

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR



SZAKDOLGOZAT

OE-NIK	Hallgató neve:	**Varga Dorottya**
2024	Hallgató törzskönyvi száma:	**T/008649/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Varga Dorottya**

Törzskönyvi száma: T/008649/FI12904/N

Neptun kódja: [redacted]

A dolgozat címe:

Receptadatbázis és cukrászműhely kezelőrendszer

Recipe database and confectionery workshop management system

Intézményi konzulens: Rusznák Attila

Külső konzulens:

Beadási határidő: 2024. december 15.

A záróvizsga tárgyai: Számítógép architektúrák

Big Data és üzleti intelligencia

specializáció

A feladat

Felgyorsult életvitelünkhöz tartozik, hogy szeretünk mindent gyorsan, egyszerűen és egyhelyen intézni, ez a cukrászokra is igaz.

A hallgató feladata egy olyan webes alkalmazás megtervezése és megalkotása, mely képes a receptek, alapanyag készletek, ügyféladatok, rendelések és üzleti folyamatok kezelésére. További feladata a hallgatónak a szükséges eszközök és alkalmazások felkutatása és bemutatása. A minta rendszer elkészítése, implementálása, tesztelése, majd annak részletes bemutatása a meghatározott kritériumok alapján.

A dolgozatnak tartalmaznia kell:

- a feladat pontos ismertetését,
- a szakirodalom feldolgozását a szükségeg eszközök tekintetében,
- az adatbázis megtervezése és megvalósítása,
- a webalkalmazás kialakítása,
- az alkalmazás tesztelése,
- végül a továbbfejlesztési lehetőségek elemzése.



[redacted]
Dr. habil. Heller Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2026. december 15.**
(ÓE HKR 54.§ (10) bekezdés szerint)

A dolgozatot beadásra alkalmasnak tartom:

[redacted]

.....
külső konzulens

intézményi konzulens

ABSZTRAKT

A szakdolgozatom célja egy olyan webes alkalmazás megvalósítása, mely elősegíti, hatékonyabbá és egyszerűbbé teszi a cukrászműhelyek mindennapi munkáját. Az alkalmazás lehetőséget biztosít, hogy mindent egy helyen intézhessenek az üzleti folyamatoktól kezdve, a rendelések nyomon követése, a receptek, készletinformációk és ügyfél adatok elérésén át az ügyfelek számára elérhető weboldalig. Szeretném levenni a cukrászok válláról a háttérben zajló folyamatok okozta terhet, hogy minél inkább a kreativitás és az alkotás kerülhessen a középpontba.

Elsődleges szempont volt számomra az egyszerű, letisztult felhasználóbarát felület kialakítása. Funkcionalitás szempontjából három csoportot hoztam létre ügyfél, cukrász, vagy dolgozó és vezető, és mindhárom kategória tagjai a számukra releváns információt érhetik el, az adatok védelme és az átláthatóság megőrzése érdekében.

A dolgozatban végig követhető mind a tervezés és a megvalósítás folyamata. Igyekeztem nagy hangsúlyt fektetni a megfelelő technológiák kiválasztására, figyelembe véve a kompatibilitásukat, a területhez kapcsolódó szabályozásokat, ipari standardeket, tervezési mintákat, valamint a jó gyakoltatokat. Emellett a minta adathalmazok kialakítása, a funkciók és a felhasználói felület megtervezése is fontos mérföldkő volt. Az alkalmazás tesztelésével és értékelésével, összefoglaló képet alkothatunk az elkészült alkalmazásról.

Saját tapasztalataimat és ismeretimet ötvöztem, hogy olyasmit alkothassak, ami tényleg az emberek segítségére lehet, ezen túl pedig figyelembe vettem, hogy miben lehetne tovább fejleszteni, vagy jobban megoldani a jövőben.

ABSTRACT

The purpose of my thesis is to create web application, that can help to make more effective and easier the daily work of confectionery workshops. The application provides an opportunity to handle everything at one platform from business processes, through handling orders, get information about recipes, inventory and customer data, to the webpage available for clients. I would like to take the burden off the confectionaries that processes from the background mean, so they can focus on creativity and creation.

For me the most important aspect was the create a simple, clear-out, user-friendly platform. In the view of functionality, I made three groups, client, confectionary or employee and manager, the members of all three categories can reach the relevant information for them, to protect data and transparency.

In my paper we can follow the process of planning and implementing. I tried to highly emphasis on choosing the right technologies, taking into their compatibility, regulations, industry standards, design patterns, and best practices related to the field. As well as creating test dataset, functions and the platform planning was also a main point of the project. By testing and evaluation of the application we got a summary image of it.

I combined my experiences and knowledge to create something that can be useful for people and beyond that, I took account what could be further development or solved better in the future.

TARTALOMJEGYZÉK

1 Bevezetés 3
2 Irodalomkutatás 4
2.1 Piackutatás 4
2.2 Adatok forrása 4
2.3 Eszközök 5
2.3.1 Hardver fejlesztői környezet 5
2.3.2 Szoftver fejlesztői környezet 6
2.3.3 Adatbázis környezet 7
2.3.4 Backend 10
2.3.5 Frontend 12
3 Tervezés 14
3.1 Forrás rendszer bemutatása 14
3.2 Architektúra 16
3.3 Use case 17
3.4 Weblap tervezet 18
4 Fejlesztés 20
4.1 Adatbázis 20
4.2 Fejlesztő környezet kialakítása 21
4.3 Backend és frontend fejlesztése 22
4.3.1 Modell és adatbázis kapcsolat 22
4.3.2 Frontend és ügyfél oldalak 23
4.3.3 Dolgozói oldalak és CRUD metódusok 24
4.3.4 Adatvizualizáció a vezetőségi oldalakon 25
4.3.5 Regisztráció, autentikáció 26
4.3.6 Fejléc 26
4.3.7 Frontend és backend kommunikációja 27
5 Biztonság 28
5.1 Biztonsági elvek 28
5.1.1 CIA Triad 28
5.1.2 OWASP Top 10 29

5.1.3 GDPR és NIS2 szabályozások 29
5.2 Biztonság a gyakozlatban 30
5.2.1 Környezeti változók 30
5.2.2 Bcrypt 30
5.2.3 JSON Web token 30
5.2.4 Adatvalidáció 31
5.2.5 Jogosultság kezelés 31
5.2.6 CRUD műveletek biztonságos megvalósítása 31
6 Tesztelés 32
7 Felhasználói kézikönyv 36
8 Továbbfejlesztés 39
8.1 Tesztelési javaslatok megvalósitása 39
8.2 Bankkártyás fizetési lehetőség 39
8.3 Mobil applikáció 40
9 Összegzés 41
10 Summary 42
Irodalomjegyzék 43
Ábrajegyzék 46
Táblajegyzék 47
Mellékletek 48
1. melléklet 48
2. melléklet 50
3. melléklet 51

1 BEVEZETÉS

Életünk során számos olyan alkalom van, amikor az édességek, desszertek fontos szerepet kapnak, melyek végig kísérnek életünk fontos eseményein, legyen szó születésnapról, esküvőről, vagy egy egyszerű baráti délutáni csevegésről. A cukrászat nem csak egy mesterség, hanem egy művészeti ág is, ahol az ízek, színek és formák találkoznak, egy-egy különleges élményt nyújtva.

Hobbi cukrászként sokszor megkapom a kérdést mért nem ezt választottam szakmának és ezáltal én magam is elgondolkodok a kérdésen, illetve elképzelem milyen lenne, ha így tettem volna. Elmerengek milyen lenne édességeket készíteni a saját kis cukrászdámban, ahogy a vendégek a kirakat termékeit válogatnák, a teraszon majszolnák a gyerekek a süteményeket és a pultnál megrendeléseket adnának le egy-egy szülinapi, vagy esküvői tortára.

Ez az idilli kép eszembe juttatta, hogy nem lenne ennyire rózsaszín és tejszínhabos a helyzet, hiszen rengeteg üzleti folyamat történik a háttérben, amit a legtöbbször nem is látunk. Komoly kihívást jelenthet az alapanyagok beszerzése és nyomon követése, a rendelések rendszerezése, vagy épp az adatok tárolása. Számomra mindig kihívást jelentett a receptek megőrzése, melyiket melyik könyvben találom, vagy épp az interneten találtak esetében, kinyomtattam-e, ha igen hova tettem, kidobtam-e, több verziónál, melyik volt az igazán jó.

Ezeket végig gondolva és saját tapasztalataim által választottam dolgozatom témájának és célomul egy olyan webes alkalmazás megvalósítását, mely képes receptek tárolására, ugyanakkor a készlet információk, a rendelések és a vevői adatok is nyomon követhetőek segítségével. Szeretnék egy olyan alkalmazás létrehozni, mely egyszerűbbé és hatékonyabbá teszi a cukrászok mindennapjait, ami rámutat az optimális irányra. Előnyeinek szánom a könnyű és gyors adatélérést, mely gördülékenységet hoz a mindennapokba emellett a kényelmet és a rugalmasságot, hogy bárhol és bármikor elérhető a rendszer.

Úgy gondolom egy ilyen alkalmazás hasznos lehetne a cukrászok számára hiszen minden megtalálható lenne egy helyen, ráadásul kényelmesen és bármikor elérhető lenne. A létrehozása személyes érdekem is az említett recept tárolási probléma miatt.

2 IRODALOMKUTATÁS

2.1 Piackutatás

A projekt legelején úgy gondoltam, hogy mivel konkrétan a cukrászokra vonatkozóan terveztem mindent, így nem, vagy alig fogok más olyan alkalmazást találni, mint ami elképzeléseimben szerepelt, azonban tévedtem. Úgy gondoltam az alkalmazás erejét az fogja jelenteni, hogy ötvözi a receptgyűjtő oldalak és az üzleti folyamatokat kezelő rendszerek tulajdonságait.

A BakeSmart, a SweetShop Manager, a CakeBoss és a Bakery Managent System is már létező rendszerek, melyek a sütőipari vállalkozók munkáját segítik üzleti tevékenységeikben, ügyfél kapcsolattartással, üzleti elemzéssel receptek nyilvántartásával, automatikus készlet nyomon követessel. Bár mindegyik egy célt szolgál kicsit mindegyik másra fókuszál. Az első kettő viszonylag általánosabb e-kereskedelmi platform, míg a CakeBoss inkább a tortakészítők igényeire koncentrál, a Bakery Managent System, pedig az élelmiszer ipari vállalkozásokra. A BakeSmart és a Bakery Managent System több funkciót kínál, azonban ennek okán több tanulást is igényel a használatuk, emellett rendkívül erőteljes analitikai megoldásokat is kínál. A másik két alkalmazás egyszerűbb és rendkívül felhasználóbarát megoldást nyújt. A BakeSmart és a CakeBoss felhő alapú szolgáltatások, míg a SweetShop Manager és Bakery Managent System letölthető alkalmazások. [1] [2] [3]

Ennek ismeretében tűnődtem el azon, hogy hogyan lehetne még javítani ezeken az alkalmazásokon. Az automatizáció irányába kezdtem el gondolkodni, ezért szeretnék készíteni egy olyan komponenst az alkalmazásba, ami automatikusan generál alapanyagigénylést a készlet fogyásának figyelembevételével. Ezen felül az ügyfelek számára elérhető felületet is szeretnék létrehozni, ahol informálódhatnak a cukrászdáról és rendeléseket is le tudnak adni előre.

2.2 Adatok forrása

Fiktív cukrászda lévén a valós adatok nem állnak rendelkezésemre, így ennek pótlására kerestem megoldástokat.

Elsőnek adatgenerátor használatán gondolkoztam el, mint például a generatedata.com. [4] Az online teszt adatgenerátorok felhasználóbarát felülettel, sokféle adatszerkezetet kínálnak. Esetenként kis mértékben szerkeszthető vagy korlátozható, módosítható, hogy projektünkhöz megfelelő adatszerkezetet hozhassunk létre, akár különböző fájl típusokban is van lehetőségünk exportálni őket. Sajnos a legtöbb esetben nem képesek nagy mennyiségben adatott generálni, emellett, formátumban, vagy tartalmát tekintve sem feleltek meg az elvárásaimnak és mivel az alkalmazás alapját az adatbázis adja így az adatmodell kiválasztása kulcsfontosságú lépés.

Az említett problémák kiküszöbölését saját programkódok írásával szerettem volna megoldani az adatok generálását, hiszen, ha már van egy alap adatgyűjtemény, akkor már viszonylag könnyen formázható az adatmodell kialakítási paradigmáknak megefelelően (pl: 3NF). A következő kihívást a receptek összegyűjtése és adatbázisba tárolható formára hozása jelentette. Bár ez programkódból való generáláskor nem feltétlenül jelentene problémát, hiszen egyedi adatéma kialakítására ad legetőséget. Azonban van olyan adattábla (pl: Recipe) ami igényli a valóságnak megefelelő adatokat ahhoz hogy használható és értelmezhető legyen, így az Interneten is kerestem minta adatbázisokat.

Rengeteg publikus adathalmaz érhető el az online térben, ezek formázott formában tölthetőek le legtöbbször CSV, vagy JSON formátumban, ez által fenntartva a lehetőséget, hogy a félstruktúrált adatokat több különböző adatbázis kezelő rendszer is fogadni tudja, cserébe viszonylag kötöttek tartalmukat tekintve. Legnépszerűbb opciók a Kaggle [5], a Google Dataset Search [6] és a GitHub [7]. Én ezutóbbin találtam meg a számomra legmegfelelőbb adathalmazokat, amiket egyesítve és program kóddal formázva hoztam létre a sajátomat.

A gondolatmenetnek köszönhetően az egyes táblák adatai különböző forrásokból származnak. A customer tábla adatait és az order tábla Order_Date oszlopát a mackaroo online generátorral készítettem, ezutóbbi többi oszlopát saját kód segítségével állítottam elő. [8] A Recipe táblát pedig két GitHub repository adathalmazát ötvözve, az ingredients táblát adatait pedig tisztán saját kóddal alkottam meg biztosítva, hogy egy, a jelen felhasználásnak megfelelő adatszerkezetet kapjak a folyamat végére. [9] [10].

2.3 Eszközök

2.3.1 Hardver fejlesztői környezet

A projekthez szükséges mintarendszert a laptopomon fogom megvalósítani, ami egy 4 éves Lenovo Ideapad 5. (2-1. ábra) A gép egy 10. generációs Intel Core i5-ös processzorral, 8 GB memóriával és 512 GB háttértárral rendelkezik. Az elmúlt években ezek a paraméterek elegendőnek bizonyultak a tanulmányaim folyamán készített projektekhez, így bízom benne, hogy a jelenlegihez is megfelelően stabil hátteret tudnak majd biztosítani.

