

**Szegedi Tudományegyetem**  
**Informatikai Intézet**

# **SZAKDOLGOZAT**

**Kocsor Levente Ferenc**

**2019**

**Szegedi Tudományegyetem  
Informatikai Intézet**

**Adatbázis alapú recept-kezelő portál tervezése és fejlesztése**

**Szakdolgozat**

Készítette:

**Kocsor Levente  
Ferenc**

informatika szakos hallgató

Témavezető:

**Dr. Németh Tamás**

egyetemi adjunktus

**Szeged  
2019**

## Feladatkiírás

A feladat egy adatbázis alapú recept-kezelő portál tervezése és fejlesztése. Az alkalmazásban legalább három féle jogosultsági szintet lehessen egy-egy felhasználóhoz rendelni: látogató, felhasználó, üzemeltető. A regisztráció során megadott adatokra a szokásos validációt alkalmazza, valamint az alkalmazás biztonsági védelemmel legyen ellátva (SQL injection,...).

A regisztrált felhasználónak legyen joga új tartalmat felvinni a weboldalra, amit a többi felhasználó is láthat. A felvitt tartalommal ne lehessen visszaélni, nem odaillő adatokat/szöveget nyilvánossá tenni. Legyen valamilyen folyamathoz kötve a recept felvitele (pl. jóváhagyás, moderáció alkalmazása).

A recepteknél az elkészítési instrukciókon felül legyen megjeleníthető kép és az elkészítéshez szükséges hozzávalók listája. Az ételeket címkézni kell valamilyen kategória szerint amire később szűrni lehessen kereséskor.

Mivel webes portálról beszélünk ezért biztosítani kell a recepteknél hozzászólási felületet. A hozzászólásokat lehessen moderálni valamilyen szinten. Csak a regisztrált felhasználóknak legyen joga írni hozzászólást.

A főoldalon legyen többféle ajánló receptekről. Az alkalmazásban lennie kell egy személyre szabott ajánlásnak is, ami valamilyen felhasználó előzményt vesz alapul.

Az oldalon többféle módon lehessen keresni. A recept nevén kívül egyéb adatokra is kell szűrési lehetőséget kell biztosítani. Ha több paraméterre szűrünk akkor legyen megengedő keresés, ami csökkenő sorrendben mutatja, hogy mennyi egyezést talált.

Az alkalmazásban több kategóriájú jogkör legyen a moderátorok között, amelyeket egy oldalon lehessen kezelni. Legyen kimutatás egyes adatokról az oldalon, amit az oldal üzemeltetője láthat.

## Tartalmi összefoglaló

- **A téma megnevezése:**

Adatbázis alapú webes alkalmazás készítése

- **A megadott feladat megfogalmazása:**

A feladat webes alkalmazás készítése, valamint egy összetett, de átlátható adatbázis létrehozása egy adott témához fűződően. Az alkalmazáshoz tetszőleges programnyelvek használhatók

- **A megoldási mód:**

A feladatot PHP + MySQL alapokra építettem. Először a webes felületet terveztem, majd lépésről lépésre dolgoztam ki a komponenseit.

- **Alkalmazott eszközök, módszerek:**

Alkalmazott szoftver-eszközök, melyek a webes felület létrehozását segítették:

Photoshop CC, PhpStorm, Notepad++

Alkalmazott szoftver-eszközök, melyek a PHP+MySQL motort segítették:

XAMPP (PHP 7.4.3, Apache 2.4.41 + MYSQL 7.4.3 szerver egységes működtetése - freeware)

Alkalmazott szoftver-eszközök, melyek a prezentációt segítették:

Creately, MS Word 2019, Photoshop CC

Használt programozási és szabványos lekérdező nyelvek:

PHP, SQL, HTML, Bootstrap, CSS, JavaScript, jQuery

- **Elért eredmények:**

A Diplomamunka eredményeként létrejött egy olyan felhasználói interfész, valamint egy olyan adatbázis, amely lefedi az előre elképzelt funkciókat.

- **Kulcsszavak:**

MySQL, PHP, SQL, adatbázis-kezelés, dinamikus weboldal, szerver oldali programozás

# Tartalomjegyzék

<b>Feladatkiírás.....</b>	<b>3</b>
<b>Tartalmi összefoglaló .....</b>	<b>2</b>
<b>Tartalomjegyzék.....</b>	<b>3</b>
<b>BEVEZETÉS.....</b>	<b>5</b>
<b>1. ALKALMAZÁS FELÉPÍTÉSE.....</b>	<b>2</b>
1.1. Webes alkalmazás .....	2
1.2. Statikus és Dinamikus weboldalak.....	2
1.3. Architektúrák.....	3
1.3.1 Kliens-szerver modell.....	3
1.3.2 Többrétegű modell .....	4
<b>2. FEJLESZTÉSI TECHNOLÓGIÁK.....</b>	<b>5</b>
2.1. Kliensoldali technológiák .....	5
2.1.1 HTML .....	5
2.1.2 CSS.....	5
2.1.3 JavaScript .....	6
2.1.4 Bootstrap .....	6
2.1.5 jQuery.....	7
2.2 Szerveroldali technológiák.....	7
2.2.1 Apache.....	7
2.2.2 XAMPP .....	8
2.2.3 PHP .....	8
2.2.3 phpMyAdmin .....	10
2.3 Adatbázis-kezelés .....	10
2.3.1 SQL .....	11
2.3.2 RDBMS.....	11
2.3.3 MySQL.....	11
<b>3. TERVEZÉS.....</b>	<b>14</b>
3.1 Az alkalmazás felhasználói.....	14
3.1.1 Látogató/Visitor szerepkör .....	15
3.1.2 Felhasználó szerepkör .....	15
3.1.3 Moderátor szerepkör.....	16
3.1.4 Admin szerepkör .....	17
3.2 Az alkalmazás főbb funkciói .....	18
3.2.1 Autentikáció és autorizáció .....	18
3.2.2 Receptek és hozzászólások.....	18
3.2.3 Keresési lehetőségek .....	18
3.2.4 Toplista és profilok.....	19
3.2.5 Saját receptek és profil .....	19

