tölthetők be más domainekről, ezáltal szabályozva az adatok, továbbá források megosztását különféle webhelyek között. Ez a folyamat segít abban, hogy biztonságosan implementálhassuk a szolgáltatások közti adatáramlást, miközben megóvjuk a felhasználókat a kártékony webes támadásoktól, mint amilyen az adathalászat. A CORS politikák különösen fontosak a mikroszolgáltatások architektúrájában, ahol az egyes szolgáltatások különféle domainekről származó kéréseket is kezelhetnek. Továbbá, a CORS támogatásával a fejlesztők finomhangolhatják az engedélyezett HTTP-fejléceket, módszereket, ezen kívül hitelesítési beállításokat, amik irányítják, hogy melyik kliens milyen jellegű kéréseket indíthat a szerver felé. Ezen beállítások jó alkalmazása növeli az alkalmazások interoperabilitását, miközben fenntartja a felhasználói adatok biztonságát, ezen kívül a rendszer integritását.

C#

A .NET keretrendszerrel együtt a C# nyelv lehetővé teszi a fejlesztők számára, hogy gyorsan és hatékonyan készítsenek biztonságos, méretezhető és jól karbantartható alkalmazásokat. Ez a nyelv támogatja a kiterjesztett hibakezelést is, ami döntő fontosságú a magas rendelkezésre állású rendszerekben, ahol a jó kivételkezelés biztosítja az alkalmazás állandó működését. A C# nyelven a fejlesztők olyan fejlett funkciókat használhatnak, mint a delegátusok és események, amelyek támogatják az eseményvezérelt programozás elegáns megvalósítását, így növelve a kód moduláris felépítését és olvashatóságát. A delegátusok típusbiztos funkciómutatók, amelyek lehetővé teszik a funkciók átadását más funkciókhoz vagy eseményekhez, így dinamizálják a projekt működését. Az események használatával az osztályok jelezhetik a belső állapotváltozásokat, vagy olyan fontos események bekövetkezését, amelyekre más osztályok eseménykezelési módszerekkel reagálhatnak. Ezek a folyamatok kulcsfontosságúak az eseményvezérelt programozásban, lehetővé téve a kód modularizálását és a rendszerkomponensek közötti hatékony kommunikációt. Ezen kívül a C# generikus támogatása lehetővé teszi a típusbiztos kód írását, ami csökkenti a típushoz kapcsolódó futásidejű hibák lehetőségét, így növeli az alkalmazás stabilitását. Végül a C# integrált támogatása a LINQ-hoz (Language Integrated Query), amely rendkívül hatékonnyá és kifejezővé teszi a gyűjteményekkel való munkát, tovább erősíti ennek a programozási nyelvnek a különböző szoftverfejlesztési forgatókönyvekben való alkalmazhatóságát.

ORM és CRUD

Az ORM technológia segít az adatbázis-műveletek objektumorientált kezelésében, amely áthidalja az objektum-orientált programozási nyelvek és a relációs adatbázisok közötti szemantikai különbségeket. Ennek eredményeként a fejlesztők könnyen kezelhetik az adatbázisban tárolt információkat, mintha csak egyszerű

programozási objektumokkal dolgoznának, ezáltal csökkentve az összetett SQL lekérdezések szükségességét. A CRUD műveletek, mint például az adatok létrehozása, olvasása, frissítése és törlése, az adatkezelés gerincét alkotják, és minden típusú adatvezérelt alkalmazásban jelen vannak, a webes felületektől a mobilalkalmazásokig. Ami a biztonsági dimenziót illeti, kulcsfontosságú, hogy minden CRUD-műveletet megfelelő hitelesítési és engedélyezési lépések védjenek, így csak a jogosult felhasználók férhetnek hozzá a kritikus adatokhoz. Például a JWT technológia használatával a szerver csak akkor teszi lehetővé az adatok frissítését vagy törlését, ha a felhasználó sikeresen hitelesítette és rendelkezik a megfelelő jogosultságokkal, ezáltal csökkentve az adatokkal való visszaélés kockázatát.

WPF Alkalmazás és karbantartás

A WPF egyik legnagyobb előnye, hogy XAML (eXtensible Application Markup Language) alapú felületleíró nyelvet használ, amely lehetővé teszi a felhasználói felület vizuális és logikai elemeinek elkülönítését, így a tervezők és a fejlesztők hatékonyabban tudnak együtt dolgozni. Emellett a WPF támogatja az adatkötési funkciót, amely automatizálja az adatok és az interfész elemei közötti szinkronizálást, így csökkenti a redundáns kód mennyiségét és növeli az alkalmazás válaszidejét. A keretrendszer kiterjedt animációs és grafikus képességei lehetővé teszik az alkalmazások lenyűgöző vizuális effektusokkal való gazdagítását, amelyek javítják a végfelhasználói élményt. A WPF által kínált MVVM (Model-View-ViewModel) architektúra elősegíti a fejlesztési mechanizmusok áttekinthetőségét és a program karbantarthatóságát, mivel a modell, az észlelés és a nézetmodell rétegek szétválasztása segít a fejlesztési feladatok jobb megszervezésében. Végül a Visual Studio integrált fejlesztői környezetének használata, beleértve a fejlesztői eszközöket és a hibakereső képességeket, tovább egyszerűsíti a WPF-alkalmazások fejlesztését és karbantartását, így biztosítva, hogy a hallgatói csapatok gyorsan és hatékonyan kezelhessék projekteiket.

Adatbázisok Bevezetése

Mi, mint informatikai szakos tanulók, sok esetben találkozunk adatbázis-kezelő rendszerekkel, amik az adatok tárolására, lekérdezésére, frissítésére, továbbá kezelésére szolgálnak. Ezek nélkül a rendszerek nélkül a mi általunk fejlesztett alkalmazások sem működhetnének effektíven, hiszen az adatbázisok biztosítják az adatok strukturált, ezen kívül könnyen hozzáférhető tárolását. Az adatbázisok különféle típusai, mint a relációs, NoSQL, esetleg a NewSQL, különféle igényeket szolgálnak ki, emiatt a projektünk alkalmával releváns volt a megfelelő csoport kiválasztása. A relációs adatbázisok, mint a MySQL, ideálisak voltak számunkra, mert jól strukturált, komplex lekérdezéseket tesznek lehetővé, amik elengedhetetlenek voltak a feladatinkhoz.