Laptop lévén könnyen magammal vihetem bárhova, így bármikor bárhol dolgozhatok a projekten, akár az iskolában, a munkahelyen, vagy otthon, mivel sokszor ingázok ez egy fontos szempont számomra, hogy megfelelően ki tudjam használni a rendelkezésre álló időmet. Ezen okból emelném ki a 15,6’’-os kijelzőt, könnyű súlyát és habár az évek során természetesen amortizálódott, még mindig igen hosszú akkumulátor-élettartamát, mely kényelmessé teszi a munkavégzést még útközben is. Valamint a tesztelés során kifejezetten fontosnak bizonyult a hordozhatóság,

Jelenleg Windows 10-es operációsrendszer fut rajta. Nem terveztem virtuális környezetet kialakítani, így mindent közvetlenül a gépre fogok telepíteni, ennek okán fontos szempont lesz a megfelelő technológiák kiválasztásakor, hogy azok Windows kompatibilisek legyenek.



2-1. ábra Lenovo Ideapad 5

2.3.2 Szoftver fejlesztői környezet

A webes alkalmazás munkafolyamatainak elkezdéséhez, szükség lesz egy fejlesztő környezetre. Ezek a rendszerek lehetővé teszik a hatékony és kényelmes munkavégzést az alkalmazások fejlesztésekor. A lehetőségeket vizsgálva több opciót is számba vehetünk.

Elsőnek a Microsoft Visual Studioja, merült fel, ami egy integrált fejlesztő környezet és több programozási nyelvet is támogat, többek között a C#-ot, JavaScriptet és a Pythont is. Ez a sokoldalú kódfejlesztő informatív hibakeresővel és fordítóval is rendelkezik, valamint integrált tesztek futtatására és asztali, mobil, webes és felhő alapú alkalmazások fejlesztésére is alkalmas. Könnyen kapcsolható GitHub-hoz így a verziókövetés és az osztott munka is kivitelezhetővé válik. [11]

A Visual Studio Code (2-2. ábra) szintén a Microsoft kódszerkesztő terméke, mondhatni a Visual Studio egyszerűsített nyílt forráskódú verziója, de még így is rengeteg kiegészítővel és bővítménnyel rendelkezik. Webes és felhő alapú alkalmazások fejlesztéséhez kiváló választás lehet, vagy ha macOS-en, esetleg Linuxon szeretnénk használni, mivel nem csak Windows-zal kompatibilis, így a jövőben is platformfüggetlenül folyhat a projekt továbbfejlesztése. A kódkiegészítő, hibakereső és verziókövető funkciók, ugyan úgy elérhetőek, mint nagytestvérénél. Egyszerű, letisztult, de mégis felhasználóbarát felületet kínál. [12] [13]



2-2. ábra Visual Studio Code logo

Egy másik jó opció lehet az InteliJ IDEA fejlesztőkörnyezet, mely első sorban a Java-t támogatja, de emellett Kotlinnal és Scalaval is működik. Hibakeresés és automatikus kódkiegészítés ebben az esetben is elérhető és integrált rendszerekkel, tesztelő eszközökkel is rendelkezik. Támogatja a projektmanagementet, verziókövetést és szoftverkonfigurációt. Elérhető ingyenes, nyílt forráskódú verzió is, természetesen valamivel limitáltabb eszközkészlettel. [14]

Az InteliJ IDEA-hoz hasonlóan a PyCharm is JetBrains IDE termék, azonban ez kifejezetten a Python programozási nyelvhez lett fejlesztve. Erős kódszerkesztő és hibakereső, támogatja a virtuális környezeteket, fájlszerkezeteket és a csomagkezelést. Automatikus kódkiegészítéssel és kód sablonokkal segíti a fejlesztő munkáját. Támogatja a Django és Flash keretrendszereket így kiváló választás webes alkalmazások készítéséhez. Integrált verziókezelés is elérhető ezzel elősegítve a csapatmunkát. Az intelligens refaktorizálóval egyszerűen átrendezhetjük, optimalizálhatjuk kódunkat az alkalmazás veszélyeztetése nélkül. [15]

Személyes tapasztalatom eddig az első hárommal volt és közülük is inkább a Microsoft termékekkel. Mivel ez a projekt sok új kihívást fog jelenteni elképzeléseim szerint így ebben a témában a számomra ismertebb eszközök közül választanék. Webes applikáció lévén, véleményem szerint bőven kielégíti igényeimet a Visual Studio Code elsődlegesen az alcsony rendszerigénye illetve webalkalmazás fejlesztésre való optimalizáltsága miatt, erre esik választásom.

2.3.3 Adatbázis környezet

Mivel a weboldal mögött egy adatbázis lesz, ami az adatokat tárolja így ennek kiválasztása is kulcskérdés a projekt szempontjából. Bár az adatbázis motor kiválasztását a legtöbb esetben az adatok jellege, struktúráltsága adja, itt hasonlóan fontos szempont volt a skálázhatóság is. Azonban itt nem csak az eszközt, de a technológiát is érdemes végig gondolni.

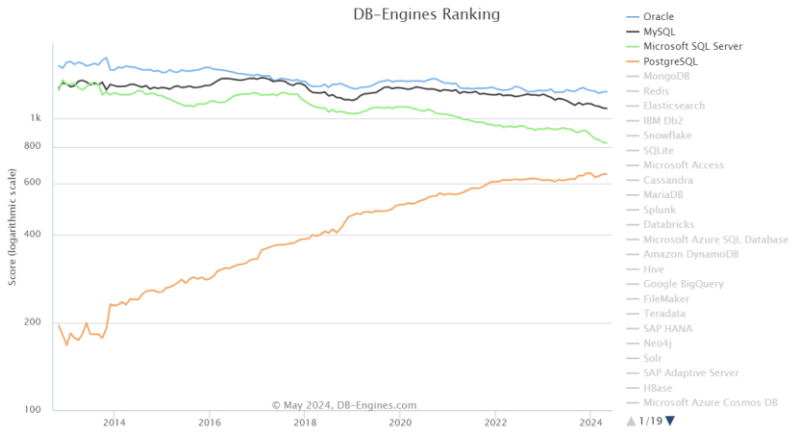
A relációs adatbázisok erősen strukturált formában, táblákban tárolják az adatokat, amihez a hozzá fejlesztett SQL nyelv segítségével érhetünk el és manipulálhatunk. A táblák közötti kapcsolatot az elsődleges és idegen kulcsok határozzák meg. Rendelkezik az úgynevezett ACID tulajdonságokkal, azaz autonóm (A), a műveleteket egy oszthatatlan egységként kezel, konzisztens (C), a tranzakció meg kell, hogy feleljen a megkötéseknek, izoláció (I), biztosítja, hogy egyidejűleg ne mehessen végbe több tranzakció és tartósság (D), az adatok egy rendszerhiba esetén is fennmaradjanak. Legismertebb adatbáziskezelők az Oracle, a MySQL, a PostgreSQL és a Microsoft SQL Server.

A NoSQL adatbázisokat a BigData terjedése és a skálázhatóság iránti megnövekedett igény miatt alkották meg. Szakít az SQL adatbázisok merev strukturált szerkezetével, ezzel biztosítva rugalmasságot, horizontális skálázhatóságot, nagy adatmennyiség gyors

feldolgozását, megtartva a megbízhatóságát. A nem relációs adatbázisok dinamikus sémát használnak, azaz nem táblákban tárolják az adatokat. Lehetőséget ad akár különböző rekordok tárolására. Leggyakoribb típusai a dokumentum alapú, mely komplex hierarchikus adatokat is képes tárolni, a kulcs-érték alapú, az egyik legegyszerűbb és leggyorsabb válasz idejű, a gráf alapú, ahol az adatok kapcsolata látható és az oszlop alapú, ami a sorok helyett oszlopokban tárolja az adatokat a magas szintű skálázhatóság érdekében. Strukturálatlan, vagy félig strukturált adatokat is tudnak értelmezni, mint például a JSON, vagy az XML. Legnépszerűbb képviselői a MongoDB, a Redis, a Neo4j és az HBase. [16]

A receptek tárolása miatt elgondolkodtam a dokumentum alapú NoSQL használatán, mely a különböző komplexitású és terjedelmű receptek kezelésére lenne alkalmas. Ugyanis az ilyen adatokat célszerű egy struktúraként eltárolni ez által csökkentve a tábla összekapcsolásokat, meylek költséges műveletek. Azonban a többi táblával a kapcsolat kialakítása egyszerűbb és hatékonyabb a relációs adatbáziskezelő használatával, hiszen az SQL elterjedtebb lekérdező nyelv mely megkönnyiti a fejlesztést az által, hogy az egyes szintaktikai változatai kevésbé térnek el mint a NoSQL lekérdező „nyelvjárások” esetében, valamint a lekérdezések írását elősegíti a táblák összekacsolása és a jellemzően fejledt lekérdezés optimalizáló motor. A NoSQL adatbázisok esetén az alapvető adattervezési filozófia más, így a táblák közötti kapcsolatkialakítás legtöbbször igen korlátozott, hiszen az elosztott architectúrát használnak és a demormalizált az adatszerekezetek használata miatt problémás, így ezek alapján a relációs technológiát választom.

Relációs adatbázisok közül a már említetteket vizsgáltam részletesebben. Mindegyik az első négy hely valamelyikét foglalja el a DB-Engines szerint az adatbázisok rangsorában, mely rangsor az adatbázis motorok elterjedtsége alapján jött létre. És bár a motorok technikai paramétereit nem vették közvetlen számításba a piaci eloszlás remek indikátorként funkcionál jelen esetben. (2-3. ábra) [17]



2-3. ábra DB-Engines Ranking

Az Oracle relációs adatbázis kezelő rendszer, ami széleskörben elterjedt, az első ilyen jellegű eszközök egyike. Az évek és verzió frissítések során így vált rendkívül stabil és biztonságos, jól dokumentált. Gyors adatelérést és feldolgozást tesz lehetővé. Szigorúan strukturált séma alapú, nagy a hibatűrő képesége. Számos funkcióval rendelkezik, mint például a jogosultság kezelés, adatmentés és helyreállítás melyek fontos szerepet játszanak a hiba megelőzésében, valamint az elhárításában. Komplex lekérdezéseket is jól kezel, ezzel segítve az adatelemzést és jelentéskészítést, támogatja PL/Sql-t, mely segítségével tárolt eljárások futtatására is van lehetőség az időzített feladatok zökkenőmentes elvégzése valamint a protokollok gyors lefutása érdekében. Az újabb verzióin a felhő használat is kivitelezhető, amitől skálázhatóvá válik a rendszer és lehetővé teszi az adataink fizikai elkülönítését a munkavégzés helyétől csökkentve az esélyét annak hogy katasztrófa esetén minden elvesszen. Bár egy nagy múltú szoftverről van szó mely elavult technológián és gondolgkodásmódon alapult, ennek ellenére a mai napig implementálva vannak a legújabb technikai megoldások, ez által biztosítva a hosszútávú megbízhatóságot és a naprakész technikai hátteret. Általában nagyvállalati környezetben használják és a szükséges licensz magas költségekkel jár. Pénzügyi és állami szektorokban elterjedt.

Egy másik népszerű megoldás lehet a MySQL, ami szinte minden operációs rendszeren jól működik. Eredetileg egy nyílt forrású eszköz volt, de mostanra az Oracle-höz tartozik, ettől függetlenül ingyenes verzió is elérhető belőle. Könnyen és egyszerűen telepíthető és használható, letisztult struktúra és stílus jellemzia felhasználói élményt javítva és a munkafolyamatokat könnyítve ezzel. Felhő kompatibilis, azonban nincs beépített eszköz skálázhatóságra, ami komoly kihívást jelenthet, hiszen előre meg kell határozni a rendszer teljesítmény igényét amit ha alábecsül a tervező akkor túlterheltséghez vezet, ellenkező estben erőforrást pazarol a rendszer. Ezek miatt első sorban webes alkalmazásokhoz és kisebb vállalati környezetekben használják.

A PostgreSQL egy ingyenes és nyílt forráskódú objektum-relációs adatbáziskezelő. Változatosabb adat típusokkal rendelkezik (pl. tömbök) és földrajzi adatok kezelésére is képes melyek lehetőséget biztosítanak a pontosabb adat reprezentálásra, például lehetőség van akár poligonk, utak és tájak pontos és részletes eltárolására a koordinátákon felül. És bár a projekt szempontjából ezek a extra funkciók nem fontosak jelenleg fenntartják a lehetőséget egy több lokációra kiterjedő rendszer fejlesztésére. Ezen felül nem relációs lekérdezések támogatására is lehetőséget nyújt. Talán legnagyobb előnye a szinte mindenre kiterjedő funkciókészelte, mely magába foglalja a tranzakció- és jogosultság kezelést, indexelést, de még szerver oldali funkciókat is. Jól használható nagy mennyiségű és komplex adathalmaz feldolgozására is. Több mint 30 év alatt elég stabil és kiforrott lett a rendszer, köszönhető nagy támogatói közösségének, azonban a dokumentációja elmaradott. Leginkább közép és kis vállalkozások használják.

Az MSSQL, azaz a Microsoft SQL Server egy kereskedelmi adatbázis kezelő rendszer. Microsoft termék lévén, jól integrálható a cég más termékeivel. Ennek köszönhetően

könnyen összekapcsolható az Azure felhőszolgáltató platformmal, vagy a Power BI üzleti intelligencia eszközével. Hatékony eszköze az adattárolásnak, kezelésének, manipulálásának és indexelésének, emellett lekérdezés és egyéb optimalizálási lehetőséget is kínál. Az adatbiztonság és integritás érdekében jogosultság kezelési, átviteli titkosítási, auditálási, adat mentési és helyreállítási funkciók is rendelkezésre állnak. Képes a T-SQL használatára, ami az SQL nyelv egy kibővített változata és jól dokumentált, azonban igen magas költségekkel jár a licensz beszerzése. [18]

Tanulmányaim és munkám során mindegyik adatbáziskezelőt megismerhettem kisebb-nagyobb mértékben. Véleményem szerint az adatszerkezet és a felhasználási cél ismeretében a PostgreSQL megfelelő választásnak tűnik, mivel képes kevésbé strukturált adatokat is kezelni, mint például egy JSON fájl, melyek az architektúra rétegei közötti kommunikáció lebonyolítására használt fájlok, így igény szerint az összes tranzakció könnyen naplózhatóvá alakítható.

2.3.4 Backend

A backend (szerver oldal) a felhasználó számára láthatatlan része az alkalmazásnak, azonban mégis kulcsfontosságú. Ez valósítja meg az adatkezelést és feldolgozást, valamint a logikát, kéréseket dolgoz fel és kommunikál a frontenddel. Az alkalmazás backendjének esetében három keretrendszert vizsgáltam meg, mindhárom nyílt forrású és nagy támogatói közösségnek örvend, mivel ezek lehetővé teszik, hogy a felhasználók fejlesszék a keretrendszert, valamint megtalálják és kijvaítsák az esetleges hibákat. Fontos szempont volt, hogy olyan rendszereket keresek, melyek támogatják a már kiválasztott PostgeSQL-t, mivel az adat struktúra a legkötöttebb pontja a projektnek, azon a legnehezebb változtatni, így az azzal való kompatibilitás volt az elsődleges szempont a további rétegek kivásztásánál.