3.2.6 Recept szerkesztés.....	19
3.2.7 Recept felvitele.....	19
3.2.7 Moderálás.....	20
3.2.8 Vétózás.....	20
3.2.9 Üzenetek.....	20
<b>3.3 Adatbázis-terv.....</b>	<b>20</b>
3.3.1 Szükséges táblák meghatározása.....	20
3.3.2 ER modell.....	21
3.3.2 Relációs adatmodell.....	22
<b>4. IMPLEMENTÁCIÓ.....</b>	<b>23</b>
<b>4.1 Adatbázis implementációja.....</b>	<b>23</b>
4.1.1. Táblák létrehozása.....	23
4.1.2. Idegen kulcs és további megszorítások.....	24
4.1.3. felhasználók.....	24
4.1.4. recept.....	24
4.1.5. pontozás.....	25
4.1.6. kategória.....	25
4.1.7. kategoriák.....	25
4.1.8. jogok.....	26
4.1.9. alapanyagok.....	26
4.1.10. merkekegység.....	27
4.1.11. alapanyagok_merkekegység.....	27
4.1.12. hozzaavalok.....	27
4.1.13. hozzaszolasok.....	28
4.1.14. hozzaszolasok_report.....	28
4.1.15. uzenetek_temak.....	28
4.1.16. uzenetek_ticket.....	29
4.1.17. uzenetek.....	29
<b>4.2. Főbb funkciók megvalósítása.....</b>	<b>30</b>
4.2.1. db.con.php.....	30
4.2.2. menu.php.....	30
4.2.3. Autentikáció és autorizáció.....	31
4.2.4. Receptek és hozzászólások.....	33
4.2.5. Keresési lehetőségek.....	35
4.2.6. Toplista és profilok.....	37
4.2.7. Saját receptek és profil.....	38
4.2.8. Recept szerkesztés.....	38
4.2.9. Recept felvitele.....	38
4.2.10. Moderálás.....	39
4.2.11. Üzenetek.....	40
4.2.12. Vétózás.....	40
4.2.13. Kimutatások.....	41
<b>Irodalomjegyzék.....</b>	<b>2</b>
<b>Nyilatkozat.....</b>	<b>3</b>
<b>Mellékletek.....</b>	<b>4</b>
<b>Köszönetnyilvánítás.....</b>	<b>5</b>

## BEVEZETÉS

Régen a legfinomabb receptjeink családban maradtak, vagy jószomszédok között, de mára már szinte minden embernek van internet hozzáférése és így a nagymama híres hókiflijét a világ minden pontján elkészíthetik. A mai világban azonban fontos, hogy minél kevesebb idő menjen el az ételünk tervezésével és elkészítésével. Azonban nem csak az íze számít már, hanem a benne lévő tápértékek is.

A szakdolgozatom célkitűzése az, hogy létrehozzak és bemutassak egy webes alkalmazást, amelyet ChefBot néven kereszteltem el. Ez az alkalmazás egy webes recept portál, melynek feladata, hogy egyszerűen és átláthatóan lehessen az oldalra feltöltött recepteket kezelni.

Sok hazai oldal foglalkozik ezzel a témával, de a versenytársakkal szemben itt a receptekhez pontos tápértéki adatok is társulnak, amik a hozzávalók változtatásával dinamikusan számlálódnak ki.

Az általam létrehozott alkalmazással szeretném minimalizálni az időt, ami a kereséssel és az adminisztrátori oldalon történő feladatokkal eltelik. Az elengedhetetlen moderátori átvizsgálások megkönnyítése szempontjából külön kimutatásokat és átlátható kezelői felület hoztam létre.

Manapság szinte mindenkinek van internet hozzáférése, vagy internet vételére alkalmas eszköze. Fontos, hogy eszközfüggetlen felületet nyújtsunk a felhasználóknak, vagy legalábbis az általunk szolgáltatott funkciók egyenértékűek legyenek mindenhol. A felhasználói élmény mindenképp számítógépen teljes, de okostelefonokon is lefedi az összes funkcióját.

Az internet térnyerésének köszönhetően az informatika területén az egyik leggyorsabban, legdinamikusabban fejlődő ág a webes alkalmazások fejlesztése. Az első fejezetben a dinamikus weboldal tulajdonságait taglalnám mielőtt az alkalmazott eszközökre térnénk. Miután megismertük az alkalmazás alapjait képző komponenseket, betekintünk a tervezési fázisban előállított szerkezeti döntésekre, majd azok hatását a kész projektre. Később a funkciók összetettebb szekcióit nézzük a kódból és elemezzük, valamint azok esetleges továbbfejlesztési lehetőségeit is végig vesszük.

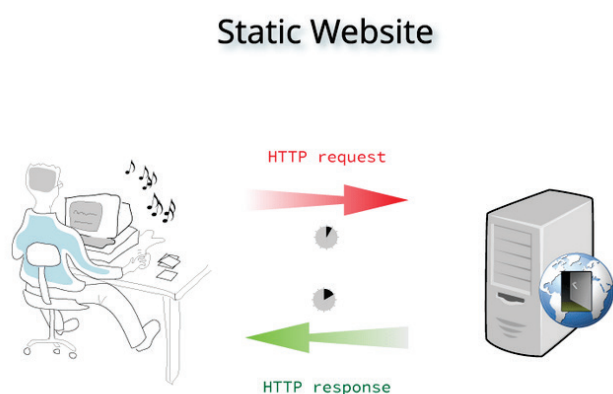
# 1. Alkalmazás felépítése

## 1.1. Webes alkalmazás

A webes alkalmazás fejlesztés több informatikai ágazat kombinációja. A kezdeti statikus weboldalak fokozatosan lecserélték a dinamikus oldalak, melyek alkalmazási logikát használnak, ezeket nevezzük webalkalmazásoknak. A hagyományos és a webes alkalmazások fejlesztése közötti különbséget a technológia hordozza. Eltérő a navigációs struktúra, a felhasználói felület és a használati szokások. Egy webalkalmazástól egyszerűen, könnyen használható funkciókat és platformfüggetlenséget várnak el a felhasználók.

## 1.2. Statikus és Dinamikus weboldalak

Az internetet böngészve két weboldallal találkozhatjuk magunkat szemben: a statikus oldallal, melyek fix adattartalommal bírnak, és akárhány újra töltés után is változatlanok, valamint a dinamikus weboldallokkal. A dinamikus oldalak több tényezőtől függően is adhatnak vissza



adatot és lehetőséget biztosítanak a felhasználó-szerver közti interakcióra, adatcserére.