MySQL: A Választott Adatbázis-kezelő

A MySQL egy széles körben használt, nyitott forráskódú relációs adatbázis-kezelő rendszer. Az SQL nyelvet alkalmazza, ami az adatmanipuláció standard eszköze. Különösen kedvelt a webfejlesztők körében a megbízhatósága, továbbá a könnyű integrálhatósága okán. Mi is a MySQL-t választottuk projektünk adatbázis-kezelési feladataira, mivel jól funkcionál együtt a PHP, ezen kívül a Apache webszerverrel, amit szintén sok esetben használunk. A MySQL egyik legnagyobb előnye a rugalmassága, valamint a mindenre kiterjedő támogatottsága. Az ingyenes elérhetőség mellett, a közösségi, továbbá kereskedelmi segítség is biztosított, így könnyen hozzá tudtunk férni segítséghez, továbbá erőforrásokhoz, amikor csupán szükségünk volt rá.

Webszerverek és Tárhely

A webszerverek esszenciális szerepet töltenek be a weboldalak, továbbá webes alkalmazások életében. Ezek a szerverek tárolják a weboldalakhoz tartozó állományokat, mint példának okáért HTML, CSS, ezen kívül JavaScript fájlokat, valamint teszik őket elérhetővé az interneten keresztül. Emellett, a biztonsági protokollok kezelése, valamint a felhasználói kérések professzionális irányítása is a webszerverek teendője. A webszerverek kiválasztásánál figyelembe vettük a teljesítmény, biztonság, továbbá megbízhatóság szempontjait is, amik elengedhetetlenek voltak a stabil működéshez. Az IIS mellett másik sok esetben használt webszerver a Apache, mely szintén széles körű funkcionalitást, valamint konfigurálhatóságot kínál, de mi az IIS-t részesítettük előnyben a Microsoft ökoszisztéma okán.

IIS

Az IIS, esetleg Internet Information Services, a Microsoft által kifejlesztett webszerver, amit különösen a Windows szerver környezetekben használnak. Az IIS támogatja a HTTP, HTTPS, FTP, FTPS, SMTP, valamint NNTP protokollokat, tehát többféle szolgáltatást képes nyújtani. Mi is az IIS-t alkalmazzuk a szerveroldali technológiák tanulmányozása alkalmával, mert jól integrálódik más Microsoft termékekkel, mint a .NET keretrendszer, valamint a Microsoft SQL Server, melyeket

szintén tanulmányozunk. Az IIS biztosítja a szükséges skálázhatóságot, továbbá biztonsági funkciókat, amik elengedhetetlenek a webes alkalmazások zavartalan működéséhez. Ezek a funkciók magukban foglalják a kérések menedzselését, az autentikációs protokollok kezelését, ezen kívül a szerver állapotának permanens monitorozását, ami elengedhetetlen a biztonságos, valamint effektív működés szempontjából.

Tárhely Szolgáltatások

A tárhely cégek elengedhetetlen szereplői a webes jelenlétnek. Ők biztosítják azokat az adattárolási kapacitásokat, melyekre az összes online programnak szüksége van. Ezek lehetnek dedikált fizikai szerverek, esetleg virtuális szerverek vagy felhőalapú tárolási megoldások, mint az AWS, Google Cloud, esetleg Azure. A választás alkalmával figyelembe vettük a költségeket, a skálázhatóságot, valamint az elérhetőséget is. A felhőszolgáltatások különösen vonzóak voltak számunkra, mivel ezek dinamikusan alkalmazkodnak a forgalom változásaihoz, ami kritikus lehet nagyobb történések, esetleg marketing kampányok alkalmával. Ezen szolgáltatások biztonsági intézkedései, mint az adatok automatikus titkosítása, valamint a redundáns adattárolás, tovább növelik a bizalom mértékét a mi általunk készített alkalmazások stabilitása iránt.

Felhőalapú szolgáltatásokat részesítjük előnyben, mivel ezek flexibilisen skálázhatók, továbbá könnyedén kezelhetők. Ezek a platformok lehetővé teszik számunkra, hogy projektjeinket rapid módon skálázzuk, továbbá reagáljunk a változó igényekre anélkül, hogy drága fizikai infrastruktúrát kellene fenntartanunk. Az adatbiztonság, továbbá a hozzáférhetőség (accessibility) mellett ezek a platformok nagyszerű adminisztrációs instrumentumokat is kínálnak, amik segítenek a rendszergazdáknak az effektív erőforrás-kezelésben, továbbá a hibaelhárításban.

PHPMyAdmin

A phpMyAdmin egy ingyenes, webalapú adatbázis-kezelő eszköz, amelyet elsősorban a MySQL adatbázisok kezelésére terveztek. Ez a program lehetővé teszi a felhasználók számára, hogy könnyedén és intuitív módon kezeljék adatbázisaikat a böngészőjük segítségével. A phpMyAdmin grafikus felhasználói felülete egyszerű és átlátható, így a felhasználók könnyen navigálhatnak az adatbázisokban, táblákban és rekordokban.

Az eszköz széleskörű funkcionalitást kínál, lehetővé téve az adatbázisok és táblák létrehozását, módosítását és törlését, valamint SQL lekérdezések futtatását és sok

más művelet elvégzését. Ezen felül a phpMyAdmin erős biztonsági funkciókat is biztosít, például SSL titkosítást, amely növeli az adatok védelmét és biztonságát.

Az eszköz hordozható jellege lehetővé teszi a hozzáférést bármilyen eszközről és helyről, ahol internetkapcsolat áll rendelkezésre. Emellett a phpMyAdmin nyílt forráskódú, ami azt jelenti, hogy ingyenesen használható és módosítható, valamint széleskörű közösségi támogatást és fejlesztést kínál.

Összességében a phpMyAdmin egy megbízható és népszerű megoldás azok számára, akik MySQL adatbázisokkal dolgoznak. A könnyű telepítés, a felhasználóbarát felület, a széleskörű funkcionalitás és a biztonsági funkciók révén a phpMyAdmin ideális eszköz a kisebb és közepes méretű weboldalak, blogok és alkalmazások adatbázis-kezeléséhez.