Elsőnek a Django, ami egy Python alapú keretrendszer, könnyen tanulható és gyors fejlesztést tesz lehetővé egyszerű szintaktikájának, a fejlesztést könnyítő beépített funkciók és az elterjedtségéből adódóan a közösségi segítségnyújtásnak segítségével. A rengeteg beépített tulajdonságnak köszönhetően, melynek része, hogy nagy adatmennyiséget is tudunk kezelni, valamint a kódunk hossza is csökkenthető, elősegítve az átláthatóságot. Alapértelmezetten rendelkezik biztonsági rendszerekkel, mint a web applikációs protokollok valamint az adattárolási titkosítás és a naplózás, azonban nem képes több kérést egyszerre kezelni, ezzel lassítva a futást. MVT architektúrát követ, ami a Model, ami az app üzleti logikáját tartalmazza, a View, ami a felhasználói interfészeket kezeli és a Template ahol a statikus HTML és dinamikus elemeket definiáljuk. Objektum-reláció leképzést tesz lehetővé így csökkentve az SQL kód írásának szükségességét. Ez a fejlesztés gyorsításával jár hiszen a specifikus SQL kód helyett python kódból tudunk lekérdezést indítani. Másik előnye hogy a ORM technologia alkalmazásával az adatbázis motor cserélhető ez által nyitva hagyva a lehetőséget egyéb adatbázis kezelő rendszerek használatára valamint összességében jövőbiztos és naprakész lesz a rendszer.

A Node.js (2-4. ábra) egy JavaScript alapú esemény vezérelte keretrendszer. Mivel nem kifejezetten web keretrendszer, így lehetőséget nyújt a böngészőn kívüli kódfuttatásra ez által alkalmassá vált a szerveroldali programozásra, a mikroszervíz architectúrával és API-ok használatával való kompatibilitásra. Ezen felül valós idejú alkamlazások és parancssoros eszközök is elérhetők. Egyszerre több kérést is tud fogadni, rugalmassága, skálázhatósága és gyors megjeleníthetősége miatt az egyik legkedveltebb fejlesztői program. Támogatja a könyvtárrendszert és sok kiegészítőmodullal rendelkezik, melyek autentikációra, naplózásra és routingra is használhatunk, ezzel lefedve az alapvető internetes protokollok működését, valamint a biztonsági elvárások ide vonatkozó jelentős részét. Közvetlen hozzáférést biztosít http kérésekhez és válaszok kezeléséhez biztosítva a rétegek közti megefelelő kommunikáció. Népszerűségének megfelelően nagy támogatói közösséggel rendelkezik, viszont kevesebb beépített funkció érhető el, mint a Django esetében. [19] [20]



2-4. ábra Node.js logó

Ruby on Rails egy Ruby nyelvet használó keretrendszer, ami első sorban a kódírásra fordított idő lerövidítésére koncentrál. Megköveteli az újra használható kódrészeket a „Ne ismételd magad” elv szerint ez segít a rejtett függőségek és túlzottan komplex kódok kiküszöbölésében is. Gyorsan tanulható és jól olvasható. Rendkívül dinamikus, azonban pont ez okozza esetleges gyengeségét mivel csökkentheti a teljesítményét. Djangohoz hasonlóan ez is rendelkezik beépített eszközökkel adatbáziskezeléshez és képes az Object-Relational Mappingre, mely elősegíti az adatbázis és a program összekapcsolását, mely a Django-hoz hasonlóan nyitva hagyja a lehetőséget az adatbázis motor cserélhetőségére. [21]

Mindhárom lehetőségnek megvannak az előnyei és hátrányai, nagyon vonzó, azonban igazán mélyrehatóan egyiket sem ismerem, minimális mértékben ismerem a Python és JavaScript nyelveket. Emellett ismeretségi körömben körbe kérdeztem olyanokat, akiknek van már tapasztalata ezekkel a keretrendeszerekkel és az ő javaslatukra a Node.js-re esett a választásom, melynek számos előnyét ki tudom majd használni.

2.3.5 Frontend

A frontend kezeli a felhasználó számára látható elemeket, például szövegdobozokat, képeket, gombokat, a felhasználói felületet és élményt. Ennek választásánál szintén fontos szempont volt, hogy a már kiválasztott komponensekkel jó együtt tudjon működni, ne okozzon a fejlesztés során kihívást a különböző elemek összeillesztése, így a következőben vizsgált eszközök mind kompatibilisek a Djangoval.

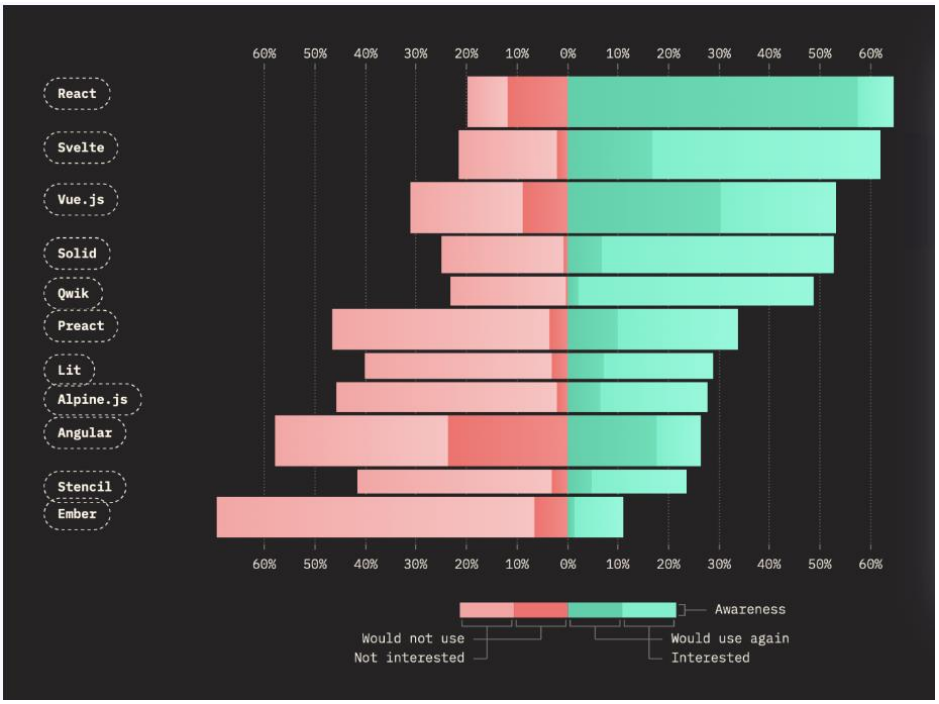
A React.js egy deklaratív JavaSript könyvtár, amelyt kifejezetten dinamikus oldalak fejlesztéséhez készítettek. Támogatja a jól elkülönülő újra felhasználható komponenseket, ami segíti az átláthatóságot és kiszűri a felesleges kódismétléseket. Több külső funkció könyvtár is elérhető, hogy segítsék a fejlesztők feladatát, amiket rendszeresen fejlesztenek is. Az alkalmazás állapotát egy virtuális document object modelben tárolja, hogy javítsa az alkalmazás reakció idejét, a nem lényeges változások így nem történek meg a valós modellben, kizárólag szerveroldalon leszenek renderelve. Amint szükséges azonban könnyen megjeleníthetőek. [22]

Másik lehetőségnek az Angulart választottam, egy nyílt forráskódú JavaSript alapú kertrendszer, a Google terméke. Teljeskörű megoldást kínál a tervezéstől a fejlesztésen és tesztelésen át, az üzembehelyezésig. A Reack.js-hez hasonlóan ez is újra használható komponensek használatát támogatja. Beépített reaktív eszközök (pl observerek) segítségével követhetjük nyomon a program állapotát. Jól dokumentált és közösségi támogatása is jelentős, használata azonban mélyebb ismereteket igényel. [23]

A korábbiakhoz hasonlóan a Vue.js is egy JavaScript keretrendszer, ami egyszerű, letisztult könnyen tanulható, mégis hatékony. Itt is automatikusan történik az alkalmazás állapotának kezelése és komponens alapú, így összetettebb UI felületek is egyszerűen összerakhatóak. Egyszerre képes HTML-t JavaScriptet és CSS-t kezelni, amik segítségével teljes mértékben testreszabható a weboldal felépítése, szerkezete és kinézete, stílusa. Emelet a kiegészítő könyvtárak segítségével kedvünk szerint testre szabhatjuk alkalmazásainkat. [24]

A Django tartalmaz beépített sablonokat, amik összekapcsolják a HTML-t és a Pythont. Könnyen és egyszerűen tanulható és használható, rugalmas mivel Python kód írható a HTML-be és újra felhasználható. A HTML strukturálja és formázza a weboldalakat. Azonban a sablonok használata nem teszi lehetővé a komolyabb frontend elemek megvalósítását, kihívás lehet a bonyolultabb adatkezelés is, a natív debug hiánya is problémát okozhat, valamint a többi opcióhoz képest kevésbé moduláris és nehezebb testreszabni. További problémát jelent a limitált logikai elemek elérhetősége, mivel egy „logika mentes” sablonra törekedtek a készítők, így az elágazások és ciklusok elérhetőek, azonban a bonyolultabb logikai elemek nem. [25]

Ez alapján a Django sablonok használata aggályokat támaszt az adatkezelés kapcsán, így ezt kizárom az opciók közül. A JavaScript alapúak közül igazán egyiket sem ismerem, így az Angulárt is háttérbe sorolom a tudásigényessége miatt. Számomra a React.js a legszimpatikusabb és jól integrálható Node.js-szel, valamint népszerűsége is mellette szól, hiszen számos beépülő elem elérhető valamint a közösségi és professzionális segítségnyújtásra is nagy hangsúlyt fektetnek. (2-5. ábra) [26]

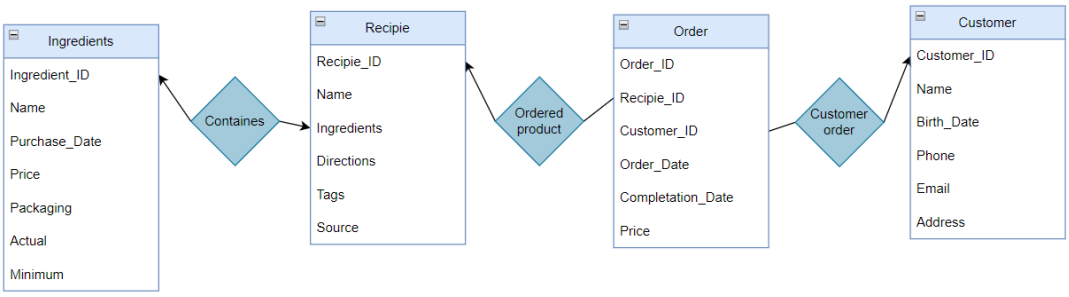


2-5. ábra Frontend rangsor a simform.com szerint 2022-ből

3 TERVEZÉS

3.1 Forrás rendszer bemutatása

Az adatbázis négy adattáblával rendelkezik, amit a következő egyed-kapcsolat diagram (3-1. ábra) mutat be.



3-1. ábra EK diagram

Kis alkalmazás révén igyekeztem redukáálni a tárolt adatok mennyiségét és minél egyszerűbb adatszerekezetet létrehozni, ez által csökkentve az adatbázis helyigényét, a komplexebb lekérdezések elkerülését, mely a futásidő javulását eredményezi. Valamint a hibafaktorok szűrését. Bár az adatsémákat utólag nehezebb módosítani, fontosnak tartom fenntartani a lehetőséget a bővíthetőségre, ugyanis gyakori esemény egy alkalmazás életciklusában, hogy az üzleti szempontok változásának következményeként egyéb adatok eltárolása is szükségessé válik. Jelen állás szerint ezeket az adattáblákat határoztam meg melyekre feltétlenül szükség van az alkalmazás működéséhez és a projekt jelenlegi céljainak eléréséhez.

Az Ingredient tábla összegyűjti a Recipe tábla alapanyag megnevezéseit és mértékegységeit és a beszerzés dátumával, valamint a termék árával egészül ki, amit a következő kód (1. melléklet) segítségével értem el. (3-2. ábra)

```
{
  "Name": "all-purpose flour",
  "PurchaseDate": "2024-03-05T00:00:00",
  "Price": 262,
  "Packaging": "cups",
  "Actual": 129,
  "Minimum": 16
},
{
  "Name": "chopped pecans",
  "PurchaseDate": "2024-11-20T00:00:00",
  "Price": 824,
  "Packaging": "cup",
  "Actual": 138,
  "Minimum": 23
},
```

3-2. ábra Az Ingredient tábla forrásfájlja

A Recipe tábla kialakítása volt a legkörülményesebb mivel a legtöbb adattagot egyszerűsíteni kellett ahhoz, hogy a megfelelő attribútumokkal számolni lehessen. A Recipe tábla 437 receptet tartalmaz, azonban nem használtam fel minden mezőt a forrás fájlból (3-3. ábra), az url és a language mezőket kihagytam, illetve a title mezőt Name-re neveztem.


```
{
  "directions": [
    "Stir the orange juice and sugar together in a pan over medium heat. Add the cranberries and bring to a boil."
    " Cook and stir until the cranberries burst. "
    "Stir in the raspberries; reduce heat and simmer 10 to 15 minutes. Refrigerate or serve warm."
  ],
  "ingredients": [
    "1 cup fresh orange juice",
    "1 cup white sugar",
    "1 (12 ounce) bag fresh cranberries",
    "2 cups fresh or frozen raspberries"
  ],
  "language": "en-US",
  "source": "allrecipes.com",
  "tags": [
  ],
  "title": "Cranberry-Raspberry Dessert Sauce",
  "url": "http://allrecipes.com/recipe/130638/cranberry-raspberry-dessert-sauce/"
},
```


3-3. ábra A Recipe tábla forrás fájlja

Az Order tábla 5000 sorához az Order_Date oszlop dátumait online teszt adat generátorral [8] hoztam létre, amit saját programkóddal3-4. ábra kiegészítve hoztam létre (2. melléklet) a többi oszlop tartalmát (3-4. ábra).


```
{
  "Recipie_ID": 201,
  "Customer_ID": 330,
  "Order_Date": "2023-08-08T00:00:00",
  "Completation_Date": "2023-08-20T00:00:00",
  "Price": 4092
},
{
  "Recipie_ID": 5,
  "Customer_ID": 1538,
  "Order_Date": "2021-06-23T00:00:00",
  "Completation_Date": "2021-06-25T00:00:00",
  "Price": 21713
},
```