A dinamikus oldalak rendszerint több felhasználói jogkörrel rendelkeznek és az abban foglalt userekkel kommunikáció zajlik. Technikai részt nézve a statikus úgymond „beleégeti” az adatot a weboldalba, ami dinamikus társával

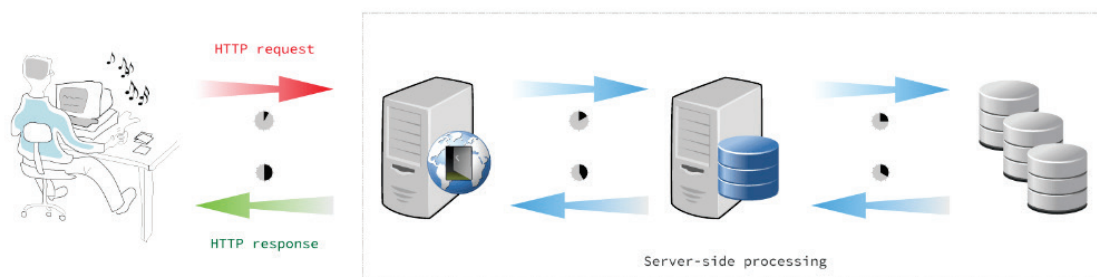
*1.2.1 ábra statikus weboldalak működési elve*

ellenben gyorsabb elérést biztosít megjelenítéskor, de korántsem olyan egyszerű az oldalon lévő adatok kezelése nagy terjedelemmel. Dinamikus oldalak részéről sem feltétlen gondnélküli az életünk, mivel, ha lehetőséget biztosítunk a felhasználónak a kommunikációra, tartalom felvitelre, azáltal hibalehetőségek tárháza merül fel. Napjainkban kiélezett verseny megy a legvárhatóbb user input kihasználási lehetőségekkel és ezektől nem tekinthetünk el.



Felépítésileg a dinamikus oldalak nem léteznek a szerveren, csak a kérésekre válaszul lesz előállítva és ezek kerülnek vissza a felhasználó felé. Az oldalakat jellemzően egy motor hajtja fenntartásuk ezért jóval egyszerűbb. A szerveroldalról egy adatbázisban vannak eltárolva az adatok, amit szükség esetén az előre definiált lekérdezések kinyernek.

### Dynamic Website



[1.2.2 ábra dinamikus weboldalak működési elve](#)

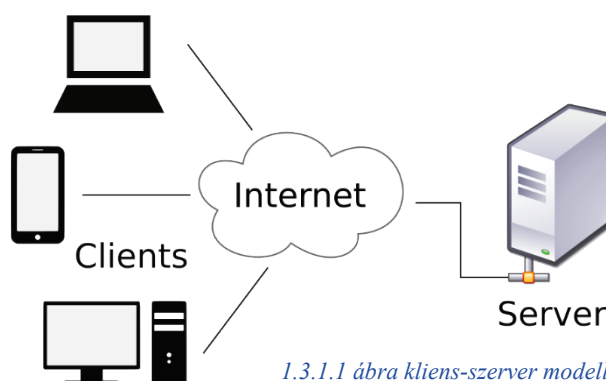
## 1.3. Architektúrák

### 1.3.1 Kliens-szerver modell

A kliens-szerver modell egy szoftver architektúra modell. Két részből áll, kliens és szerver. A kapcsolat köztük történhet hálózaton vagy egyazon gépen. A kliens mindig egy kapcsolatot kezdeményez a szerverrel, ellenben a szerver mindig kérélmekre vár bármely kliens felől. A kliens-szerver alkalmazás egy elosztott rendszer.

A kliens és szerver közötti kapcsolatot UML (Unified Modeling Language) szekvenciadiagrammal szokás ábrázolni. A szerver kommunikálhat más szerverekkel annak érdekében, hogy kiszolgálja a kérélmeket. Ha további adatra van szükség akkor kérelmezhet a kienstől információt feldolgozás előtt.

Az adattárolás centralizált, ezért jól kezelhetők a hozzáférések és az üzemeltetők könnyebben tudják módosítani az információkat. Skálázhatóság szempontjából nem előnyös modell.

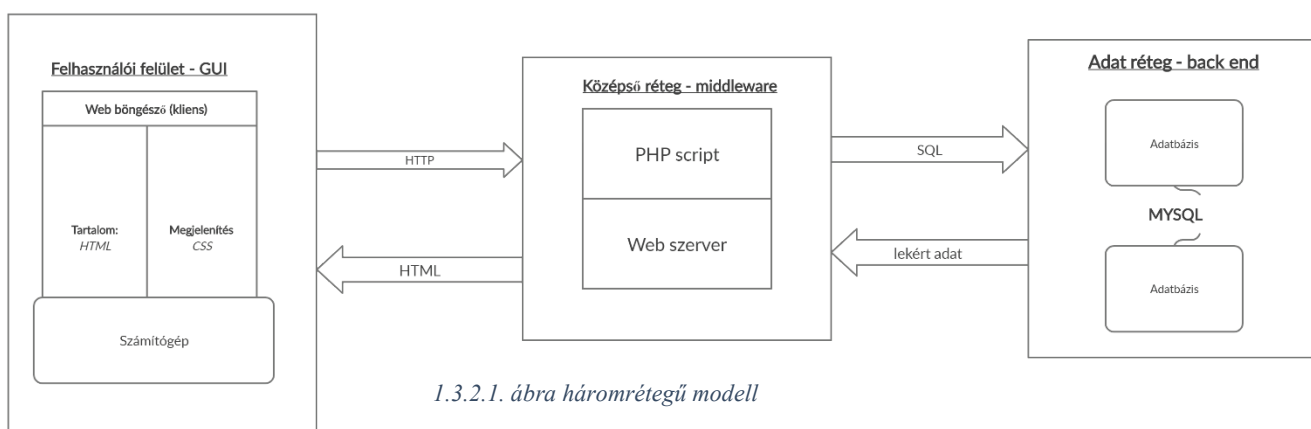


[1.3.1.1 ábra kliens-szerver modell](#)

### 1.3.2 Többrétegű modell

A webes alkalmazások többrétegű (minimálisan három) architektúrára épülnek.