Csapat kommunikációs eszközök (Github, Discord, Messenger, Google drive)

Github

A GitHub egy olyan platform, ami elsődlegesen a szoftverfejlesztőknek alakult ki, hogy közösen tudjanak dolgozni a projekteken. Támogatásával könnyedén lehet verziókat kezelni, vagyis ha elrontottunk valamit, vissza tudunk lépni az előző, funkcionáló verzióra. Amikor a webshopunkat csináltuk, a GitHubon tároltuk a kódjainkat, hogy bármikor hozzáférhessünk. Nagyon értékes, hogy többen is dolgozhatunk egy fájlon belül anélkül, hogy összeakadnánk. Továbbá, a GitHubon keresztül követhetjük, hogy ki milyen változtatásokat hajtott végre, ami megkönnyíti a csapatmunkát. A pull requestek támogatásával javaslatokat tehetünk a kódba, amiket a csapattagok átnézhetnek, továbbá jóváhagyhatnak, így biztosítva a kód minőségét. A projektünkhöz használt issue tracker arra szolgált, hogy nyomon követhessük a felmerülő nehézségeket, valamint feladatokat.

Discord

A Discord egy kommunikációs platform, ami különösen népszerű a játékosok, ezen kívül a technológiai közösségek körében. Lehetőséget biztosít szöveges, hang-, valamint videóalapú kommunikációra is. Mi is használtuk a Discordot a webshop projektünk alkalmával, mert praktikus volt rapid üzenetváltásra, ezen kívül

megbeszélésekre. Egy privát szerveren belül több csatornát is létrehoztunk különféle témákra, így az összes lényeges adat egy helyen volt. Ráadásul, mivel sokan játszunk is, így a munka mellett volt idő egy kis kikapcsolódásra is. A Discord ezzel együtt lehetőséget adott arra, hogy dokumentumokat, továbbá képeket is megosszunk közvetlen módon a csatornákon, ami megkönnyítette a vizuális anyagok cseréjét. Az élő hangbeszélgetések funkciója különösen értékes volt a hosszabb megbeszélésekhez.

Messenger

A Messenger, amit sokan csupán Facebook Messengerként ismernek, egy azonnali üzenetküldő alkalmazás. Egyetemistaként sokat használjuk, mert rapid módon, ezen kívül egyszerűen lehet vele kommunikálni. A webshop projektünk alatt főként rövid egyeztetésekre használtuk, amikor rapid válaszra volt szükség. Alternatíva van csoportok létrehozására is, ami segített abban, hogy a program az összes résztvevőjét naprakészen tartsuk az érvényben lévő feladatokról, továbbá változásokról. A Messenger videóhívás funkciója is sokat segített, amikor távolról kellett dolgoznunk, így az arcok láthatósága is hozzáadott a kommunikáció személyességéhez. A Messenger gyorsasága, ezen kívül közvetlensége okán sok esetben volt az primer eszközünk sürgős nehézségek megoldására.

Google Drive

A Google Drive egy felhőalapú tároló szolgáltatás, amit arra használunk, hogy dokumentumokat, táblázatokat, ezen kívül egyéb fájlokat tároljunk, valamint osszunk meg egymással. Nagyon praktikus, mert bárhol, bármikor hozzáférhetünk a fájljainkhoz, ha van internetkapcsolatunk. A webshopunkhoz kapcsolódó összes dokumentációt, terveket, valamint grafikákat itt tároltuk, így mindkettőnk részére elérhető volt. A Drive integrálható más Google-szolgáltatásokkal is, mint példának okáért a Docs, esetleg a Sheets, ami további előnyöket biztosít a csapatmunka alkalmával. A dokumentumokon való egyidejű munka lehetősége, továbbá a változások automatikus mentése, ezen kívül verziókövetése kiemelkedően értékes volt a projektünk alkalmával. Ezen felül, a Google Drive biztonsági funkciói, mint a kétfaktoros hitelesítés, extra védelmet nyújtottak adatainknak.

A Frontend Mélyebben

A Login :

```
import Footer from './Footer';
import Login from './Login';
import User from './User'; // Importáld a User komponenst
import { Hover } from './Hover';
import './App.css';
const App = () => {

  return (
    <Router>
      <div className="app-container">

          <Navbar />

          <InfiniteScrollBanner />



          <div className="content-container">
            <Routes>
              <Route path="/" element={<Home />} />
              <Route path="/women" element={<Women />} />
              <Route path="/men" element={<Men />} />
              <Route path="/children" element={<Children />} />
              <Route path="/shoes" element={<Shoes />} />
              <Route path="/login" element={<Login />} />
              <Route path="/user" element={<User />} />
            </Routes>
          </div>

          <Hover />
        <Footer />
      </div>
    </Router>
  );
};

export default App;
```

Ez a kód egy React alkalmazást implementál, mely alkalmazza a React Router könyvtárat az útvonalak kezelésére. A BrowserRouter összetevő támogatásával kialakít egy útvonalakat kezelő konténert. A <Routes> komponensben definiálja az alkalmazás útvonalait, ezen kívül a <Route> összetevők az egyes útvonalakhoz rendelik hozzá a jó komponenseket, melyeket a felhasználó a böngészőben kattintva, esetleg navigálva elérhet.

Példának okáért az /women útvonalhoz rendelt összetevő a Women, mely a női termékeket mutatja be az alkalmazásban. Hasonlóan, az /login útvonal a Login komponenssel van .összekapcsolva, mely a felhasználó bejelentkezési felületét tartalmazza.

Ez a kód biztosítja az alkalmazás navigációját, továbbá az egyes oldalak tartalmának dinamikus megjelenítését az útvonalak alapján.

To the top:

```jsx
import React, { useState, useEffect } from 'react';

export const Hover = () => {
  // Állapot definiálása a gomb megjelenítésének vezérlésére
  const [showButton, setShowButton] = useState(false);

  // Scroll eseménykezelő funkció definiálása
  useEffect(() => {
    const handleScroll = () => {
      // Ellenőrizze a görgetési pozíciót, és állítsa be az állapotot ennek megfelelően
      if (window.scrollY > 20) {
        setShowButton(true);
      } else {
        setShowButton(false);
      }
    };

    // Scroll esemény figyelése
    window.addEventListener('scroll', handleScroll);

    // Eseményfigyelő eltávolítása a komponens megszűnésekor (takarítás)
    return () => {
      window.removeEventListener('scroll', handleScroll);
    };
  }, []); // Az üres tömb azt jelzi, hogy az useEffect csak a komponens mount és unmount esetén fussa le

  // Vissza a tetejére gombra kattintáskor végrehajtott funkció
  const topFunction = () => {
    document.body.scrollTop = 0; // Safari támogatás
    document.documentElement.scrollTop = 0; // Chrome, Firefox, IE és Opera támogatás
  };

  // A komponens JSX kódja
  return (
    <div>
      {/* Vissza a tetejére gomb */}
      <button
        id="myBtn"
        onClick={topFunction}
        title="Vissza a tetejére"
        style={{ display: showButton ? 'block' : 'none' }}
      >
        Vissza a tetejére ↑
      </button>
    </div>
```

Ez a kód egy React komponenst definiál, ami egy "Vissza a tetejére" gombot jelenít meg. Amikor a felhasználó görgeti az oldalt lefelé, és eléri a 20 pixeltől való távolságot az oldal tetejétől, akkor a gomb megjelenik.