3-4. ábra Az Order tábla forrás fájlja

A Customer tábla adatait a már említett generátorral hoztam létre [8] itt kiválaszthattam milyen adatokat szeretnék és a dátum, a telefonszám, valamint a cím formátumát is. Egyszerre 1000 rekordot képes generálni, így 2-szeri alkalmazással értem el a tábla 2000-es rekord számát (3-5. ábra).


```
{
  "Name": "Chickie Dyerson",
  "Borth_Date": "1943-10-15",
  "Phone": "(414) 7974702",
  "Email": "cdyerson0@admin.ch",
  "Address": "150 Karstens Trail"
},
{
  "Name": "Ernestus Flori",
  "Borth_Date": "1971-08-27",
  "Phone": "(531) 7630336",
  "Email": "eflori1@latimes.com",
  "Address": "9022 Eagan Drive"
},
```


3-5. ábra A Customer tábla forrás fájlja

3.2 Architektúra

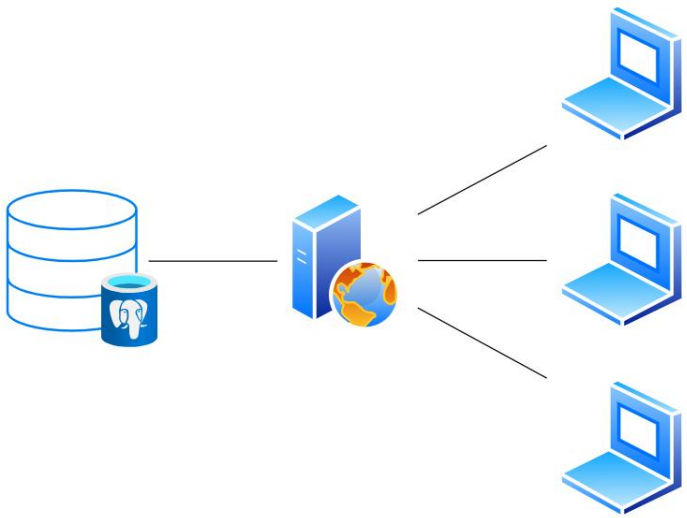
A projektet egy egyszerű, webesalkalmazásokhoz jól illő adatbázis-webszerver-kliens architektúra (3-6. ábra) alapján szeretném elkészíteni, mely elkülöníti a fontosabb komponenseket egymástól bizotsítva az egyes elemek továbbfejlesztését és cserélhetőségét, hiszen előre deklarált alkalmazásprogramozási felületen (API) keresztül kommunikálnak az egyes rétegek.

Az adatbázis tárolja és kezeli az alkalmazás által használt adatokat. Ebben a rétegben történik az adatok felvétele, manipulálása, és lekérdezése. Jelen esetben PostgreSQL adatbáziskezelő tölti be a szerepet.

A webszerver közvetítőként szolgál a kliens és az adatbázis között. Fogadja a kliens kéréseit és válaszol neki, az adatbázisból származó adatok segítségével. Ennél a projektnél a webszerver a Express.js technológiát alkalmazza.

A kliens kéréseket küld a webszervernek és a kapott válaszokat megjeleníti a felhasználó számára, tehát ez a réteg felelős a felhasználói interakciókért. A webböngésző a React.js-t használja a megjelenítés elősegítésére.

Az architektúrának köszönhetően az alkalmazás rétegei elkülönülnek, így könnyebben karbantarthatóak. Ugyanakkor, nem feledkezhetünk meg a rétegek összehangolásának foltosságáról, ami kihívást jelenthet a munkafolyamatok során hiszen különböző célt látnak el különböző megközelítéssel. Ezért is volt fontos szempont számomra a komponensek kiválasztásakor, hogy jól összehangolhatóak legyenek.

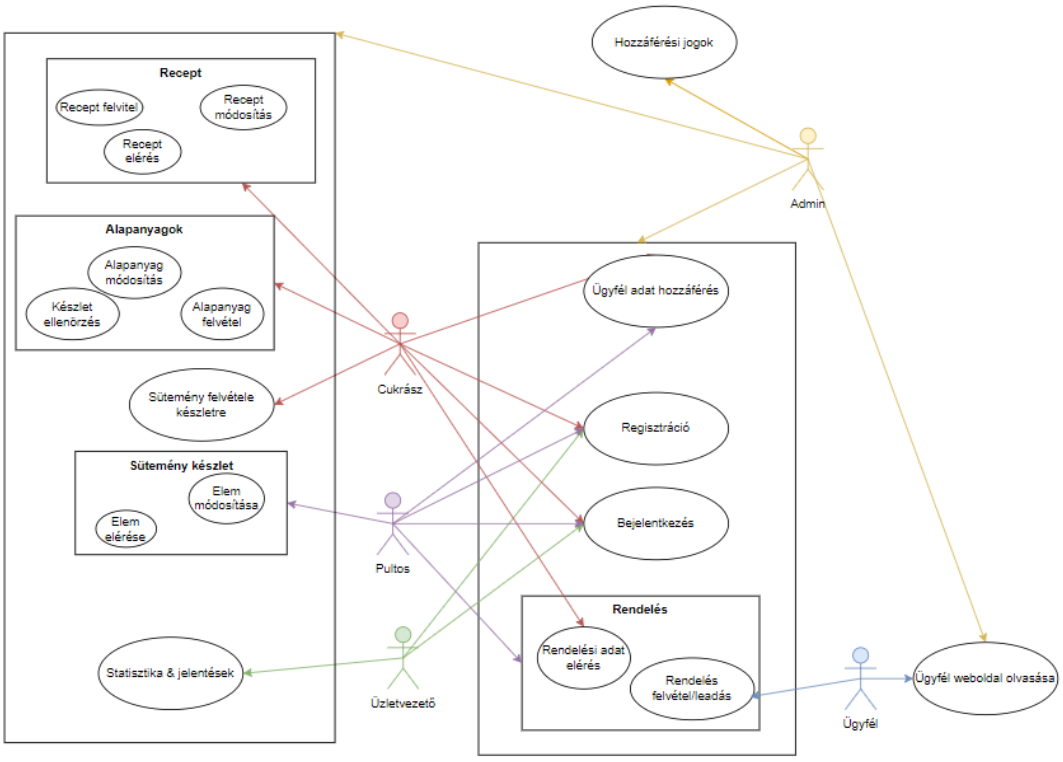


3-6. ábra A webalkalmazás architektúrája

3.3 Use case

A következő ábrán (3-7. ábra) az alkalmazás use case diagramját láthatjuk, azaz, hogy milyen szerepkörökben jelenhetnek meg az alkalmazás felhasználói, illetve, hogy az egyes szerepekhez milyen jogosultságok és hozzáférések tartoznak.

Természetesen az admin teljes, mindenre kiterjedő jogkörrel rendelkezik. A cukrász első sorban a receptek, a rendelések és az alapanyagok adatait kezelheti, hiszen számukra ezek szolgáltatnak szükséges információt a munkához. A pultos is hozzáfér a rendelésekhez, illetve a sütemény készletekhez, így jelezheti a cukrászoknak, ha hiány lép fel valamiből a cukrászda pultjában. Az üzletvezető első sorban a statisztikákat és a jelentéseket láthatja, hiszen számára teljesen irreleváns hány gramm liszt kell a tortába, viszont a megfelelő adatok igencsak befolyásolhatják az üzlettel kapcsolatos döntéseket. Az ügyfelek a nyilvános weboldalt láthatják, illetve bizonyos termékekre azon keresztül rendelést is adhatnak le.



3-7. ábra Az alkalmazás Use Case diagramja

3.4 Weblap tervezet

A következő képek a webes alkalmazás tervezet kinézetét mutatják be.

Az ügyfelek által elérhető lesz a cukrászda és a kínálat ismertetése. Mivel a sütemények automatikusan rendelhetőek lesznek, így ezt is ezen a felületen érhetik majd el, valamint a cukrászda elérhetősége is fel lesz tüntetve. (3-8. ábra)



3-8. ábra Ügyfelek által elérhető weboldal

Innen érhető el a bejelentkezési oldal (3-9. ábra), ahol az alkalmazottak és a vezetőség érhetik el a dolgozói oldalakat bejelentkezést követően.



3-9. ábra Bejelentkezési oldal

A sikeres bejelentkezést követően az alkalmazottak itt találják a recepteket, a hozzávalókat és a rendeléseket is, illetve innen módosíthatják az automatikusan generált alapanyag igényléseket is, egy-egy nagyobb megrendelés esetén, ha szükséges. (3-10. ábra)



3-10. ábra Dolgozók által elérhető weboldal

A vezetőség számára szintén egy kicsit más menüsor áll rendelkezésre, kímélve őket a felesleges információtól, a részükre releváns részekre koncentrálva. Ők a jelentéseket és a statisztikai adatokat érik el, melyek az üzleti döntéshozatalt segítik. Emellett partnereik adatait és dolgozóikat is nyomon követhetik ebből a nézetből. (3-11. ábra)



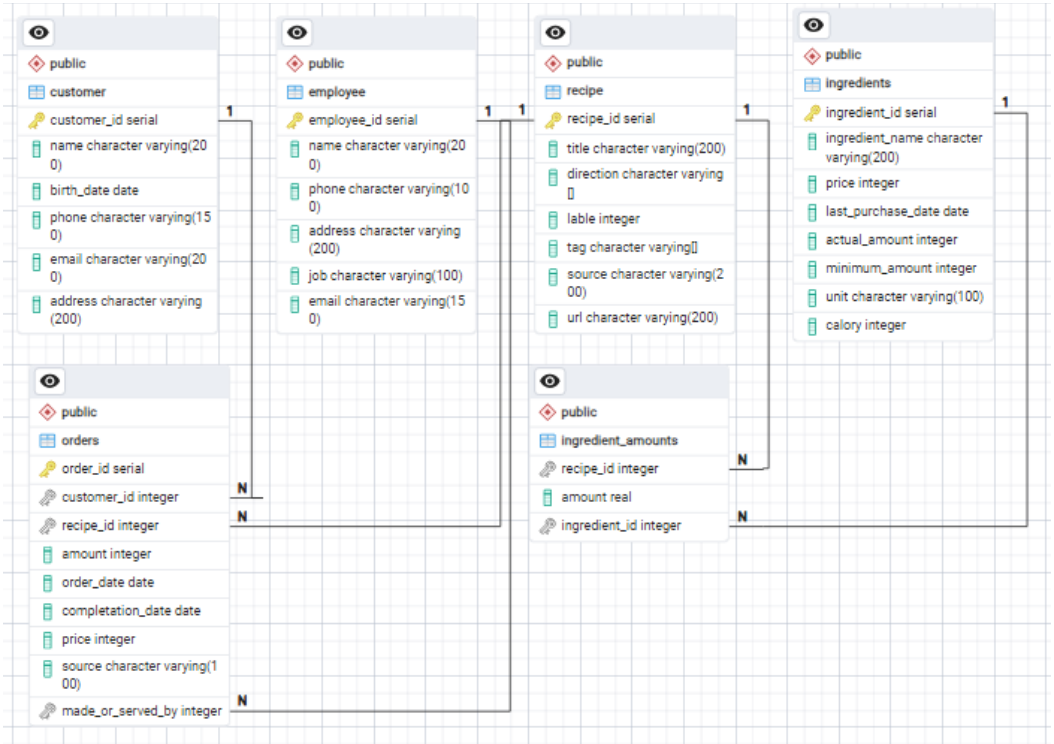
3-11. ábra Vezetőség által elérhető weboldal

4 FEJLESZTÉS

4.1 Adatbázis

A fejlesztési folyamatokat az adatbázis kialakításával kezdtem, mivel ez alapozza meg az egészet, ezt a legnehezebb változtatni a struktúra miatt. Létrehoztam egy schema-t és beállítottam a jogosultságokat, így biztosítva, hogy az adatok elérhetősége a kezdetektől el legyen különítve a szerepköröknek megfelelően. Ezután létrehoztam a táblákat és az elsődleges kulcsoknak „serial” típust választottam, így az adatbázis kezelő létrehoz egy szekvenciát a táblához és az elsődleges kulcsok automatikusan generálódnak. Az adatokból, program kód segítségével INSERT, azaz beillesztő parancsokat készítettem, ezek lefuttatásával töltöttem fel a táblákat.

Az eredeti terveim szerint a receptek hozzávalóit tömbökben tároltam volna, azonban problémát jelentett, hogy ebben a formában az alapanyag mennyiségekkel és mértékegységekkel nem lehetett volna számolni. Elgondolkodtam saját adattípus tömbként való használatán azonban ezt nem lett volna optimális ennek a megvalósítása, a shéma egyszerűsítése érdekében, a probléma megoldására elkészítettem a recipe (ez a tábla tartalmazza a recepteket alapanyagok nélkül), ügyelve arra hogy a későbbiekben ez bővíthető legyen, az ingredients (ez tárolja azokat az alapanyagokat, amelyek bármely receptben előfordulnak) táblákat, illetve egy olyan táblát, ami azt tartalmazza, hogy a receptekben milyen alapanyagok és milyen mennyiségben szerepelnek. Az elkészült táblákat és kapcsolataikat a következő ábra (4-1. ábra) szemlélteti.



4-1. ábra Végleges EK diagramm

A táblák létrehozása és az adatok betöltése után készítettem egy triggert (esemény bekövetkezéséhez kötött tárolt eljárás) is, ami az orders tábla price oszlopába számítja ki a rendelés árat, az alapanyag árak, a szükséges alapanyag mennyiség és a rendelés mennyiség tekintetében az adatsor beszúrását követően. (3. melléklet)

4.2 Fejlesztő környezet kialakítása

A fejlesztői környezet kialakításakor a kliens és a szerver oldali beállításokat is elvégeztem. Igyekeztem a meglévő tudásom kiegészíteni a fejlesztésre és biztonságra vonatkozó szabványokkal (GDPR, NIS2, internet protokollok, stb), valamint a fejlesztői ajánlások (clean code, SOLID elvek). Elsőnek a kliens oldalon telepítettem a Vite-t parancssorból, ami létrehozta a projektet az általam kiválasztott React és Typescript technológiákkal, majd telepítettem a szükséges függőségeket. Olyan modulokat melyek elengethetetlenek a dinalikus megjelenítéshez és a felület logikai kezeléséhez. Ezután integráltam a Material UI könyvtárat, hogy könnyebben ki tudjam majd alakítani a felhasználói felületet. Ez egy React-komponens könyvtár, ami modern, reszponzív design-elemeket biztosít. A kész elemek jelentően gyorsították a munkámat.