- felhasználói felület – front end (GUI) - egy webszerver, ami a felhasználói felületet alkotó (statikus vagy dinamikusan előállított) weboldalakat szolgáltatja.
- középső (alkalmazás szerver réteg) – middleware - az alkalmazáserver a dinamikus tartalmak előállítását és feldolgozását végzi. Itt található a Java EE, PHP, ASP.NET stb. alkalmazások.
- adat réteg (adatbázis) – back end - az adatok tárolását és hozzáférését végzi, tipikusan egy relációs adatbázis-kezelő rendszer, mint például a MySQL, PostgreSQL, stb.



Egyszerre nagyon sok felhasználó használhatja az-az, a szerver oldali terhelés a speciális technológiáknak köszönhetően sok felhasználó esetén jóval kisebb lehet, mint ugyanannyi felhasználónál a kétszintű architektúra szerverén. A kliens gépen csak egy böngésző található, szinte minden logika a szervereken található. Ezért ezeket az alkalmazásokat vékony kliens rendszereknek nevezzük. A szerveren elkülönül az adattárolás, a logika és a prezentáció.

Az egyes szinteket azok a programozók implementálhatják, akik arra specializálódtak, így specializálódott szerepkörök miatt professzionálisabb alkalmazások készíthetők. A rendszer különböző szintjei önmagukban is tesztelhetők, ami nagy projekteknél hatalmas előny lehet. Mivel csak egy böngészőre van szükség, jellemzően semmilyen vezérlőt sem kell a kliensre telepíteni. A verziófrissítés csak a szerveret érinti, a kliensek nem.

## 2. Fejlesztési technológiák

### 2.1. Kliensoldali technológiák

#### 2.1.1 HTML

A HTML (Hypertext Markup Language, azaz hiperszöveges jelölőnyelv) egy leírónyelv, amit internetes weboldalak létrehozására használnak. Mára internetes szabvánnyá vált a W3C (World Wide Web Consortium) támogatásával. A HTML nem programozási, hanem egy kód nyelv, ami azt jelenti, hogy akár egy Jegyzetömbben is alkothatnánk weboldalakat. A hipertext jelenti az interneten található oldalakat, amelyek képek szövegek animációk kombinációjából tevődik össze. A HTML ezeknek a dokumentumoknak az elrendezését, formázását tartalmazza. A HTML használatakor egy szövegblokk körül van véve tagekkel, amelyek jelzik a böngészőnek, hogy hogyan jelenjen meg a szöveg (például dőlt betűvel). HTML olyan platform-független stílusok gyűjteménye, amelyek meghatározzák egy webes dokumentum különböző komponenseit. A weboldalak készítésének kedvelt eszköze, mert minden böngésző tudja értelmezni.

#### 2.1.2 CSS

Cascading Style Sheets, azaz CSS, egy egyszerű tervezési nyelv. Azzal a céllal készítették, hogy kitöltse a weboldalakon megnőtt igényt az egyre bonyolultabb, kiforrottabb elemekre és formázásokra, amelyeket az egyszerű szöveges információk ábrázolására kitalált HTML-lel nem volt megoldható. A CSS használatával szabályozni lehet a szöveg színét, betűtípusát, bekezdések távolságát, oszlopok méretét és elrendezését, háttérképek vagy háttérszínek használatát és további stílusvariációit.

Sokkal rövidebbé, rugalmasabbá és szabadon kezelhetőbbé tette a HTML dokumentumokat. A CSS-t leggyakrabban a HTML jelölőnyelvvvel használják. Egy stíluslapot több oldalhoz is hozzárendelhetünk, így kódot és időt spórolhatunk. Ha egy stíluslapon módosítunk, akkor az összes HTML oldalon érvénybe lép a változás, amihez hozzá van kapcsolva a CSS fájl. Ha a böngészőben megnyitunk egy oldalt akkor az cache-be lementi a stíluslapot, és legközelebb az oldal megnyitásakor nem kell megvárni, hogy letöltődjön a CSS fájl, mert a böngésző már a gyorsítótárban tárolja. Így időt spórol a felhasználóknak.

### 2.1.3 JavaScript

A JavaScript túllép a HTML-en, egy teljes programozási nyelv bonyolultsága és összetettsége nélkül. Ez egy interpretált nyelv. Szintaxisát tekintve hasonló más magasszintű nyelvekkel (Java, C), de azoknál korlátozottabb. Többparadigmás programozási nyelv, ami támogatja az objektumorientált, imperatív (procedurális) és deklaratív programozási stílusokat. Ez a nyelv prototípus alapú, ami azt jelenti, hogy az öröklődést prototípusok (és nem osztályok) segítségével valósítja meg. A JavaScript nyelvben minden objektum, vagy éppen objektummá alakul át, amikor szükséges. A JavaScript kód vagy a html fájlban vagy külön .js kiterjesztésű szövegfájlban tárolandó. Ezek a fájlok tetszőleges szövegszerkesztő programmal könnyedén szerkeszthetőek. A JavaScript esetében a futási környezet egy webböngésző.

A JavaScript szkriptnyelv. Szkriptnyelveknek azokat a programozási nyelveket nevezzük, amelyek egy adott alkalmazás vagy működési környezet vezérlését teszik lehetővé. A JavaScript a böngésző és a betöltött dokumentum állapotát változtatja meg. A JavaScript kódot a böngészőbe épített JavaScript értelmező (interpreter) értelmezi sorról sorra. Az értelmezett nyelvekben nincsen fordítási fázis, nem a lefordított kód fut, hanem az első lehetőségénél az értelmező elkezd a program végrehajtását, így a programban lévő hiba akkor derül ki, amikor az értelmező ráfut a hibát tartalmazó sorra.

Problémát jelenthet, hogy a felhasználók hozzáférhetnek a kódhoz, így módosíthatják az értékeket és az azokra épülő validációkat, amelyekkel tovább dolgoznánk. Ezt a problémát többszörös adatellenőrzési szintek megvalósításával lehet orvosolni. Kliens oldali ellenőrzéssel, ami azonnali visszajelez a nemkívánatos adatokra, valamint a szerver oldali ellenőrzéssel, ami biztosít minket, hogy nem lett megkerülve az első.