A komponens használja a useState hookot az állapot kezelésére, amely meghatározza, hogy meg kell-e jeleníteni a gombot vagy sem. A useEffect hook segítségével figyeljük a görgetési eseményeket, és dinamikusan változtatjuk az állapotot ezek alapján.

Amikor a gombra kattintanak, egy topFunction függvény hívódik meg, ami visszaviszi a felhasználót az oldal tetejére.

Összességében a kód egy egyszerű és hatékony módja annak, hogy egy vissza a tetejére gombot jelenítsünk meg dinamikusan egy React alkalmazásban.

Ez a komponens ideális választás lehet olyan weboldalakhoz, ahol hosszú tartalmat kell görgetni, és a felhasználóknak könnyen el kell jutniuk az oldal tetejére. A "Vissza a tetejére" gomb segítségével egyszerűen és gyorsan lehet visszajutni az oldal

elejére, anélkül, hogy a felhasználónak manuálisan kellene görgetnie az egész tartalmat. Ezáltal növelhető a felhasználói élmény és a weboldal használhatósága.

Adatbázis

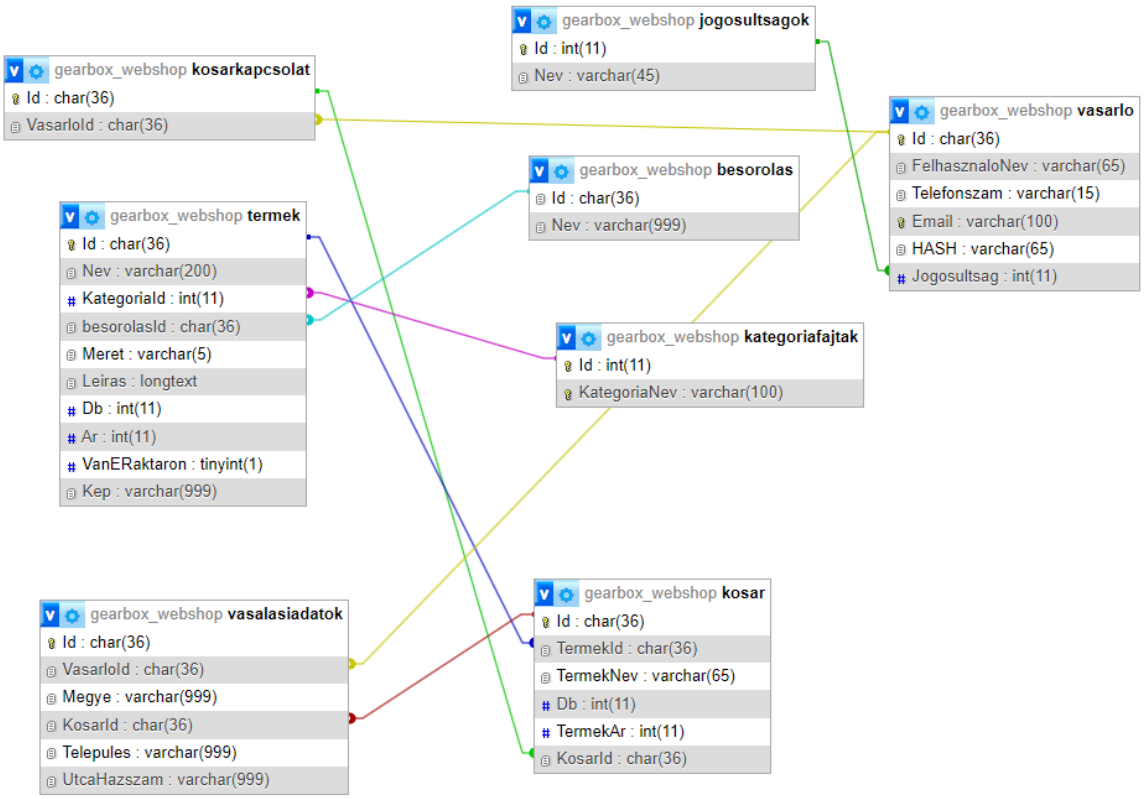
A fájl a következő részekből áll:

Beállítások: A fájl elején találhatók az SQL beállítások, mint például az SQL mód, időzóna beállítások stb.
Adatbázis létrehozása: Ebben a részben van a CREATE DATABASE parancs, ami létrehozza az adatbázist, ha még nem létezik, valamint a USE parancs, ami azt mondja meg a MySQL-nek, hogy az azt követő műveletek az adott adatbázison fognak történni.
Táblák létrehozása: Minden tábla létrehozása a CREATE TABLE parancsokkal. Itt láthatók az oszlopok nevei, típusai, kulcsai stb.
Adatok beszúrása: Az adott táblákba tartozó adatok beszúrása a INSERT INTO parancsokkal. Ezek a parancsok a tábla struktúrájának megfelelő adatokat illesztenek be.
A dump fájlok rendkívül hasznosak lehetnek az adatbázisok mentéséhez, migrálásához vagy másolásához, mivel lehetővé teszik az adatbázis struktúrájának és tartalmának egyszerű visszaállítását vagy másolását egy másik rendszerbe.