A szerver oldal kialakításakor a „npm i -D ts-node-dev” parancs futtatásával elértem a Typrescript támogatását, majd típusdefiníciókat telepítettem annak érdekében, hogy a fordító statikusan ellenőrizni tudja a kódot, valamint a modern JavaScript funkciókat is elérje. Ezt követően az express.js webszerver és a Typescript fordításához, futtatásához szükséges modulok is a projektbe kerültek. Azért, hogy a már kialakított Postresql adatbázist használni lehessen a „npm i @sequelize/postgres” parancsal egy ehhez szüséges ORM (Object Relational Mapping) alapú csomagot is telepítettem, azért, hogy a program forráskódjában reprezentáva legyenek az adattípisok, melyek az adatbázis manipulációs műveletekhez kellenek

A fejleszői környezet kialakítáásának utolsó lépéseként telepítettem és kialakítottam egy React Routert a kliens oldalon. A különböző útvonalak az alkalmazás más-más funkcióira vezetnek, ami megkönnyíti az oldalak közötti navigációt a felhasználó számára és javítják a kód olvashatóságát.

4.3 Backend és frontend fejlesztése

4.3.1 Modell és adatbázis kapcsolat

A backenden kialakítottam az adatbázis tábláinak megfelelően a modelleket, megadtam a táblák mezőinek nevét, típusát és a táblák kapcsolatát is. A kapcsolatok beállításakor az adatbázisnak megfelelő kapcsolatokat igyekeztem létrehozni hasMany, belongsTo, belongsToMany elemekkel, melyek az egyedkapcsolatokatokat hatását érhetjük el a modeljeink között. Az így létrehozott oszályok az ORM koncepció részei, ezek biztosítják a kapcsolatot az adabázis táblái és a program kódja között és így a programban objektumokként kezelhetjük őket ezen felül lehetővé teszi az adatok struktúrált kezelését, és platform függetlenné teszi, azaz bármikor várhatunk MySQL, vagy épp Oracle adatbázisr, működni fog. A programhoz az adatbázist (4-2. ábra) a következőképp kapcsoltam:

```
dotenv.config()
const pw=process.env.DBPASSWORD;

const sequelize = new Sequelize({
  dialect: 'postgres',
  database: 'Szakdolgozat',
  username: 'postgres',
  password: pw,
  host: 'localhost',
  port: 5432,
  clientMinMessages: 'notice',
});

sequelize.addModels([Customer, Employee, IngredientAmounts, Ingredients, Order, Recipe, User])
sequelize.sync()
export default sequelize
```

4-2. ábra Adatbázis összekapcsolása a programmal

Az adatbázis paramétereinek magadása után, hozzá adtam a táblák modelljeit is, így a sequalize segítségével ORM alapon használhatom majd őket. A Sequelize lehetővé teszi az adatbázis-modellek és az SQL műveletek program szintű használatát, csökkenti az SQL utasítások használatából adódó hibákat, valamint SQL szempontból nyelv független, ezáltal könnyítve az esetleges jövőbeli adatbáziskezelő rendszer váltást. Biztonsági szempontból az adatbázis jelszavát egy környezeti változóként a .env fájl tartalmazza, amit a dotenv csomaggal töltöttem be, így ez nem szerepel közvetlenül a forrás fájlban. Ennek használata egy bevett gyakorlat és lehetőséget ad az érzékeny adatok elkülönítésére, megakadályozva, hogy direktbe a kódba legyen égetve, írva, vagy véletlen felfedjük azokat, mivel ehhez a fájlhoz hozzáadódik a .gitignor, ami kivonja a verzókövetés alól. Ezekkel elérve a CIA triád és az OWASP Top 10 betartását, melyek védenek a nem targetált támadások nagyrészétől. Valamint az elkülönülésének köszönhetően könnyen áttekinthető és karbantartható, ráadásul platformfüggetlen is.

4.3.2 Frontend és ügyfél oldalak

Ezt követően a frontenden kezdtem el dolgozni, ahol a legnagyobb kihívás az jelentette, hogy a felhasználók számára egyszerű és intuitív felületet biztosítsak. Az ügyfeleknek, dolgzóknak és vezetőségi tagoknak is más, a számukra releváns tartalmak jelennek meg, így fontos volt számomra a könnyen navigálhatóság. Ahhoz, hogy az egyes szerepköröket lefedve valamint az egyes felhasználók képességeit felmérve megfelelő legyen a használhatóság, számos szempontot kell figyelembe vennünk, hiszem egy admin jogosultságú felhasználó magasabb informatikai tudással rendelkezik mint egy átlagos vásárló, emiatt és a rá eső feladatok miatt más szemléletmód kell a számára releváns felületek kialakításához.

Ezt a folyamatot a fejléc és az ügyfelek számára látható oldalak elkészítésével kezdtem. A frontend készítésekor nagy segítségemre volt a Material UI [27] oldala, ami egy nyílt forrású, React komponenseket tartalmazó könyvtár és kész sablonok segítik az egyes vizuális elemek megalkotását, gyorsítva a repeptitív folyamatokat és javítva a fejlesztési időt. Az ügyfelek számára látható legtöbb oldal esetébenesetében csak a megjelenésre kellett koncentrálnom mivel ezek az oldalak nem a backendtől kapják tartalmukat, hanem a statikus szöveg miatt ők maguk tartalmazzák azt. Kivételt képzett azonban az Árlista és a Rendelés oldalak, ahol a kliens és szerver oldalaknak kommunikálniuk kellett az adat megjelenítéséhez. Az Árlista esetén egy összetettebb, aggregáló lekérdezést készítettem (4-3. ábra), hogy a termék neve mellett, az adott sütemény árát és kalória tartalmát is ki tudjam kalkulálni az egyes alapanyagok értékeinek ismeretében. Az aggregáló műveletek és a csoportosítás, több utána járást igényelt, de végül sikerült elérni a kívánt eredményeket. A legnagyobb kihívást a modellek helyes összekapcsolása és a szintaxis jelentette, szerencsére az első sikeres megvalósítást követően a többi endpoint hívást már könnyebb volt elkészíteni.

```
app.get('/pricelist', async (req, res) => {
  try {
    const result = await Recipe.findAll({
      attributes: [
        'title',
        [fn('SUM', col('ingredientAmounts.price')), 'price'],
        [fn('SUM', col('ingredientAmounts.calory')), 'calory'],
      ],
      where: { lable: 0 },
      include: [
        {
          model: Ingredients,
          as: 'ingredientAmounts',
          attributes: [], // No need to select individual attributes
        },
      ],
      group: ['Recipe.recipe_id', 'Recipe.title',
        'ingredientAmounts->IngredientAmounts.recipe_id',
        'ingredientAmounts->IngredientAmounts.amount',
        'ingredientAmounts->IngredientAmounts.ingredient_id'],
    });
    res.status(200).json(result);
  } catch (err) {
    console.error(err);
    res.status(500).send('Internal Server Error');
  }
});
```

4-3. ábra Árlista lekérdezése

A Rendelés oldalon az ügyfelek süteményekre irányuló rendeléseit lehet leadni, itt a hibák, például, nem megfelelő formátumú értékadás, vagy elgépelés, csökkentésének érdekében az adatbázisból az Árlista hívásához hasonlóan készítettem egy endpoint hívást, ami a sütemények neveit listázza és egy legördülő listában ezekből válogathat az ügyfél. Az űrlapon meg kell adni a megrendelő email címét, a választott süteményt, kívánt mennyiséget és az átvétel dátumát. A megadott adatok formai helyességét a zod könyvtár segítségével ellenőrzöm. Az űrlap alján található Megrendelés feliratú gomb megnyomásával az adatok továbbítódnak a szerver felé. A kapott adatok között email cím és a recept neve szerepel, az adatbázis táblája viszont az ügyfél és a recept azonosítóját várja, így egy-egy filter utasítással szűröm ki a szükséges adatot és create utasítással tárolom el a táblában. Az email cím és a recept megnevezése könnyebben megjegyezhető az emberek számára, így ezeket az adatokat kérem be, viszont az adatbáziskezekő könnyebben és gyorsabban keres kulcs alapján, így neki ezt továbbítom, ezértvan szükség az adatok kikeresésére a kapott adat ismeretében.

4.3.3 Dolgozói oldalak és CRUD metódusok

A fejlesztési folyamatokat a dolgozói oldalak létrehozásával folytattam. Itt mindegyik megjelenő laphoz szükség volt CRUD (Create, Read, Update, Delete) műveletekre, azaz létrehozó, olvasó, módosító és törlő funkciókra.

Az adatok létrehozása a rendelés rögzítéséhez hasonlóan történt minden esetben. A frontenden egy űrlap gyűjti be az adatokat, amiket a zod könyvtár segítségével ellenőrzők, majd a backend create utasítással a megfelelő táblába szúrja az adatot.

A megjelenítést legtöbb esetben egyszerű, vagy lapozható táblázat segítségével oldottam meg. A lapozhatóság segíti korlátozni az egyszerre megjelenő adat mennyiségét, ezzel átláthatóbbá és feldolgozhatóbbá téve az információt, emellett az oldal hatékonyságát is javítja, hiszen nem kell egyszerre sok adatot betöltenie. Állapotkezelő változók segítségével, mint például az aktuális oldal száma, a megjelenítendő elemek száma, dinamikusan befolyásolható az oldal megjelenése. A lapozó vezérlők teszik lehetővé az oldalak közötti navigálást, valamint az oldalon megjelenő sorok számát.

Az adatok módosításakor legördülő listából lehet kiválasztani a módosítani kívánt adatsort. Eltárolódik a kiválasztott elem, az űrlap kitöltésével és a gomb megnyomásával (4-4. ábra) az új adatok a szerver oldalra érkeznek, amennyiben van kiválasztott elem és a szabályoknak is megfelelnek az új adatok. Az szerver ellenőrzi a módosítani kívánt adat létezését és update utasítással frissíti.



4-4. ábra Alapanyag frissítése a weblapon

A törlést szerettem volna a lehető legegyszerűbben a lehető legkevesebb hibalehetőséggel megoldani, így a frontenden egy legördülő listából lehet kiválasztani a törlendő elemet, amelyet a rendszer a backendről kérdez le. Habár ez extra forgalom, de így biztosítva van a törlendő elem kivlaztása és annak naprakészsége, ezáltal lehetőséget sem adva null érték, vagy elgépelésből fakadó hiányzó érték törlésének kisérletére. Ez a kiválasztott elem érkezik vissza a szerver oldalra és egy destroy utasítás segítségével történik a törlés a megfelelő táblában.

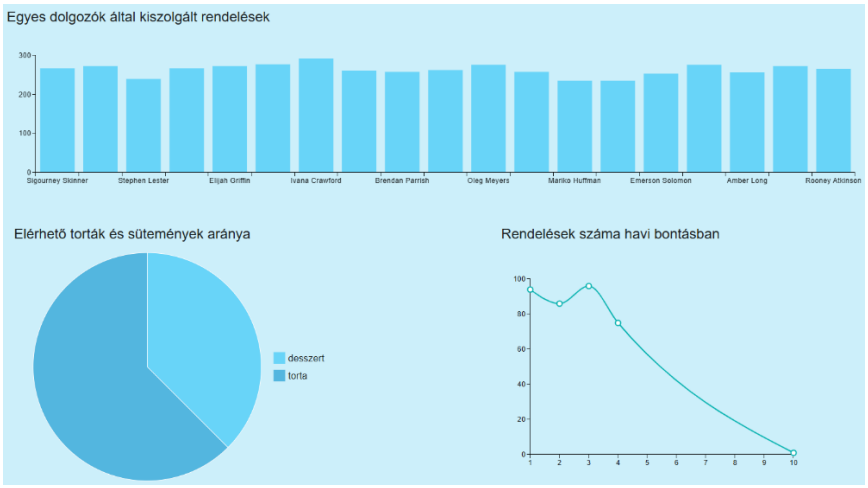
Az alapanyag igényléséhez a backend endpoint hívást készít melyben a minimunérték alá esett alapanyagkészleteket listázza ki, illetve le is tárolja a listát egy-egy viewban az adatbázisban. A kliensnek továbbított lista jelenik meg a felhasználónak és ezt tudja módosítani, vagy épp új elemet felvenni a listára ha épp nagyobb begrendelés érkezetezért előre láthatólag több alalpanyagra lesz szükség a szokásosnál. A módosítani vagy felvenni kívánt adatok a backendre küldi a frontend ahol módosításra kerül a kívánt változtatás a viewban.

A CRUD műveletek alatt, egy-egy egyszerű utasítát értünk néhány paraméterrel, azonban a sequalize segítségében ezek SQL utasításokká fordítódnak és ezek futnak le az adatbázisban.

4.3.4 Adatvizualizáció a vezetőségi oldalakon

A vezetőség számára elérhető oldalak között készült egy statisztikai oldal, ahol vizuálisan jelennek meg az adatok. A @mui/x-charts könyvtár telepítése után már képes voltam diagrammokat létrehozni, amelyek majd a döntési folyamatokat segíthetik. A három

különböző diagramm esetében különböző módon oldottam meg az adat lekérést, volt, ahol egy korábbi végpontot használtam fel és úgy választottam ki a szükséges adattagokat, egy másiknál újat hoztam létre és a harmadiknál egy korábbi végpontot a frontenden csoportosítottam, alakítottam át, ezáltal csökkentve a fejlesztési időt és az ismétlődő endpoint hívásokat. A különböző diagramm típusokat megfelelően paraméterezve pedig könnyen kirajzolhatóvá vált az információ (4-5. ábra).



4-5. ábra A program által létrehozott diagrammok

4.3.5 Regisztráció, autentikáció

A regisztrációs adatlap kitöltése után a zod könyvtár ellenőrzi, hogy a megadott adatok megfelelnek-e az elvárt schemának, például, hogy a kötelező mezők ki lettek-e töltve, megfelelő formátumú email címet adtak meg, vagy a jelszó hosszúságát. Ezután az adatokat egy POST kéréssel küldjük a megfelelő végpontra. A szerver ellenőrzi, hogy létezik-e az email cím az adatbázisban, ha igen hiba üzenetet küld, viszont, ha nem bcrypt segítségével titkosítjuk a jelszót és eltároljuk az adatokat. A bcrypt egy lépszerű titkosító algoritmus, melyet kifejezetten jelszavakhoz fejlesztettek.

A belépés hasonlóan zajlik az regisztrált felhasználó megadja az email címét és jelszavát az űrlapon amit a kliens tovább küld a szervernek itt a rendszer ellenörzi, hogy létezik e az email cím és a bcrypt.compareSync beépített metódussal ellenőrzi a megadott jelszó és a tényleges egyezését. A sikeres hitelesítést követően egy JSON Web Tokent (JST) generál. A tokent a böngésző eltárolja, így a későbbiekben is lekérhető, ha szükséges. A kilens ezt a tokent kapja vissza a szervertől, amit hitelesített kérésekhez használ.

4.3.6 Fejléc

A fejléc (4-6. ábra) felelős az oldalak közötti navigációért, ami szintén a Material UI könyvtár eszközeit használja. A logó helyezkedik el legfelül, középen, alatta pedig a dinamikusan vezérelt menüpontok a felhasználók ezeket szerepkörtől függően érhetnek el.