### 2.1.4 Bootstrap

A Bootstrap nem más, mint egy CSS és jQuery nyelveken megírt keretrendszer, amely előre definiált utasításokkal és függvényekkel képes a weboldal dizájnját megjeleníteni. A keretrendszerben osztályok vannak definiálva melyeket hozzáadunk a megfelelő HTML elemekhez. Multifunkcionálisan alkalmazható eszközkészletet kínál, aminek a segítségével gyorsabban, átláthatóbban és hatékonyabban dolgozhatunk. A HTML struktúra és a CSS tulajdonságok mellett számos JavaScript bővítménnyel is rendelkezik. Leggyakrabban említett hátrány a méret, de nem kell az egész kódot használnunk projektünkbe, csupán mindig azt, amire szükségünk van.

### 2.1.5 jQuery

Az egyik legnépszerűbb JavaScript keretrendszer, rengeteg bővítmény tölthető le hozzá. A benne írt kódunk böngészőfüggetlen lesz, azaz bármelyik böngésző bármelyik verziójában ugyanazt az eredményt kapjuk. A framework a HTML-t és a JavaScript-et próbálja jobban összekapcsolni. A JavaScript-ben megismert parancsokat használhatóbbá és egyszerűbbé teszi úgy, hogy az eredeti nyelvből nem veszítünk. Írhatunk jQuery kódot úgy is, ha nem ismerjük a JavaScript-et, azonban bonyolultabb műveletekhez feltétlenül fontos, hogy kellően ismerjük az alapot, amire a keretrendszer épül. Mivel nagy mennyiségű kódrészletet tesz közénk és a programnyelv közé figyelniük kell rá, hogy a lehető leghatékonyabb kódot írjuk meg, hogy a programjaink kellően gyorsak legyenek.

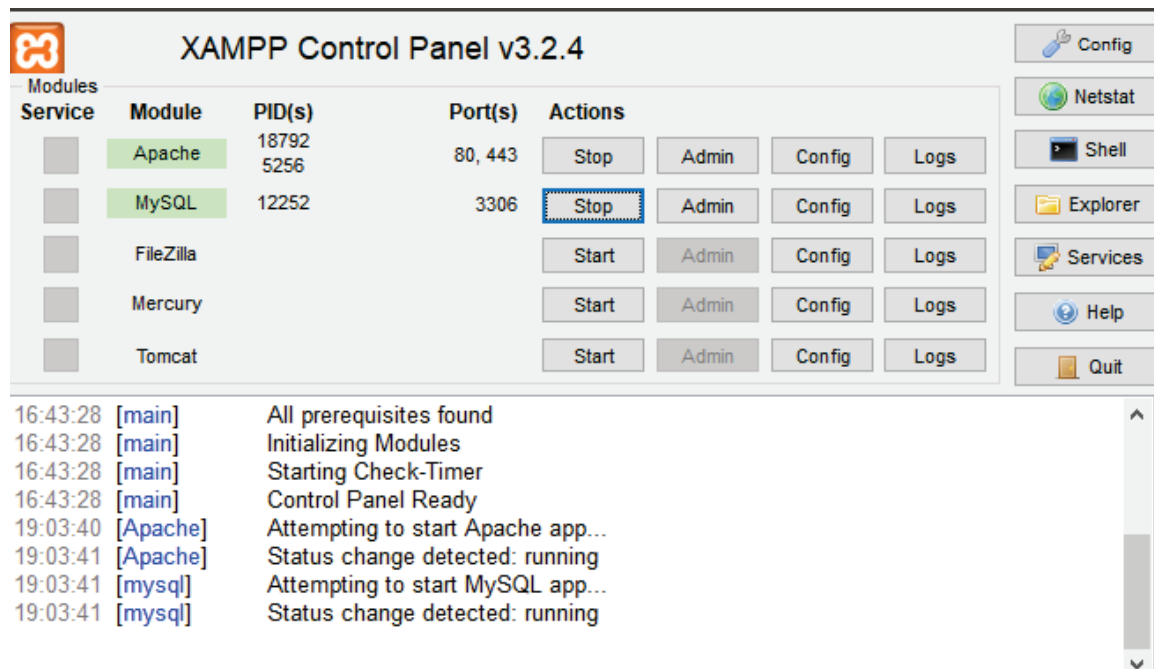
## 2.2 Szerveroldali technológiák

### 2.2.1 Apache

Az Apache egy több operációs rendszeren elérhető, nyílt forráskódú webkiszolgáló. A fejlesztők célkitűzése egy biztonságos és hatékony szerver létrehozása bővíthetőségi képességgel, ami támogatja a http szolgáltatásokat. A jelenlegi http szabványok nagy részét lefordított modulok keretein belül biztosítja kiegészítésként. A virtual hosting segítségével egyetlen Apache installáció ki tud szolgálni több különböző webhelyet is. Az Apache hibaüzenetei konfigurálhatóak. Több felhasználói felület is van az egyszerűbb kezelés érdekében. Statikus és dinamikus weboldalak egyaránt használják. Sok webalkalmazást az Apache által nyújtott szolgáltatásokhoz terveztek. A fejlesztési fázisban a XAMPP programcsomag keretein belül használtam számítógépemen a kódom teszteléséhez.

## 2.2.2 XAMPP

A XAMPP egy alkalmazás-kiszolgáló programcsomag. Fő elemei az Apache webservert, MySQL adatbázis-kezelő rendszer, PHP fordító és a phpMyAdmin modul. Előnye, hogy minden fontosabb kiszolgálóhoz biztosít egy átlátható kezelői felületet:



2.2.1.1. ábra XAMPP GUI

Aki adatbázissal támogatott webszervert szeretne üzemeltetni, annak össze kell hangolnia a webservert, az adatbázisszerver, PHP szolgáltatásait, valamint a kommunikációs portokat és erre nyújt egy leegyszerűsített megoldást. A csomagban megtalálható a webkiszolgáló és az adatbázis-kiszolgáló, amit a projektemhez alkalmaztam.