☐	besorolas	☆	Tartalom	Szerkezet	Keresés	Beszúrás	Kiürítés	Eldobás	4	InnoDB	utf8mb4_hungarian_ci	32.0 KB	-
☐	jogosultsagok	☆	Tartalom	Szerkezet	Keresés	Beszúrás	Kiürítés	Eldobás	2	InnoDB	utf8mb4_hungarian_ci	16.0 KB	-
☐	kategoriafajtak	☆	Tartalom	Szerkezet	Keresés	Beszúrás	Kiürítés	Eldobás	5	InnoDB	utf8mb4_hungarian_ci	32.0 KB	-
☐	kosar	☆	Tartalom	Szerkezet	Keresés	Beszúrás	Kiürítés	Eldobás	0	InnoDB	utf8mb4_hungarian_ci	64.0 KB	-
☐	kosarkapcsolat	☆	Tartalom	Szerkezet	Keresés	Beszúrás	Kiürítés	Eldobás	0	InnoDB	utf8mb4_hungarian_ci	32.0 KB	-
☐	termek	☆	Tartalom	Szerkezet	Keresés	Beszúrás	Kiürítés	Eldobás	67	InnoDB	utf8mb4_hungarian_ci	80.0 KB	-
☐	vasalasiadatok	☆	Tartalom	Szerkezet	Keresés	Beszúrás	Kiürítés	Eldobás	0	InnoDB	utf8mb4_hungarian_ci	48.0 KB	-
☐	vasarlo	☆	Tartalom	Szerkezet	Keresés	Beszúrás	Kiürítés	Eldobás	11	InnoDB	utf8mb4_hungarian_ci	48.0 KB	-
	8 tábla	Összesen							89	InnoDB	utf8mb4_hungarian_ci	352.0 KB	0 B

besorolas: Tartalmazza a kategóriák besorolását.
jogosultsagok: Felhasználói jogosultságokat tárol.
kategoriafajtak: A kategóriák típusait vagy fajtáit tartalmazza.
kosar: Felhasználók kosarában lévő elemeket tárol.
kosarkapcsolat: Felhasználók és a kosárban lévő elemek közötti kapcsolatokat tárolja.

termek: Webáruházban elérhető termékeket tárol.
vasalasiadatok: Vásárlási adatokat tárol, például a rendelések adatait.
vasarlo: Vásárlókkal kapcsolatos információkat tartalmazza.



SQL Kapcsolatok: Az Adatok Hidakon Át

SQL kapcsolatok olyan fontos hídként szolgálnak az adatok között, mint amilyen egy személyes út a világ két pontja között. Ezek a kapcsolatok adataink mozgását segítik elő az adatbázisok és az alkalmazások között, lehetővé téve az adatok lekérdezését, frissítését és kezelését.

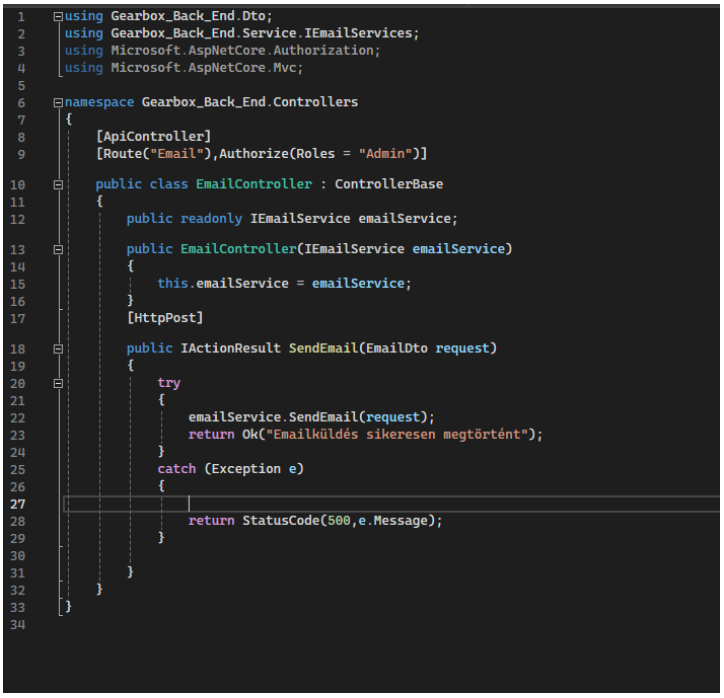
Gondolj rájuk úgy, mint egy telefonvonalra a két barát között. Az egyik barát elküldi az üzenetet (lekérdezés), a másik pedig válaszol (eredmények visszaküldése). Ez a kapcsolat lehetővé teszi a kommunikációt és az információk cseréjét.

Az SQL kapcsolatok lehetnek helyi vagy távoli. A helyi kapcsolatok azonos gépen futó adatbázisokhoz való kapcsolódást jelentenek, míg a távoli kapcsolatok az interneten vagy egy belső hálózaton keresztül érik el az adatbázist.

Az adatbázis-kezelő rendszer, például MySQL vagy PostgreSQL, biztosítja az SQL kapcsolatok kezelését. Az alkalmazások különböző programnyelvekben, például Python, Java vagy PHP segítségével kommunikálhatnak ezekkel a kapcsolatokkal.

Fontos megjegyezni, hogy a kapcsolatoknak vannak biztonsági vonatkozásai is. Jogosulatlan hozzáférést megakadályozó intézkedéseket kell alkalmazni, hogy az adatok ne essenek rossz kezekbe.

Back end



```
using Gearbox_Back_End.Dto;
using Gearbox_Back_End.Service.IEmailServices;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace Gearbox_Back_End.Controllers
{
    [ApiController]
    [Route("Email"),Authorize(Roles = "Admin")]

    public class EmailController : ControllerBase
    {
        public readonly IEmailService emailService;

        public EmailController(IEmailService emailService)
        {
            this.emailService = emailService;
        }
        [HttpPost]

        public IActionResult SendEmail(EmailDto request)
        {
            try
            {
                emailService.SendEmail(request);
                return Ok("Emailküldés sikeresen megtörtént");
            }
            catch (Exception e)
            {

                return StatusCode(500,e.Message);
            }

        }
    }
}
```

Ez egy C# nyelven írt ASP.NET Core Web API vezérlő osztály. A EmailController nevű osztály egy HTTP POST kérést kezel a /Email végponton keresztül. Az osztályt az [ApiController] attribútummal jelölik meg, ami jelzi, hogy ez egy vezérlő, és az alapértelmezett Web API viselkedést fogja használni.

A [Route("Email")] attribútum meghatározza az útvonalat, amelyen keresztül a vezérlő elérhető lesz, azaz a /Email útvonalat. Az [Authorize(Roles = "Admin")] attribútum arra utal, hogy az akció csak az "Admin" jogosultsággal rendelkező felhasználók számára hozzáférhető.

A EmailController konstruktorában egy IEmailService típusú függőséget injektálnak, amelyet az osztály tagváltozójaként tárolnak.

Az EmailController osztályban van egy SendEmail elnevezésű akciómetódus, ami az HTTP POST kéréseket kezeli. Ez a metódus egy EmailDto típusú paramétert vár, amelyet request néven adnak át. A metódus megpróbálja elküldeni az e-mailt a emailService.SendEmail() metódus segítségével, majd a kérést siker esetén az "Emailküldés sikeresen megtörtént" szöveggel válaszol, ellenkező esetben pedig egy 500-as hibaüzenettel és az esetleges hibaüzenettel.