A null értékkel létrehozott állapot változót API hívásokkal a useEffect hook felülírja a bejelentkezett felhasználó adataival az oldal automatikusan újra renderelősik és frissiti a menüsort. Mivel a felhasználói információt a dekódolt tokenből nyeri, így email címet és szerepkört kapja adatként, ezért egyszerű szerepkör ellenőrzéssel más-más menüpontokat jelenít meg dinamikusan. Ha még nincs bejelentkezett felhasználó, akkor jobb oldalt a bejelentkezés és regisztráció navigációs linkek érhetőek el, viszont, ha van, akkor egy üdvözlő szöveg látható.

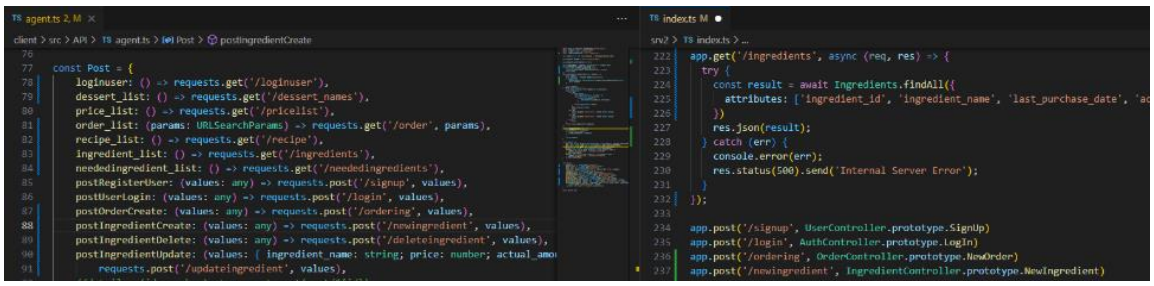


4-6. ábra Fejléc

4.3.7 Frontend és backend kommunikációja

Az alkalmazás GET és POST kérések (4-7. ábra) segítségévet kommunikál. GET kérés esetén a frontend betöltésekor az agent kérést küld a szerver felé a megfelelő végponton keresztül. A backend fogadja a kérést és lekéri a megfelelő adatokat az adatbázisból, amit válaszként visszaküld a frontendnek JSON formátumban. A frontend ezt a useEffect hook segítségével állapotváltozókba menti és megjeleníthető formába alakítja.

POST kérés esetén, pont fordítva történik, a fontend küld valamilyen információt, jellemzően valamilyen esemény által vezérelve, például egy gomb lenyomása. Ekkor meghívódik az agent megfelelő függvénye és továbbítódnak az értékek JSON formátumban az adott végpontra. A backend kiolvassa az érkező értékeket és végrehajtja a szükséges műveletet az adatbázisban, valamint választ küld a művelet sikerességéről, amit a frontend fogad.



4-7. ábra Frontend és backend kapcsolata

5 BIZTONSÁG

5.1 Biztonsági elvek

5.1.1 CIA Triad

A CIA Triad (5-1. ábra) betarásával egy alapcszintű belső és külső védelmet tudunk biztosítani az alkalmazásuknak függetlenül annak paramétereitől. A CIA triád a Titkosság (Confidentiality), Integritás (Integrity) és Elérhetőség (Availability) elérése egy rendszer esetében.

A titkosság vonatkozik az adatok hozzáférhetőségére, az erre vonatkozó jogosultságokra, valamint az adatok titkosított tárolására. Különös tekintettel a felhasználói személyes adatokra melyek rendkívül szenzitív adatoknak számítanak a GDPR és NIS2 értelmében.

Az integritás betratása érdekében ügyelnünk kell az adatok sérthetetlenségére vonatkozik elsősorban. A hozzáférés korlátozás itt is nagy szerepet játszik ugyanis kizárólag az végezhet bármilyen módosítást az adatokon, aki arra jogosult, implikálva a hozzáértő kezeket. Ebbe a kategóriába tartoznak a napló fájlok, valamint a biztonsági mentések.

Ahhoz, hogy az elérhetőséget is biztosítani tudjuk külön figyelmet kell a rendszerre és az adatokra is fordítani. Kiemelt szerepet játszik a redundancia melynek pontos felmérése és meghatározása kulcsfontosságú. Amennyiben a teljes rendszer redundánsan üzemel, rengeteg erőforrást pazarolunk el. A redundancia csökkentése kizárólag a fő- és szenzitív komponensekre egy megefelelő út lehet azonban ez már függ az adtott felhasználástól. Hasonlóan fontos figyelni a terheléselosztását mely elsődleges és természetes védelemként tud funkcionálni a DDos (túlterheléses) támadások ellen.

Amennyiben sikerül ezeket a szempontokat biztosítani úgy a kiberbiztonsági stratégiánk megfelelő pilléreken áll.



5-1. ábra CIA elvek

5.1.2 OWASP Top 10

Az OWASP (Open Web Application Security Project) Top 10 egy lista a leggyakoribb fenyegetésekről melyek az alkalmazásokat és online rendszereket érhetik. A 2024-es lista elemeinek az alábbiakat jelölték meg:

Broken Object Level Authorization vagyis a Hibás Objektum Hitelesítés mely az objektumokra vonatkozó nem megfelelő jogosultságkezelés végrehajtásánál jöhet elő. Mely eredményezhet illetéktelen hozzáférést, távoli kód végrehajtást, valamint adat maipulációt.

A Hitelesítési hiba okozhat még problémát szerencsére az erős jelszavak használata és a többlépcsős azonosítás (MFA) hozzájárul az illetéktelen hozzáférés eelkerüléséhez.

API hívások használatakor fontos odafigyelnünk azok biztonságos használatára, ugyanis az alkalmazások rétegei közötti kommunikáció megbénulása és kikerülése növeli a sérülékenységet.

Server side request forgery (Szervoldali kérés hamisítása) mely http kérések indításával szerez meg érzékeny adatokat vagy hozzáférést rendszerekhez.

5.1.3 GDPR és NIS2 szabályozások

Az Európai Unióban az összes éles alkamazás és webes rendszer adatvédelmi jogszabály által kötelezve van a GDPR (General Data Protection Regulation) vagyis az Általános Adatvédelmi Rendelet betartására. A 2018. május 25.-én hatályba lépő rendelet célja, hogy megvéje az Európai Unió állampolgárainak személyes adatait biztosítva azok biztonságos kezelését.

Ez a rendelet kitér a felhasználók saját adatainak módosítására és törlésére, valamint az adatok felhasználásának korlátozására. Kiemelt hangsúlyt kap az adatkezelés átláthatóságának biztosítása és annak kikényszerítése. A felhasználók minden esetben tályékoztatva kell, hogy legyenek arról, hogy pontosan milyen adatokat és milyen formában tárol el az adott szolgáltatás. A jogszabály megsértése esetén jelentős pézzbírság kerül kiszabásra mely elérheti akár az éves globális forrgalom 4%-át. [28]

A Directive on Security of Network and Information Systems a Hálózati és Információs Rendszerek Biztonságának Irányelve mely szintén az Európai Unióban érvényes irányelv 2020 decembere óta. Az elódje (NIS irányelv) 2016 óta van használatban és az európai kritikus infrastruktúra védelmét hivatott megerősíteni

A NIS2 a tagállamok közötti együttműködést hivatott növelni, melyre a kibertámadások megelőzése érdekében van szükség. Lehetőséget biztosítva a Kibertámadások bejelentésére és az ilyenkor életbelépő protokollok kidolgozására. Továbbá kibővíti az érintett szektorok számát bevonva az egészségügyet, a pénzügyi szektort, az energia és a közlekedési infrastruktúrákat.

A NIS2 összeségében az EU kiberbiztonsági ellenálló képességét hivatott növelni. [29]

5.2 Biztonság a gyakozlatban

5.2.1 Környezeti változók

Az érzékeny adatok számára, mint például jelszavak, kiemelt biztonságot nyújt a .env fájl. Ennek köszönhetően az ilyen jellegű adatok nem kerülnek a forráskódba, illetve a .env a .gitignore fájlba is felvételre kerül, így nyílvános verziókezelés esetén sem szivároghat ki annak tartalma.

5.2.2 Bcrypt

A bcrypt (5-2. ábra) egy kifejezetten jelszavak titkosítására használt algoritmus, mely hash-eléssel védekezik első sorben a brute force és rainbow table támadásokkal szemben. A bcrypt sózással és iterációkkal készít hash-eket, ami biztosítja, hogy még azonos jelszavak esetén is eltérő hash keletkezzen. Egy egyszerűen kiszámítható mégis visszafordíthatatlan matematikai műveletet használ, a sózással pedig egy egyedi random karakterláncot fűz a jelszóhoz, ez biztosítja az azonos jelszavak, megkülönböztethetőségét. Az iterációk, azaz ismétlések száma tetszőlegesen választható, a nagyobb érték növeli a biztonságot, ugyan akkor a kódolási és összehasonlítási időt is. () A bcrypr.compareSync egy biztonságos módja a jelszó helyességének ellenörzésére.

```
export class UserController{
    public async SignUp(req: Request,res: Response){
      const {nanoid} = await import("nanoid")
      const {password, name, email}=req.body;
      const salt_round=6;
      try {
        const hash_pw = (await bcrypt.hash(password, salt_round)).toString()
        const ifexist = await User.findOne({where: {email:email}})
        if(!ifexist){
          const _id=nanoid(7)
          await User.create({user_id:_id, name:name, email:email, password:hash_pw}
          res.send(200)
        }
      }
      catch (err) {
        console.error(err);
        res.status(500).send('Internal Server Error');
      }
    }
}
```

5-2. ábra Bcrypt használata regisztrációkor

5.2.3 JSON Web token

A JSON Web token a belépéskor generálódik és biztonságos munkamenetet biztosít. A token segítségével korlátozhatjuk a hozzáférést a tartalmakhoz és kliens oldalon a localStorage-ból egyszerűen beazonosíthatjuk a felhasználót a segítségével, anélkül hogy egyfolytába jelszóval kellene azonosítanunk. Mivel lejárati idővel rendelkezik, így a token eltulajdonítása esetén is minimalizálható a károsítás. Hátránya azonban, hogy visszavonhatatlan, így a lejárati ideig érvényven marad, valamint a benne tárolt adatoknem titkosítottak így nem szabad érzékeny adatot tárolni benne.

5.2.4 Adatvalidáció

A beiteli mezőkből esetlegesen érkező SQL injection és hozzá hasonló támadások ellen véd a zod könyvtárral történő adat ellenörzés. A hibás illetve átró szándékú adatok kezelésének kiszírése érdekében fontos a formátum ellenörzés.

5.2.5 Jogosultság kezelés

A felhasználói szerepkörök lehetővé teszik hogy mindenki a számár szükséges, de csak is a szükséges információhoz férhessen hozzá ezzel csökkentve érzékeny adatok kiszivárgását, illetéktelen kézbe kerülését. Ezen felül a felhasználói élményt is növeli azáltal, hogy felesleges tartalmat nem jelenít meg.

5.2.6 CRUD műveletek biztonságos megvalósítása

A CRUD műveletek implementálásakor igyekeztem minél több ellenörzés és hibaszűrést alkalmazni az adatbázis integritásának védelme érdekében. (5-3. ábra) Inyen például a törlés esetén , hogy csak legördülő listából választhatóa törlendő elem, ezzel bitzosítva hogy naprakész és létező adatsor legyen. Másik példa a módosításkor a módosítandó elem kiválasztását követően a beviteli mezők alapérelmezett értékként az eredeti adatokat tartalmazzák, ezzel irányt mutatva a bevivendő adat formátumának, mely schema ellenörzésen is átesik rögzítés elött.



5-3. ábra Hibaszűrés törléskor

6 TESZTELÉS

Egy allalmazás fejlesztése során kiemelt szerepet kap a tesztelés melynek számos formája van. Legyen szó unit tesztekről melyek az adott funkciók működését hivatottak ellenőrizni elülönítve az éles környezettől, kiszűrve az összes funkción kívül eső hatást. Léteznek még integrációs tesztekre melyek a komponensek egymással való kommunikációját ellenőrzi. Valamint End to End tesztek melyek a teljes folyamatot vizsgálják. A tejlesítmény mérésére alkalmas tesztek az erőforrások eloszlását és a rendszer terhelését vizsgálják. Bitonsági tesztek, melyek a sérülékenységeket elemzik ki.

Mivel a fejlesztés a tervezettnél tovább tartott unit tesztekre nem volt elegendő idő, ezért a teszteléshez családtagokat és barátokat, ismerősöket kértem meg hogy próbálják ki az alkalmazást és mondják el véleményüket róla. Ez lehetőséget ad arra, hogy ne csak fejlesztői szemszögből vizsgáljam a kérdést, hiszen nagyon fontos volt számomra és a projekt szempontjából a felhasználói élmény. Igyekeztem minden korosztályból és különböző informatikai ismeretekkel rendelkező tesztalanyokat gyűjteni, hogy biztosan bárki biztonsággal használhassa az alkalmazást.

Ennek a megfelelő tesztelésére egy weboldal használhatósági mély interjú módszertant választottam, melyben az alanyok egy előre meghatározott feladatot hajtottak végre, amit utána videófelvétel segítségével elemeztünk ki az alannyal közösen. Ez a megközelítés megfelelőnek bizonyult több metrika mérésére párhuzamosan. A feladat elvégzésének ideje mérésre került, ami bár nem a fő optimalizációs szempont, megefelelő visszajezés a felhasználó által eltöltött gondolkodási idő és az oldal válaszidejének arányáról. A teszt során fontos volt, hogy a feladat végrehajtás semleges környezetben történjen ahol az alany képes fókuszálni a feladatra. A felvételre szükségességét az elemzés elkülönítése indokolja. Amennyiben a felhasználó narrálja a cselekedeteit, végigvezet a gondolatmeneten, úgy az idő mérése látja kárát. Azonban, ha elvégezte a feladatot, az utólagos elemzés során van lehetőség a felvétel megállítására visszatekerésére, hogy kellő betkintést nyerjek az alany gondolatmenetére és a döntéseinek indokára, esetleg felmerülő kérdésekre észrevételekre.