## 2.2.3 PHP

### 2.2.3.1 Általános leírás

Kezdetben személyes honlapok karbantartási céljaira készült, amit eredeti elnevezése is tükröz (Personal Home Page Tools). Mára már kibővítették és túlnőtt eredeti feladatán, így létrejött egy önálló interpretált programozási nyelv, ami alkalmas dinamikus weboldalak készítéséhez. Szerkezetét tekintve a C nyelvhez hasonlít. A PHP kódok végezhetnek adatbázis-lekérdezéseket, dinamikusan létrehozhatnak képeket, fájlokat olvashatnak és írhatnak, kapcsolatot létesíthetnek távoli szerverekkel.

A PHP szkriptnyelv, ami azt foglalja magában, hogy a programunk kódját bármilyen szövegszerkesztővel meg tudjuk írni. Az esetek többségében a szövegszerkesztőbe már eleve van beépített php nézet, ami színekkel jelzi és elkülöníti az adott kódrészleteket a könnyebb

fejlesztés érdekében. A szkriptünk fordítását minden alkalommal a kiszolgáló szerver végzi meghíváskor. A fordítási időben való futás lassabb, mint egy eleve lefordított kód, de így hordozható a kódunk. A JavaScriptnél említett hátrány itt is jelen van, az-az csak futtatáskor derülnek ki a hibák, amikor a fordító eléri a kódrészletet. Az említett gyengesége miatt a tesztelés nehézkes lehet, mivel, ha jólműködő alkalmazást akarunk akkor minden ágát le kell tesztelni alkalmazásunknak.

### 2.2.3.2 Alkalmazása

PHP alkalmazásának három fő területe:

#### 1. Parancssori alkalmazás

Szerver és böngésző nélkül is készíthetünk így PHP alkalmazásokat, mindössze a PHP értelmezőre lesz szükség. Ez hasznos lehet például feladatok ütemezésére, vagy akár egyszerű szövegfeldolgozásra.

#### 2. Grafikus alkalmazás

A grafikus szó azt jelentené, hogy a program rendelkezik egy grafikus felhasználói felülettel. Ilyen célra más alternatívákkal egyszerűbb a dolgunk.

#### 3. Szerver oldali alkalmazás

Három fő dologra van szükségünk: egy PHP értelmezőre, egy webszerverre, és egy böngészőre. A webszervert egy megfelelően beállított PHP-vel kell futtatnunk. A szerver által és a böngésző segítségével tekinthetjük meg a PHP program kimenetét.

A PHP programokat általában HTML oldalba építve futtatják, ritkán HTML nélkül, parancssoros programként is alkalmazzák. Példa a HTML-be ágyazott PHP-ről:

```
<html>
  <head> </head>
  <body>
    <?php
      echo" Helló Világ!";
    ?>
  </body>
</html>
```

PHP kódot HTML kódban bárhol elhelyezhetünk. PHP kód kezdetét a következőképpen jelezzük: <?php illetve <? , PHP kód végét pedig ?> -el.

A PHP rendelkezik a többi programozási nyelvhez hasonlóan változókkal, függvényekkel és vezérlési szerkezetekkel. A változóinkat itt nem kell deklarálni, mivel típusát az első értékadás határozza meg, ami a rákövetkező alkalommal akár meg is változhat:

```
$x=2019;
$x="kétezer-tizenkilenc";
echo $x;
```

Az itt írt kódunk kiíratáskor szövegesen jeleníti meg a számot, az-az „kétezer-tizenkilenc” -et látnánk a kijelzőn.

Változóink '\$' jellel kezdődnek, amit egy kis/nagy betűnek kell követnie, ami után tetszőleges betűk, számok és alulvonások követhetnek. Megkötés, hogy számmal nem kezdődhet a változó, valamint a számokon és betűkön kívül egyéb karakterek nem megengedettek.

Terjedelmes függvénykészlettel rendelkezik, azonban lehetőséget biztosít saját függvények deklarálására. A függvény egy zárt kódrészlet, ami meghívásakor fut le és tér vissza a megadott típusú paramétereivel (haadtunk meg visszatérési értéket).

```
function fuggveny ($input) {
// a függvényünk törzse. Ide lehet írni a meghíváskor futtatandó kódot
$output=$input;
return $output;}

```

A „fuggveny(2019);” a fentiek alapján ugyan úgy 2019-el térne vissza.

### 2.2.3 phpMyAdmin

A PhpMyAdmin egy nyílt forráskódú webes felületű kliens a XAMPP-on belüli MySQL adatbázisok kezeléséhez. Egyszerű felületet biztosít az adatbázisunk kezeléséhez. Funkciói közé tartoznak:

- adatbázis készítés vagy eldobás
- táblák készítése és kezelése, törlése
- táblában található mezők szerkesztése, beillesztése, törlése
- SQL parancsok futtatása

## 2.3 Adatbázis-kezelés



### 2.3.1 SQL

Az SQL, azaz Structured Query Language (Strukturált Lekérdezőnyelv) egy számítógépes nyelv a relációs adatbázisok eléréséhez és kezeléséhez. Az adatok lekérdezésére, hozzáadására, frissítésére és módosítására használható. Az SQL parancsok négy alcsoportba oszthatók:

1. adatdefiníció (CREATE, ALTER, DROP)
2. adatkezelő (INSERT, UPDATE, DELETE)
3. adatvezérlő (GRANT/ REVOKE, COMMIT/ ROLLBACK)
4. adatlekérdező (SELECT)

### 2.3.2 RDBMS

Relációs adatbázis-kezelő rendszernek olyan programot nevezünk, amelyik az adatokat táblákban, relációkban tárolja, rendezi és onnan keresi vissza. A relációs adatbázis több összekapcsolt táblából áll. A relációs adatbázis-kezelő rendszer több adattáblát logikailag összekapcsol egymással, és megkeresi bennük a közös információkat. Ahhoz, hogy egy táblát relációnak lehessen tekinteni, a következő feltételeket kell megfelelnie:

- Minden oszlopnak egyedi neve van.
- A sorok és oszlopok sorrendje tetszőleges.
- Nem lehet két egyforma sora.

A relációs adatbázisok általában több logikailag összekapcsolható táblából állnak. A táblák között meghatározott kapcsolat van. A tervezésnél nagyon fontos, hogy ezeket a kapcsolatokat jól építsük fel. A relációs adatbázis felépítésének alapja a normalizálás, amely az adatok optimális elhelyezési módját megadó módszert jelenti.