Az osztály tartalmaz néhány using utasítást is, amelyek importálják a szükséges névtereket és típusokat a forráskódhoz.

```
{
    [ApiController]
    [Route("/login")]
    1 reference
    public class LoginController : ControllerBase
    {
        private readonly IConfiguration _configuration;
        0 references
        public LoginController(IConfiguration configuration)
        {
            _configuration = configuration;
        }
        [HttpPost("/regisztracio")]
        0 references
        public ActionResult<RegisterVasarlo> Register(RegisterVasarlo registerVasarlo)
        {
            var UjVasarlo = new Vasarlo
            {
                Id = new Guid(),
                FelhasznaloNev = registerVasarlo.Felhasznalonev,
                Telefonszam = registerVasarlo.Telefonszam,
                Hash = BCrypt.Net.BCrypt.HashPassword(registerVasarlo.Jelszo),
                Email = registerVasarlo.Email,

                Jogosultsag = 0

            };

            using (var context = new GearBoxDbContext())
            {
                if (context != null)
                {
                    context.Vasarlos.Add(UjVasarlo);
                    context.SaveChanges();
                    return StatusCode(201, "sikeres regisztráció!");
                }
                else
```

A LoginController nevű osztály két HTTP POST kérést kezel a /regisztracio és /bejelentkezes végpontokon keresztül.

A LoginController az [ApiController] attribútummal van ellátva, ami jelzi, hogy ez egy vezérlő, és az alapértelmezett Web API viselkedést fogja használni. Az [Route("/login")] attribútum meghatározza az útvonalat, amelyen keresztül a vezérlő elérhető lesz, azaz a /login útvonalat.

A vezérlő konstruktorában a IConfiguration függőség injektálódik be, amely segítségével hozzáférhetünk az alkalmazás konfigurációs adatokhoz.

Az osztály tartalmaz két akciómetódust: Register és Login. A Register metódus egy új felhasználót regisztrál az adatbázisba, az adatokat a registerVasarlo paraméterből veszi át. A jelszót a BCrypt algoritmus segítségével titkosítja, majd elmenti az adatbázisba. A Login metódus ellenőrzi a felhasználó bejelentkezési adatait, majd JWT tokent hoz létre, amelyet válaszként küld vissza, ha a bejelentkezés sikeres.

Mindkét metódus tartalmaz hibakezelést, és a válasz státuszának beállítását a különböző helyzetekben.

```
1 reference
private string CreateToken(Vasarlo vasarlo)
{
    using (var context = new GearBoxDbContext())
    {
        List<Claim> claims = new List<Claim>
        {
        new Claim(ClaimTypes.Role,context.Jogosultsagoks.FirstOrDefault(x=>x.Id==vasarlo.Jogosultsag).Nev),
        new Claim(ClaimTypes.Email,vasarlo.Email),
        new Claim(ClaimTypes.Name,vasarlo.FelhasznaloNev),
        new Claim(ClaimTypes.NameIdentifier,vasarlo.Id.ToString()),
        new Claim(ClaimTypes.MobilePhone,vasarlo.Telefonszam.ToString())
        };

        var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration.GetSection("AppSettings:Token").Value!));

        var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha512Signature);

        var token = new JwtSecurityToken(
            claims: claims,
            expires: DateTime.Now.AddDays(1),
            audience: _configuration.GetValue<string>("Authentication:Schemes:Bearer:ValidAudiences:0"),
            issuer: _configuration.GetSection("Authentication:Schemes:Bearer:ValidIssuer").Value,
            signingCredentials: creds);
        var jwt = new JwtSecurityTokenHandler().WriteToken(token);

        return jwt;
    }
}
```

```
using Gearbox_Back_End.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace Gearbox_Back_End.Controllers
{
    [ApiController]
    [Route("/Jogosultsag"), Authorize(Roles = "Admin")]
    0 references
    public class JogosultsagokController : ControllerBase
    {
        [HttpPost]
        0 references
        public ActionResult<JogosultsagDto> Post(CreateOrModifyJogosultsagok createOrModifyJogosultsagok)
        {
            try
            {
                var UjJog = new Jogosultsagok
                {
                    Nev = createOrModifyJogosultsagok.Nev,
                };
                using (var context = new GearBoxDbContext())
                {
                    if (context != null)
                    {
                        context.Jogosultsagoks.Add(UjJog);
                        context.SaveChanges();
                        return StatusCode(201, "Az adatok sikeresen eltárolva!");
                    }
                    else
                    {
                        return StatusCode(406, "Nem megfelelő az adat formátuma!");
                    }
                }
            }
            catch (Exception e)
            {
                return StatusCode(500, e.Message);
            }

        }
```

A JogosultsagokController nevű osztály a /Jogosultsag útvonalon keresztül érhető el, és csak az "Admin" szerepkörrel rendelkező felhasználók számára elérhető.

Az osztály tartalmazza az összes alapvető CRUD műveletet (létrehozás, olvasás, frissítés, törlés) a jogosultságokkal kapcsolatban. Az akciómetódusok a HttpPost, HttpGet, HttpPut, és HttpDelete HTTP műveleteket kezelik, és megfelelően kezelik a hibákat.

Minden akciómetódus tartalmaz egy try-catch blokkot, hogy kezelje a potenciális kivételeket, és megfelelően állítja be a válasz státuszát a különböző helyzetekben. Az adatbázis műveleteket az Entity Framework segítségével végzi, amelyet az GearBoxDbContext osztály használ.

Az Authorize attribútum segítségével a vezérlőhöz csak az "Admin" szerepkörrel rendelkező felhasználók férhetnek hozzá, amely lehetővé teszi a jogosultságok megfelelő kezelését és védelmét.

A TermekController nevű osztály a /Termek útvonalon keresztül érhető el.

Az osztály tartalmazza az összes alapvető CRUD műveletet (létrehozás, olvasás, frissítés, törlés) a termékekkel kapcsolatban. Az akciómetódusok a HttpPost, HttpGet, HttpPut, és HttpDelete HTTP műveleteket kezelik.

Minden akciómetódus tartalmaz egy try-catch blokkot, hogy kezelje a potenciális kivételeket, és megfelelően állítja be a válasz státuszát a különböző helyzetekben. Az adatbázis műveleteket az Entity Framework segítségével végzi, amelyet az GearBoxDbContext osztály használ.