A kísérletet 10 alannyal végeztem el, akik változatosan lettek kiválasztva, ügyeltem arra, hogy minél nagyobb korosztály tartományt lefedjek, így 19 és 82 év közötti személyeket kértem fel segíteni. Másik fontos szempont volt az eltérő szintű informatikai ismeret, így a résztvevők között szerepelt például a 19 éves unokatestvérem, aki számítástechnikai területen tanul, illetve a nagymamám is, akinek isemeretei igen visszafogottak. (6-1. táblázat)

Alany	Életkor	Tudás szint	Távoli asztaleléréssel
Testvér	21	Általános felhasználói szint	Igen
Mama	76	Alacsony tudásszint	Nem
Apa	55	Általános felhasználói szint	Nem
Szaktárs 1	24	Magasabb szintű	Igen
Barát 1	25	Általános felhasználói szint	Nem
Unokatestvér	19	Magasabb szintű	Nem
Szaktárs2	24	Magasabb szintű	Nem
Anya	47	Általános felhasználói szint	Nem
Barát2	23	Általános felhasználói szint	Nem
Papa	82	Alacsony tudásszint	Nem

6-1. táblázat Tesztalanyok eloszlása

Az esetleges hibák és eltérő feltételek szűrése érdekében a résztvevők a fejlesztésere használt laptopon dolgoztak, azért, mert a projekt kizárólag lokális hálózaton érhető el, valamind így a hálózati hibákat is szűrni tudtam. A 10 főből 2-en nem tudtak személyesen részt venni, így ők Team Viewer (6-1. ábra) segítségével távoli elérésen keresztül jelentkeztek be a gépbe, a tesztek eredményét ez nem befolyásolta.



6-1. ábra Team Viewer logó

Általánosságban kellemes benyomást keltett az alkalmazás letisztultsága és átláthatósága. Alapvetően a regisztrációt, belépést, navigálást, különböző szerepköröknek megfelelő funkciókat funkciókat vizsgáltuk. Illetve volt olyan is, akit kifejezetten a hibahelyzetek szimulálására kértem fel.

A regisztráció a legtöbb esetben problémamentesen zajlott, azonban, olyan eset is előfordult, hogy elgépelték a jelszót így az adat hibásan került rögzítésre és csak a belépés során derült ki, hogy az adott személy nem tud belépni emiatt. Ennek kiküszöbölésére a jelszót duplán kell megadni és jelzésre kerül a felhasználóknak, ha a megerősítő jelszó eltér az elsődlegestől.

Nagyon örültem. hogy a navigácó egyértelműnek és logikusnak bizonyult. A legkevésbé hozzáértőek is sikeresen megtalálták és elérték a kívánt oldalakat. Viszont igény merült fel a sütemények listázásánál, hogy lehessen szűrni, illetve, hogy allergéneket is mutasson, ez a jövőben mindenképpen beépíteném, mert valóban hasznos ötletnek találom.

A visszajelzések alapján az új elemek felvétele, régiek módosítása és törlése is egyértelmű, logikus és könnyen kezelhető volt. Elemmódosítás esetén előfordult, hogy a program nem fogadta el az új adatokat, mert nem felelt meg formailag az elvárásoknak, viszont nem érkezett egyértelmű hibaüzenet. Ez a hiba javításra került, így minden ilyen estben részletesebb hibaüzenet érkezik, például „Ebbe a mezőbe csak számokat lehet beírni”. Javaslatot is kaptam, hogy a rendelés leadása elött lehetne egy összegzés, amikor még egyszer átnézheti a vásárló az adatok helyességét. Ez a későbbiekben szintén szeretném megvalósítani.

A hibahelyzetek szimulálása során, szintén felmerülő probléma volt a kevés, illetve a pontatlan hibaüzenetek. A javítás során igyekeztem a lehető legtöbb esetet lefedni, minél konkrétabb hibajelzéssel. Két létrehozó művelet során kiderült, hogy üres érték is bevihető, ami hibás működéshez vezetett volna, ezekben az esetekben a schema ellenőrzésnél plusz feltételként hozzá adtam szempontnak, hogy az érintett adattagok, nem vehetnek fel null értéket (6-2. ábra).

```
const schema = zod.object({
    ingredient_name: zod.string().min(1, { message: "Name is required" }),
    price: zod.number().min(1, { message: "Need at least 1" }),
    actual_amount: zod.number().min(1, { message: "Need at least 1" }),
    minimum_amount: zod.number().min(1, { message: "Need at least 1" }),
    unit: zod.string().min(1, { message: "Unite is required" }),
    calory: zod.number().min(1, {message: "Need at least 1"}),
  });

  type Values = zod.infer<typeof schema>;
```

6-2. ábra Javított schema validálás

Összeségében úgy gondolom, hogy pozitív visszajelzést kaptam az alkalmazással kapcsolatban, arányaiban kisebb kibákkal, melyek leginkább a hibás adatkezelés következménye és sok javaslattal arra, hogy mit lehetne még hozzá adni a felhasználói élmény javításának érdekében. (6-2. táblázat6-2. táblázat)

Funkció	Hiba/javaslat	Megoldás
Regisztráció	Elgépelhető jelszó	Jelszó duplikált megadása és azonosságának ellenörzése
Sütemények listázása	Igény a szűrésre és allergének megjelenítésére	Jövőbeni implementálás
Új rendelés felvétele (dolgozó által)	Nem elég részletes hibaüzenet	Hibaüzenetek átfogalmazzása, konkrétabban utalva a hiba okára, jellegére
Új rendelés leadása	Rendelés összegzése leadás elött	Jövőbeni implementálás
Új alapanyag felvétele	null érték elfogadása	Feltétel javítása a validációs schemában
Új rendelés felvétele	null érték elfogadása	Feltétel javítása a validációs schemában

6-2. táblázat Hibák/javaslatok és kezelésük

7 FELHASZNÁLÓI KÉZIKÖNYV

A projekt során készült webes alkalmazás bármely böngészőn keresztül elérhető, mint például Google Chrome, a Mozilla Firefox és a Microsoft Edge, plusz eszköz telepítése nélkül. Az alkalmazás a helye webcím megadása után érhető el.

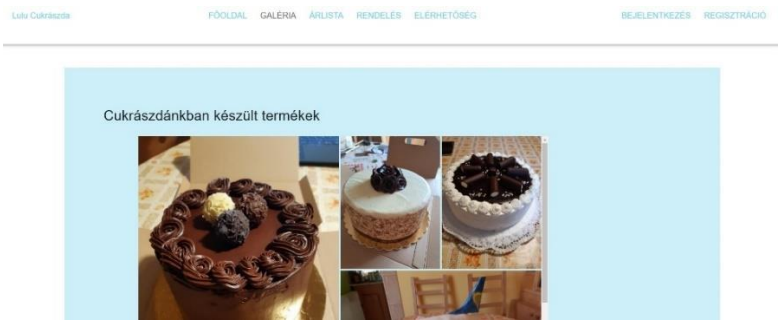
Az alkalmazás megnyitásakor a betöltődik a kezdőoldal, ahol a cukrászda logóját látjuk meg először, alatta középen helyezkedik el a menüszalag, vele egy vonalban jobbra a felhasználó azonosítására szolgáló menüpontok, lentebb pedig az oldal üdbözlő szövege. (7-1. ábra)



7-1. ábra Főoldal

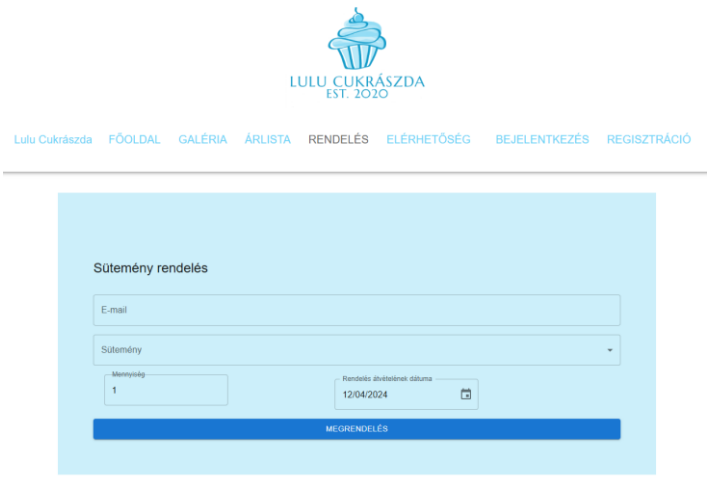
A felület lehetőséget ad regisztrációra, ahol a felületen az adatok megadása és a gomb megnyomását követően új funkciók válnak elérhetővé. Amennyiben rendelkezünk regisztrált fiókkal az email címünket és jelszavunkat begépelve a Bejelentkezés menüpont alatt található beviteli mezőkbe, majd egy gomb megnyomásának segítségével bejelentkezhetünk. Amennyiben a felhasználó egy ügyfél, akkor ezentúl lehetősége lesz sütemények rendelésére, viszont, ha dolgozó, vagy vezetőségi tag, akkor a menüsor megváltozik és más oldalakat fog elérni.

Ügyfélként a menüszalag segítségével érhet el a cukrászdával és termékekkel kapcsolatos információkat a Főoldal, a Galéria (7-2. ábra), az Árlista, vagy épp az elérhetőségek menüpont kiválasztásával.



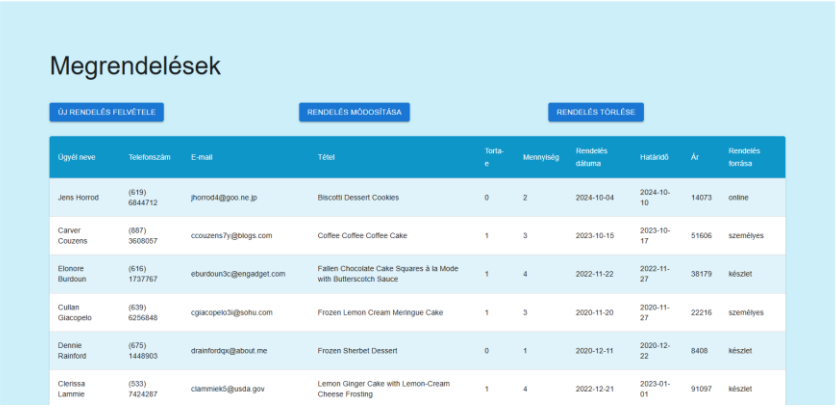
7-2. ábra Galéria

Mivel egy torta esetén több komplikáltabb kérdés is felmerülhet, például a díszítés komplexitása, így a Rendelés menüpont kiválasztásával, jelenleg csak süteményeket rendelhetünk. Az alkalmazás ügyfelek számára elérhető része vendégként is használható, azonban a rendeléshez (7-3. ábra) be kell jelentkezni, ezt követően lehet az űrlapot kitölteni és beküldeni a rendszerbe.



7-3. ábra Süteményrendelő oldal

A dolgozók a bejelentkezést követően, az új menüszalag segítségével a rendeléseket, recepteket, alapanyagolat és a jelentéseket érik el. Az első három menüpont nagyon hasonló módon működik. Az oldal megjeleníti a rendeléseket és az alapanyagok adatait, a lap tetején található három gomb pedig új adat létrehozására, meglévő módosítására, vagy törlésére szolgál. (7-4. ábra)



7-4. ábra Megrendeléseket kezelő oldal

Ha megnyomjuk a létrehozó gombot, az átirányít egy új oldalra, ahol megadhatjuk az új elem adatait és egy másik gombbal elmenthetjük (7-5. ábra). A módosítás ugyan így zajlik, megadjuk a módosítani kívánt értéket és gombnyomással mentjük. Törléskor egy legördülő listából választjuk ki a törlendő elemet és ezt követi a Törlés feliratu gomb megnyomása, ami véglegesíti a változtatást.



7-5. ábra Új elem felvétele űrlap

A receptek esetén a megjelenítés kicsit másabb mivel egy listáról ki kell választani melyik receptet szeretnénk megnézni és ez jelenik csak meg. (7-6. ábra) Itt is megtalálható a fent említett három gomb, a törlés egyszerüen a Törlés gomb megnyomásával végbe megy, ha a felületen kiválasztunk egy receptet, a megjelenő recept kerül eltávolításra. Új recept és módisítás esetén a gombok egy-egy új oldalra vezetnek ahol űrlapok kitöltését követően ismét gombokkal menthetjük el a változásokat. A módosításnál, annyi eltérés van, hogy először ki kell választani, mit szeretnénk módosítani, az alapanyagokat, vagy a leírást. A jelentéseket a rendszer automatikusan jeleníti meg.



7-6. ábra Receptek megjelenítése

Az alapanyag rendelé esetén minden rendelési mennyiségnél szerepel egy beviteli mező mejdben módosítható a kívánt mennyiség, majd a Módosítás feliratú gomb segítségével rögzíthetjük a változtatást. A lap tetején található + gombbal új elemet vehetünk fel a listára amit szintén a Módosítás gomb segítségével tárolhatunk el.

A vezetőségi tagok a nekik megfelelő menüpontok között mozoghatnak, azonban, hogy ezeket elérjék nekik is be kell jelentkezniük.

8 TOVÁBBFEJLESZTÉS

8.1 Tesztelési javaslatok megvalósitása

A projekt továbbfejlesztését mindenképp a tesztelés során javasolt elemekkel kezdeném. Az allergének esetében az alapanyagok táblában létrehoznék egy új oszlopot, ahova eltárolhatnám az információt. A receptekből kiderül melyik desszert milyen alapanyagokból épül fel, így ha bármilyen, így ha bármilyen allergiát okozó összetevőt tartalmaz azt jelezhetjük a felhasználók számára. Például az árlistán a kalória tartalom mellett ezt is feltüntethetjük. A szűrő esetében három féle szűrési opciót hoznék létre, egyet az allergéneknek ahol kiszűrhető, hogy, milyet nem szeretnénk megjeleníteni a listában, több választásos listát használnék, hiszen lehet valaki több mindenre is allergiás és olyan endpoint hívást készítenék ami kiszűri ezeket. A második szűrő a kalóriára irányulna a diétázók számára, itt legördülő listát alkalmaznék fix intervallumokkal és ezekből lehetne választani. A harmadik egy kulcsszavas kereső lenne ahol egy textfieldbe be lehetne írni egy kulcsszót és azt ellenőrizném hogy a recept neve tartalmazza-e.

A rendeléseknél egy összegzésre irányuló igény merült fel. Ebben az esetben a megrendelő gombot összegzésre cserélném és kiiratnám részletesen az adatokat, ki, mit, mikor, mennyiért, stb. rendelt. Ezután lenne lehetőség visszalépni és módosítani a rendelést, vagy véglegesíteni és ekkor kerülne ténylegesen rögzítésre.

8.2 Bankkártyás fizetési lehetőség

Mostanában mindenki bankkártyával (8-1. ábra) fizet, így praktikus lenne ezt a lehetőséget biztosítani a felhasználók szára. Szükség lesz egy fizetési szolgáltató kiválasztására (Bairon, PayPal), aki a tranzakciók feldolgozását biztosítja. A szolgáltatónál történő regisztrációt követően kapunk egy API kulcsot, amit a backend kódjában helyezhetünk el. A frontendről érkező adatokat megkapja a backend, ami átirányít a szolgáltató fizetési oldalára, majd a fizetést követően a backend értesítést kap a tranzakció sikerességéről.