### 2.3.3 MySQL

A MySQL a világon széles körben használt nyílt forráskódú, relációs adatbázis-kezelő rendszer. Főbb fókuszpontjai a gyorsaság, a platformfüggetlenség és a költséghatékonyság. Nemcsak tárolja adatainkat, hanem kezeli is őket. Lehetőséget biztosít felhasználói jogosultságok beállítására. A legtöbb nyelv, mint pl. a PHP, a C, illetve a C++, a Java alkalmas arra, hogy felhasználói felületet írjanak hozzá. Minden MySQL-nek kiadott parancs az SQL nyelven történik. A PHP esetében rengeteg előre definiált függvény közül válogathatunk, amik a MySQL-lel kommunikálnak.

Adatbázishoz való kapcsolódásra PHP-ból:

```
$dbServername = "localhost";
$dbUsername = "root";
$dbPassword = "";
$dbName = "szakdolgozat";

$conn = mysqli_connect($dbServername, $dbUsername, $dbPassword, $dbName) or die("Nem sikerült csatlakozni az adatbázishoz");
$conn->set_charset("utf8");
$conn->query("SET lc_time_names = 'hu_HU'");
session_start();
```

2.3.3.1. ábra csatlakozás az adatbázishoz php oldalról - db.con.php

Mivel a XAMPP programcsomagot használva lokálisan a gépről fut az adatbázisunk a fenti példánál ezért elérési címnek a localhost-ot adtuk és az alapértelmezett, default felhasználóval csatlakoztunk a „szakdolgozat” adatbázisra. Minden php fájlba includeolni kell a connectiont ahol adatbázisműveletet hajtunk végre. Alap adatbázisműveletek:

### **CREATE TABLE**

Miután létrehoztuk az adatbázisunkat (a CREATE DATABASE adatbázis\_neve; paranccsal) fel kell tölteni azt táblákkal. Fontos, hogy szűrjük a redundáns mezőket és már az összefüggő normalizált tervekkel készítjük el a táblákat, hogy megkönnyítsük a dolgunkat. A táblákat az alábbi módon hozzuk létre:

```
CREATE TABLE emberek (
    id int NOT NULL AUTO_INCREMENT,
    ember_neve varchar(255) NOT NULL,
    ember_kora int DEFAULT 0,
    ....
    PRIMARY KEY (Personid)
);
```

Az alábbi SQL kód létrehozza az emberek táblát a megadott három oszloppal: id, ember\_neve, ember\_kora.

id: Ez a primary kulcsunk a táblában, ami a példában azt jelenti, hogy minden emberhez ez az egyedi hozzárendelt azonosító. Az auto\_increment azt jelenti, hogy az adatbázis magának állítja az azonosítót, azaz új adat beszúrásakor azt nem kell megadnunk.

ember\_neve: ez egy varchar típusú 255karakter hosszú szöveges mező (string)

ember\_kora: ez int típusú, vagyis számokat tárol. A megadott default mező miatt ha nem adunk meg semmit felvitelkor akkor az ember életkora alapértelmezetten nulla lesz (fontos hogy nem NULL, ami üres adatot jelent).

### **INSERT INTO**

Ha már van adatbázisunk és táblánk akkor az INSERT INTO művelettel feltölthetjük azt:

INSERT INTO emberek VALUES („Teszt Elek”, 30);	INSERT INTO emberek (ember_neve, ember_kora) VALUES („Teszt Elek”, 30);
---	--

A két példakód megegyező. A második esetben nem feltétel kell minden mezőre adatot adni, például: INSERT INTO emberek (ember\_neve) VALUES („Teszt Elek”); egy teljesen működő művelet (és mivel a fenti CREATE TABLE példánál lekezeltek a default kor értékét ezért jelen esetben az nulla lenne az adatbázisban)

### **UPDATE**

Ha például az előbb behelyezett „Teszt Elek” életkorát változtatni akarjuk akkor az alábbi utasítás a megfelelő:

UPDATE emberek SET ember_kora = 10 WHERE ember_neve = „Teszt Elek”
--

Így minden felhasználónak, akinek a neve „Teszt Elek” beállítja a korát 10-re. Ha tudjuk a fent megadott emberek táblában a felhasználó id-t akkor az jobb megoldás lenne, mivel az egy egyedi azonosító, ami biztosít minket arról hogy nincs még egy olyan elem senkinél sem, míg a ha nem kezeljük le jól felvételnél az emberek nevét akkor előfordulhat hogy két „Teszt Elek” is lesz amire lefut az UPDATE.

### **DELETE**

Ha adatot akarunk törölni a táblánkból akkor a DELETE műveletet kell használnunk:

DELETE FROM emberek WHERE ember_neve = „Teszt Elek”;
--

Ez a művelet kitörli az összes olyan felhasználót, akire illik a feltétel.

### **SELECT**

A lekérdezés SQL nyelven a következő minta szerint néz ki:

SELECT <elem> FROM <forrás>	SELECT * FROM tábla;
-----------------------------	----------------------

A „SELECT” szót úgy fordíthatnánk, hogy „Szelektáljuk”, ez után jön, hogy mit. Utána egy \* csillagkaraktert látunk. Ez egy adattábla összes mezőjét jelenti. Egy adatbázisban lehet több tábla is, ezért meg kell mondanunk, melyik táblából szeretnénk lekérdezést végrehajtani.

A” szelektáláshoz” lehet szűrőket beállítani a WHERE kulcsszó után. Ha például van egy táblánk név és életkor oszlopokkal, ahonnan a 18 életévét betöltött embereket akarjuk kilistázni akkor az a lekérdezés így nézne ki:

SELECT ember_neve FROM emberek WHERE ember_kor >=18
---

Minden SQL nyelven írt lekérdezés a SELECT utasítást használja, csak a keresési feltételeket módosítva. Ha több táblából akarunk keresni, vagy a feltételünkhöz különálló táblák kellenek akkor azokat össze kell kapcsolni közös adataikkal.