Az Authorize attribútum segítségével a vezérlőhöz csak az "Admin" szerepkörrel rendelkező felhasználók férhetnek hozzá, amely lehetővé teszi a termékek megfelelő kezelését és védelmét. A Search akciómetódus keresést tesz lehetővé a termékek között a név alapján.

```
[HttpPut("{id}"), Authorize(Roles = "Admin")]
0 references
public ActionResult<TermekDto> Put(Guid id, CreateOrModifyTermek createOrModifyTermek)
{
    try
    {
        using (var context = new GearBoxDbContext())
        {
            if (context != null)
            {
                var valtoztatando = context.Termeks.FirstOrDefault(x => x.Id == id);
                if (valtoztatando != null)
                {
                    valtoztatando.Nev = createOrModifyTermek.Nev;
                    valtoztatando.KategoriaId = createOrModifyTermek.Kategoria;
                    valtoztatando.BesorolasId = createOrModifyTermek.BesorolasId;
                    valtoztatando.Meret = createOrModifyTermek.Meret;
                    valtoztatando.Leiras = createOrModifyTermek.Leiras;
                    valtoztatando.Db = createOrModifyTermek.Db;
                    valtoztatando.Ar = createOrModifyTermek.Ar;
                    valtoztatando.VanEraktaron = createOrModifyTermek.VanEraktaron;
                    valtoztatando.Kep = createOrModifyTermek.Kep;

                    context.Termeks.Update(valtoztatando);
                    context.SaveChanges();
                    return Ok("Sikeres adatváltoztatás!");
                }
                else
                {
                    return StatusCode(404, "A keresett termék nem létezik, vagy nincs eltárolva");
                }
            }
            else
            {
                return StatusCode(503, "A szerver jelenleg nem elérhető");
            }
        }
    }
    catch (Exception e)
    {
        return StatusCode(500, e.Message);
    }

}
```

```
namespace Gearbox_Back_End.Controllers
{
    [ApiController]
    [Route("/VasarlasiAdatok")]
    0 references
    public class VasarlasiAdatokController : ControllerBase
    {
        [HttpPost]
        0 references
        public ActionResult<VasarlasiAdatokDto> Post(CreateOrModifyVasarlasiAdatokDto createOrModifyVasarlasiAdatokDto)
        {
            try
            {
                var UjVasarlas = new Vasalasiadatok
                {
                    Id = new Guid(),
                    VasarloId = createOrModifyVasarlasiAdatokDto.VasarloId,
                    Megye = createOrModifyVasarlasiAdatokDto.Megye,
                    KosarId = createOrModifyVasarlasiAdatokDto.KosarId,
                    Telepules = createOrModifyVasarlasiAdatokDto.Telepules,
                    UtcaHazszam = createOrModifyVasarlasiAdatokDto.UtcaHazszam
                };
                using (var context = new GearBoxDbContext())
                {
                    if (context != null)
                    {
                        context.Vasalasiadatoks.Add(UjVasarlas);
                        context.SaveChanges();
                        return StatusCode(201, "Az adatok sikeresen eltárolva!");
                    }
                    else
                    {
                        return StatusCode(406, "Nem megfelelő az adat formátuma!");
                    }
                }
            }
            catch (Exception e)
            {
                return StatusCode(500, e.Message);
            }

        }
```

A VasarlasiAdatokController nevű osztály a /VasarlasiAdatok útvonalon keresztül érhető el.

Az osztály felelős az ügyfelek vásárlási adatainak kezeléséért. Tartalmazza az összes alapvető CRUD műveletet (létrehozás, olvasás, frissítés, törlés) a vásárlási adatokkal kapcsolatban. Az akciómetódusok a HttpPost, HttpGet, HttpPut és HttpDelete HTTP műveleteket kezelik.

Minden akciómetódus tartalmaz egy try-catch blokkot a potenciális kivételek kezelésére, és megfelelően állítja be a válasz státuszát a különböző helyzetekben. Az adatbázisműveleteket az Entity Framework segítségével végzi, amelyet a GearBoxDbContext osztály használ.

Az osztály nem igényel speciális hozzáférési jogokat, azaz mindenki számára elérhető. A vezérlő felelős az ügyfelek vásárlási adatainak kezeléséért, ideértve a létrehozást, listázást, módosítást és törlést.

A VasarlasiAdatokController osztály egy fontos része a Gearbox Back-End alkalmazásnak, amely az ügyfelek vásárlási adatait kezeli. Az osztály különböző HTTP műveleteket biztosít, mint például az adatok létrehozása, lekérdezése, frissítése és törlése. Minden művelet megfelelő státuszkóddal és hibakezeléssel rendelkezik. Az Entity Framework segítségével az osztály könnyen kommunikál az adatbázissal, biztosítva a hatékony adatkezelést.

A fenti leírásokban bemutatott kontrollerek, vagyis a JogosultsagokController, a TermekController és a VasarlasiAdatokController, az alkalmazás három kulcsfontosságú részét alkotják. Ezek a vezérlők felelősek az ügyféljogosultságok kezeléséért, a termékek adatainak tárolásáért és az ügyfelek vásárlási információinak kezeléséért.

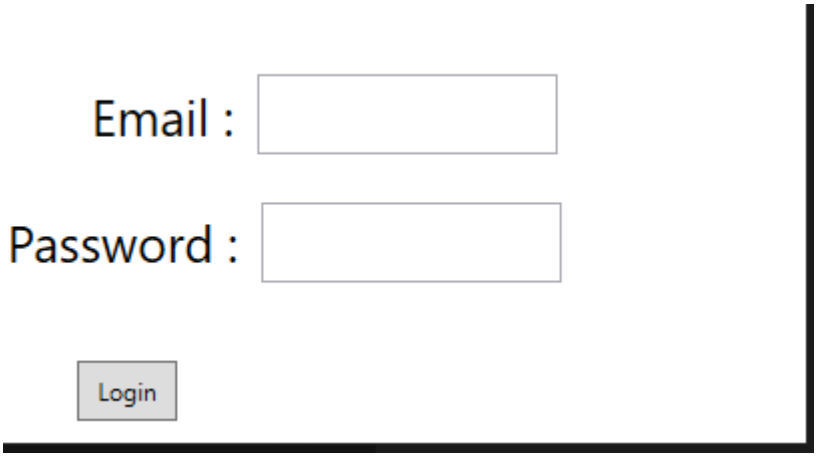
A JogosultsagokController a felhasználói jogosultságokkal kapcsolatos műveleteket végzi, lehetővé teszi az új jogosultságok létrehozását, meglévők lekérdezését, módosítását és törlését. Ezen műveletek segítségével az alkalmazás rugalmasan kezelheti az egyes felhasználók jogosultságait, és azokat a biztonsági szempontból releváns szintre állíthatja.