8-1. ábra Bankkártya illusztráció

8.3 Mobil applikáció

A mobiljaink folyton a kezünkben vannak, így célszerű lenne a webes felület mellé egy mobil applikációt is létrehozni a kényelmesebb és gyorsabb elérés érdekében. Erre több módszer is létezik, legegyszerűbb és leggyorsabb, ha csak mobilra optimalizáljuk a kész alkalmazást, azonban ilyenkor nem éri el a natív funkciókat, például az értesítéseket. Hibrid mobilalkalmazás esetén a webes technológiák natív alkalmazásként futnak, ez gyorsan fejleszthető és több platformre is használható, viszont jelentősen nagyobb és lassabb lesz a program. Harmadik opció, ha teljesen natív alkalmazást készítünk. Ebben az esetben több erőforrást igényel a fejlesztés, de gyorsabb lesz és teljes hozzáférést biztosít a natív funkciókhoz, tehát például küldhetünk értesítést, ha új rendelés érkezik. React segítségével létrehozhatunk olyan mobil applikációt, mely a már meglévő backendet használja rest API hívásokon keresztül és kész React komponenseinket is felhasználhatjuk.

9 ÖSSZEGZÉS

A web fejlesztést mindig is érdekesnek találtam, olyasminek, amit egyszer szeretnék kipróbálni, megtanulni. A szakdolgozat keretein belül rengeteg új technológiát és módszert ismerhettem meg és saját magam számára is bebizonyíthattam, hogy képes vagyok egy összetett rendszer megalkotására.

A projekt kezdetén kevés ismerettel rendelkeztem a témáról, főleg annak tekintetében, hogy az egyes elemek (adatbázis, backend, frontend) hogyan kapcsolódnak és kommunikálnak egymással. Az adatbázis megtervezését és létrehozását kifejezetten élveztem, ebben a témában konkrét elképzelésen volt mit szeretnék és hogyan, ez megadta a kezdeti lendületet. Az adatok gyűjtésére és generálására szánt időt sajnos alábecsültem és a webfejlesztés terén jelentkező tudáshiányom is lassította a projekt előmenetelét. Mindezek ellenére, úgy gondolom nagyon sokat tanultam, annak köszönhetően, hogy egy számomra teljesen idegen témát választottam és sokat fejlődtem. Jobban értem, hogy hogyan fonódnak össze az egyes elemek és válik egy egységgé, mennyi munkát és időt igényel egy-egy fejlesztési projekt.

Összeségében elégedett vagyok azzal, amit létrehoztam. Határozottan szeretném tovább fejleszteni és tökéletesíteni, többet tanulni az informatika ezen részéről és akár a későbbiekben, szívesen dolgoznék is ezen a téren.

10 SUMMARY

I always found web developing interesting that I wanted to try out and learn. Within the confines of my thesis, I got to know several modern technologies and methods and I could prove myself that I am able to create a complex system.

At the beginning of the project, I had little knowledge about the topic, especially regarding that how components (database, frontend, backend) connect and communicate to each other. I really enjoyed planning and creating the database, as I had a concrete image about this topic, what I wanted and how, that provided the initial momentum. Unfortunately, I underestimated the time spent on collecting and generating the data and the lack of my knowledge in the area of web developing also slowed down the progress of the project. Despite all this, I think I learned a lot of thanks to choosing an unfamiliar subject and I have improved a lot. I have a better understanding of how the components intertwine and become a unit, how much work and time is required to a developing project.

Overall, I'm happy with what I created. I would definitely like to further develop and improve it, learn more about this part of IT, and even in the future, I would also like to work in this area.

IRODALOMJEGYZÉK

[1] „bakesmart,” [Online]. Available: https://www.bakesmart.com/. [Hozzáférés dátuma: 30 04 2024].

[2] „Sweet Shop Manager,” [Online]. Available: https://sparkresto.com/industries/sweets-shop. [Hozzáférés dátuma: 30 04 2024].

[3] „CakeBoss,” [Online]. Available: https://cakeboss.com/features/. [Hozzáférés dátuma: 30 04 2024].

[4] „generatedata.com,” [Online]. Available: https://generatedata.com/. [Hozzáférés dátuma: 24 04 2024].

[5] „Kaggle,” [Online]. Available: https://www.kaggle.com/datasets. [Hozzáférés dátuma: 24 04 2024].

[6] „Google Dataset Search,” [Online]. Available: https://datasetsearch.research.google.com/. [Hozzáférés dátuma: 24 04 2024].

[7] „GitHub,” [Online]. Available: https://github.com/. [Hozzáférés dátuma: 24 04 2024].

[8] „mackaroo,” [Online]. Available: https://www.mockaroo.com/. [Hozzáférés dátuma: 24 04 2024].

[9] „GitHub Cake,” [Online]. Available: https://github.com/dpapathanasiou/recipes/blob/master/index/c/cake.md. [Hozzáférés dátuma: 24 04 2024].

[10] „GitHub Dessert,” [Online]. Available: https://github.com/dpapathanasiou/recipes/blob/master/index/d/dessert.md. [Hozzáférés dátuma: 24 04 2024].

[11] „Visual Studio,” [Online]. Available: https://azure.microsoft.com/hu-hu/products/visual-studio. [Hozzáférés dátuma: 06 04 2024].

[12] „Visual Studio Code,” [Online]. Available: https://code.visualstudio.com/?wt.mc_id=DX_841432. [Hozzáférés dátuma: 06 04 2024].

[13] L. Quoy, „Visual Studio vs Visual Studio Code: What’s the Key Difference?,” [Online]. Available: https://distantjob.com/blog/visual-studio-vs-visual-studio-code/. [Hozzáférés dátuma: 06 04 2024].

[14] „IntelliJ IDEA,” [Online]. Available: https://www.jetbrains.com/idea/?var=1. [Hozzáférés dátuma: 04 06 2024].

[15] „PyCharm,” [Online]. Available: https://www.jetbrains.com/pycharm/. [Hozzáférés dátuma: 06 04 2024].

[16] K. Babitz, „SQL vs NoSQL Databases: Key Differences and Practical Insights,” [Online]. Available: https://www.datacamp.com/blog/sql-vs-nosql-databases?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720824&utm_adgroupid=152984014494&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adpostion=&utm_creative=684592140620&utm_targeti. [Hozzáférés dátuma: 07 04 2024].

[17] „DB-Engines Ranking - Trend Popularity,” [Online]. Available: https://db-engines.com/en/ranking_trend. [Hozzáférés dátuma: 06 05 2024].

[18] „Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others,” [Online]. Available: https://www.altexsoft.com/blog/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/. [Hozzáférés dátuma: 10 04 2024].

[19] B. Panchal, „Django vs NodeJS: Which is Better for Web Development?,” [Online]. Available: https://radixweb.com/blog/django-vs-nodejs. [Hozzáférés dátuma: 16 04 2024].

[20] „Node.js Vs Django: Top Differences That Web Developer Should Know [2024],” [Online]. Available: https://www.geeksforgeeks.org/nodejs-vs-django/. [Hozzáférés dátuma: 16 04 2024].

[21] J. Babu, „Node Vs Django Vs Rails: The most demanding backend framework in 2023,” [Online]. Available: https://www.fortunesoftit.com/node-vs-django-vs-rails/. [Hozzáférés dátuma: 16 04 2024].

[22] „React,” [Online]. Available: https://legacy.reactjs.org/. [Hozzáférés dátuma: 16 04 2024].

[23] C. Deshpande, „What is Angular?: Architecture, Features, and Advantages,” [Online]. Available: https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular. [Hozzáférés dátuma: 16 04 2024].

[24] „Vue.js Tutorial,” [Online]. Available: https://www.geeksforgeeks.org/vue-js/. [Hozzáférés dátuma: 16 04 2024].

[25] „Templates,” [Online]. Available: https://docs.djangoproject.com/fr/4.2/topics/templates/. [Hozzáférés dátuma: 16 04 2024].

[26] H. Dhaduk, „Best Frontend Frameworks for Web Development,” [Online]. Available: https://www.simform.com/blog/best-frontend-frameworks/. [Hozzáférés dátuma: 06 05 2024].

[27] „https://mui.com/material-ui/getting-started/,” [Online]. [Hozzáférés dátuma: 29 11 2024].

[28] „AZ EURÓPAI PARLAMENT ÉS A TANÁCS 2016. április 27-i (EU) 2016/679 RENDELETE a természetes személyeknek a személyes adatok kezelése tekintetében történő védelméről és az ilyen adatok szabad áramlásáról, valamint a 95/46/EK irányelv hatályon kívül helyezésé,” [Online]. Available: https://net.jogtar.hu/jogszabaly?docid=A1600679.EUP. [Hozzáférés dátuma: 14 12 2024].

[29] „The NIS2 Directive,” [Online]. Available: https://www.europarl.europa.eu/RegData/etudes/BRIE/2021/689333/EPRS_BRI(2021)689333_EN.pdf. [Hozzáférés dátuma: 14 12 2024].

[30] O. Nir, „OWASP Top Ten 2024 – The Complete Guide,” [Online]. Available: https://www.reflectiz.com/blog/owasp-top-ten-2024/. [Hozzáférés dátuma: 14 12 2024].

[31] C. C. Almagro, „NIS2 Impact on Electronic Communications Networks Providers”.

ÁBRAJEGYZÉK

2-1. ábra Lenovo Ideapad 5 6
2-2. ábra Visual Studio Code logo 6
2-3. ábra DB-Engines Ranking 8
2-4. ábra Node.js logó 11
2-5. ábra Frontend rangsor a simform.com szerint 2022-ből 13
3-1. ábra EK diagram 14
3-2. ábra Az Ingredient tábla forrásfájlja....14
3-3. ábra A Recipe tábla forrás fájlja 15
3-4. ábra Az Order tábla forrás fájlja 15
3-5. ábra A Customer tábla forrás fájlja 15
3-6. ábra A webalkalmazás architektúrája....16
3-7. ábra Az alkalmazás Use Case diagramja....17
3-8. ábra Ügyfelek által elérhető weboldal....18
3-9. ábra Bejelentkezési oldal....18
3-10. ábra Dolgozók által elérhető weboldal....19
3-11. ábra Vezetőség által elérhető weboldal....19
4-1. ábra Végleges EK diagramm....20
4-2. ábra Adatbázis összekapcsolása a programmal....22
4-3. ábra Árlista lekérdezése....23
4-4. ábra Alapanyag frissítése a weblapon....25
4-5. ábra A program által létrehozott diagrammok....26
4-6. ábra Fejléc....27
4-7. ábra Frontend és backend kapcsolata....27
5-1. ábra CIA elvek....28
5-2. ábra Bcrypt használata regisztrációkor....30
5-3. ábra Hibaszűrés törléskor....31
6-1. ábra Team Viewer logó....33
6-2. ábra Javított schema validálás....34
7-1. ábra Főoldal....36
7-2. ábra Galéria....36
7-3. ábra Süteményrendelő oldal....37
7-4. ábra Megrendeléseket kezelő oldal....37
7-5. ábra Új elem felvétele űrlap....38
7-6. ábra Receptek megjelenítése....38
8-1. ábra Bankkártya illusztráció....39

TÁBLAJEGYZÉK

6-1. táblázat Tesztalanyok eloszlása..... 33
6-2. táblázat Hibák/javaslatok és kezelésük..... 35

MELLÉKLETEK

1. **melléklet**

Az Ingredient táblát forrás fájlját létrehozó kód

```
class Program
{
	static Random rand = new Random();

	static void Main(string[] args)
	{
		StreamReader r = new StreamReader("Recipes2.json");
		string json = r.ReadToEnd();
		List<ingredient> i =
	JsonConvert.DeserializeObject<List<ingredient>>(json);
		List<Inglist> list = new List<Inglist>();
		for (int j = 0; j < i.Count(); j++)
		{
			for (int l = 0; l < i[j].ingredients.Length; l++)
			{
				Inglist element = new
			Inglist(i[j].ingredients[l].ingredient,
				DateTime.Parse("2024." + rand.Next(1,
			13).ToString() +"."+ rand.Next(1, 28).ToString()),
			rand.Next(100, 1000),
				i[j].ingredients[l].unit, rand.Next(20, 150),
			rand.Next(15,40));
				list.Add(element);
			}
		}
		string json2 = JsonConvert.SerializeObject(list.ToArray());
		File.WriteAllText("ingredients.json", json2);
	}
}


public class ingredient
{
	public Ingredients[] ingredients;
}

public class Ingredients
{
	public double amount;
	public string unit;
	public string ingredient;
	public string comment;
}
```

```
public class Inglist
{
	public string Name;
	public DateTime PurchaseDate;
	public int Price;
	public string Packaging;
	public int Actual;
	public int Minimum;
	public Inglist(string Name, DateTime PurchaseDate, int Price, string
Packaging, int Actual, int Minimum)
	{
		this.Name = Name;
		this.PurchaseDate = PurchaseDate;
		this.Price = Price;
		this.Packaging = Packaging;
		this.Actual = Actual;
		this.Minimum = Minimum;
	}
}
```

2. melléklet

Order tábla forrás fájlját létrehozó script

```
class Program
{
	static Random r = new Random();
	static void Main(string[] args)
	{
		string[] d = File.ReadAllLines("Order.csv");
		List<DateTime> dates = new List<DateTime>();
		List<order> orders = new List<order>();

		for (int i = 0; i < d.Length; i++)
		{
			dates.Add(DateTime.Parse(d[i]));
		}
		for (int i = 0; i < dates.Count(); i++)
		{
			order o = new order(r.Next(1, 438), r.Next(1, 2001),
		dates[i], dates[i].AddDays((double)(r.Next(1, 14))),
		r.Next(1000, 25000));
			orders.Add(o);
		}

		string json = JsonConvert.SerializeObject(orders.ToArray());
		File.WriteAllText("orders.json", json);
	}
}

public class order
{
	public int Recipe_ID;
	public int Customer_ID;
	public DateTime Order_Date;
	public DateTime Completation_Date;
	public int Price;

	public order(int Recipe_ID, int Customer_ID, DateTime Order_Date,
DateTime Completation_Date, int Price)
	{
		this.Recipe_ID = Recipe_ID;
		this.Customer_ID = Customer_ID;
		this.Order_Date = Order_Date;
		this.Completation_Date = Completation_Date;
		this.Price = Price;
	}
}
```

3. melléklet

Ár számító trigger az Orders táblához.

```
CREATE OR REPLACE FUNCTION counter() RETURNS TRIGGER AS $$
BEGIN
	UPDATE orders o
		SET price=subquery.price
		FROM (SELECT o.order_id, ROUND(sum(i.price * ia.amount*o.amount))
	as price
			FROM public.ingredient_amounts ia
			inner join ingredients i on
		i.ingredient_id=ia.ingredient_id
			inner join recipe r on r.recipe_id=ia.recipe_id
			inner join orders o on o.recipe_id=r.recipe_id
			group by o.order_id) AS subquery
		WHERE o.order_id=subquery.order_id;
	RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE OR REPLACE TRIGGER Price_counter
AFTER INSERT ON public.orders
FOR EACH ROW
EXECUTE FUNCTION counter();
```