A TermekController felelős a termékekkel kapcsolatos műveletekért, ideértve az új termékek hozzáadását, meglévők lekérdezését, szerkesztését és törlését. Ez az osztály lehetővé teszi az alkalmazás számára, hogy dinamikusan kezelje a termékadatokat, így könnyen frissítheti és karbantarthatja azokat.

A VasarlasiAdatokController az ügyfelek vásárlási adatainak kezeléséért felelős. Az alkalmazás ezen részével az ügyfelek vásárlási információi könnyen kezelhetők, lehetővé téve a rendelések hozzáadását, lekérdezését, szerkesztését és törlését. Ez az osztály fontos szerepet tölt be az ügyfélinterakciók követésében és az ügyfelek vásárlási folyamatainak hatékony kezelésében.

Mindezek a kontrollerek szorosan integrálódnak az alkalmazás többi részével, biztosítva az adatok konzisztens kezelését és a funkcionalitás folyamatos biztosítását. A megfelelő HTTP státuszkódok visszatérése segíti a klienseket az alkalmazásban lévő változások megértésében és kezelésében.

WPF



A mi WPF alkalmazásunkba való belépés szigorúan ellenőrzött folyamat, amely biztosítja a rendszer biztonságát és a megfelelő jogosultságok kezelését. A felhasználók csak bejelentkezés után férhetnek hozzá az alkalmazáshoz, és csak az adminisztrátoroknak van jogosultságuk a belépésre.

Az alkalmazásunkban a bejelentkezési folyamatot azonosításra és hitelesítésre használjuk, hogy ellenőrizzük a felhasználók azonosságát és jogosultságait. Az adminisztrátorok számára különleges bejelentkezési jogosultságokat biztosítunk, amelyek lehetővé teszik számukra a teljes körű rendszerelérést és kezelést.

A felhasználók bejelentkezési adatait biztonságosan tároljuk és kezeljük az alkalmazásban, például jelszavak hash-elt formában történő tárolásával és hitelesítésével. Az adatok biztonságának garantálása érdekében az alkalmazásban erősített biztonsági intézkedéseket alkalmazunk, például SSL titkosítást a kommunikációhoz és a felhasználói munkamenetek kezelését.

Az adminisztrátoroknak széles körű jogosultságokat biztosítunk az alkalmazásban, amelyek lehetővé teszik számukra az adatok megtekintését, szerkesztését és kezelését a rendszer teljes körű felügyelete érdekében. Az adminisztrátorok felhasználói munkameneteit és tevékenységeit naplózzuk és ellenőrizzük, hogy biztosítsuk az alkalmazás biztonságát és az adatok integritását.

```
/// <summary>
/// Interaction logic for MainWindow.xaml
/// </summary>
2 references
public partial class Login : Window
{
    0 references
    public Login()
    {
        InitializeComponent();
    }
    Connection connection = new Connection();
    1 reference
    private void windowchanger_Click(object sender, RoutedEventArgs e)
    {
        WebClient webClient = new WebClient();
        webClient.Headers[HttpRequestHeader.ContentType] = "application/json";
        webClient.Encoding = Encoding.UTF8;
        try
        {
            LoginDto loginDto = new LoginDto();
            loginDto.email = EmailAdress.Text;
            loginDto.jelszo = Password.Text;
            string result = webClient.UploadString(connection.Url() + "login/bejelentkezes", "POST", JsonConvert.SerializeObject(loginDto));
            var handler = new JwtSecurityTokenHandler();
            var jwtSecurityToken = handler.ReadJwtToken(JsonConvert.DeserializeObject<Token>(result).usertoken);
            if (jwtSecurityToken.Claims.First(x => x.Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/role").Value == "Admin")
            {
                Hub eloszto = new Hub(JsonConvert.DeserializeObject<Token>(result).usertoken);
                eloszto.Show();
                this.Close();
            }
            else
            {
                MessageBox.Show("Önnek nincs hozzáférési joga az alkalmazáshoz!");
            }

        }
        catch (Exception g)
        {
            MessageBox.Show("Hiba történt a bejelentkezés alatt! : " + g.Message);
        }
    }
}
```

Ez a WPF kód egy bejelentkező ablakot definiál az alkalmazásban. Amikor a felhasználó megadja az e-mail címét és jelszavát, és megnyomja a "Belépés" gombot (windowchanger_Click eseménykezelő), az alkalmazás elküldi ezeket az adatokat a szervernek a bejelentkezéshez.

A WebClient osztályt használjuk a HTTP kérések elküldésére a szerver felé. A bejelentkezési adatokat JSON formátumban küldjük el a szervernek, majd megkapjuk a választ.

Ha a bejelentkezés sikeres volt, az alkalmazás kiolvassa a JWT tokent a válaszból, majd ellenőrzi, hogy az aktuális felhasználó adminisztrátor-e. Ha igen, akkor átirányítja az adminisztrátori felületre (Hub ablak), különben egy üzenetablakban jelezni fogja, hogy nincs megfelelő jogosultsága az alkalmazás használatához.

Ha valamilyen hiba történik a bejelentkezés közben (pl. hibás bejelentkezési adatok vagy kapcsolódási problémák), akkor az alkalmazás egy hibaüzenetet jelenít meg a felhasználónak. Ez a kód biztosítja a felhasználók hitelesítését és jogosultságainak ellenőrzését az alkalmazásunkban.

Az alkalmazásunk új dimenziókat nyit meg a felhasználók számára, lehetőséget adva számukra a határok feszegetésére és az új tapasztalatok szerzésére. Egyszerűen használható felülete rejtett lehetőségeket és rejtélyeket rejteget, amelyek felfedezésre várnak. A felhasználók számára ez egy új utazás kezdetét jelenti, ahol a végső cél még ismeretlen, és a felfedezés öröme vezeti őket az ismeretlenbe

Források

Codepen: https://codepen.io/
ReactJsExample: https://reactjsexample.com/tag/templates/
WPF :
https://learn.microsoft.com/en-us/dotnet/desktop/wpf/get-started/create-app-visual-studio?view=netdesktop-8.0

c# : https://learn.microsoft.com/en-us/training/paths/build-dotnet-applications-csharp/

Stackoverflow: https://stackoverflow.com/