Ha például az előbbi példánkat alapul véve azt akarnánk lekérdezni, hogy a 18 életévét betöltő emberek magasságát akarom megtudni, akkor az így nézne ki:

```
SELECT x .ember_neve, y.magassag FROM emberek as x, magassag as y WHERE
x.ember_kor >=18 AND y.ember_neve =x .ember_neve ORDER BY x.ember_neve DESC
```

Itt az emberek és a magasság táblát kapcsoltuk össze az ember nevével, mivel mindkét táblánkban megtalálható volt ez az elem.

### 3. Tervezés

Ebben a fejezetben határozom meg, hogyan is kellene működnie a rendszernek, milyen szolgáltatásokat kell nyújtania. Meghatározom milyen használati (use-case) esetek fordulhatnak elő az egyes felhasználókkal, milyen funkciókat érhetnek el és milyen módon használhatják a rendszert. Az esetek leírása megkönnyíti a specifikáció második fázisát mivel minden elvárt funkciót ismertetek, amit a megjelenési felületen keresztül a felhasználók elérhetnek. Az itt leírtaknak megfelelő megvalósításokat a későbbi fejezetekben fogom bemutatni.

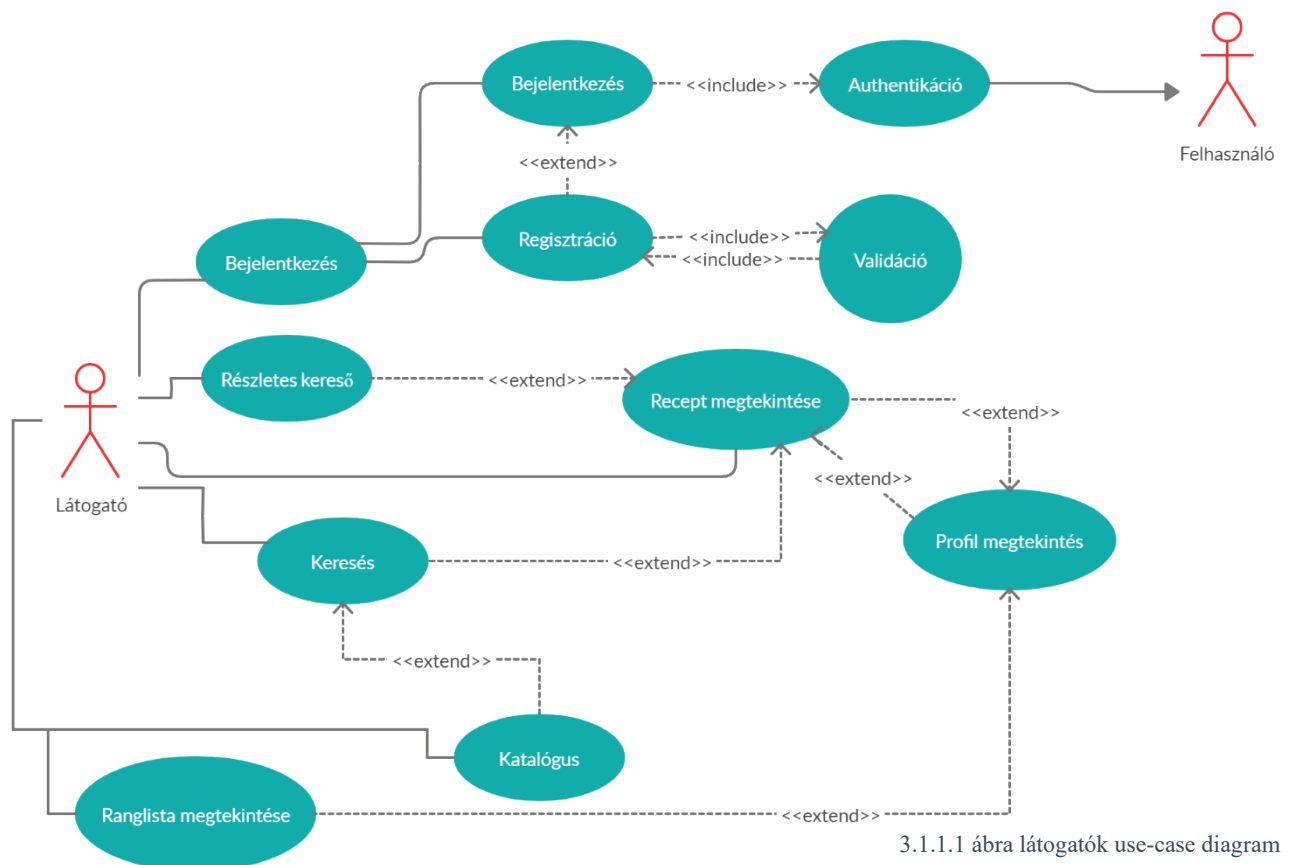
#### 3.1 Az alkalmazás felhasználói

A követelmények alapján az alkalmazás felhasználóit négy csoportba soroltam: látogató felhasználó, admin, adminisztrátor/moderátor. Minden csoport csak a saját jogkörének megfelelő funkciókat érheti el, azonban egy moderátor több szereppel is rendelkezhet, mivel az admin határozza meg melyik felhasználó milyen jogot kaphat. Például egy normál felhasználó megkaphatja meglévő tulajdonságai mellé a hozzászólások moderálása jogot is és így a felhasználói jogkörét bővítve moderátor lesz. Alapesetben minden felhasználónak rendelkeznie kell a normál szerepkörrel.

A programtól elvárt funkcionális követelményeknek megfelelő használati eseteket a lenti ábrákon látható használati eset diagram foglalja össze, illetve a következőkben külön-külön részletezem az egyes szerepkörökkel szemben támasztott követelményeket.

### 3.1.1 Látogató/Visitor szerepkör

- Képes regisztrálni, amivel egy teljeskörű felhasználót hoz létre.
- A bejelentkezéssel be tud lépni a már létező felhasználói fiókjába.
- A főoldalon válogathat a legújabb és legjobb receptajánlatok közül.
- Rendelkezésre áll a katalógus menüpont, ahol kategóriákra bontottan kereshet a meglévő receptek közül.
- A keresés és részletes keresés teljes funkciói rendelkezésre állnak.
- Képes recepteket megtekinteni és onnan az alkotó profiljára látogatni.
- Profilok látogatása és az azok által készített ételek megtekintése.
- Ranglista megtekintése, ahol a legtöbbpontos felhasználók profilját tudja megnézni.

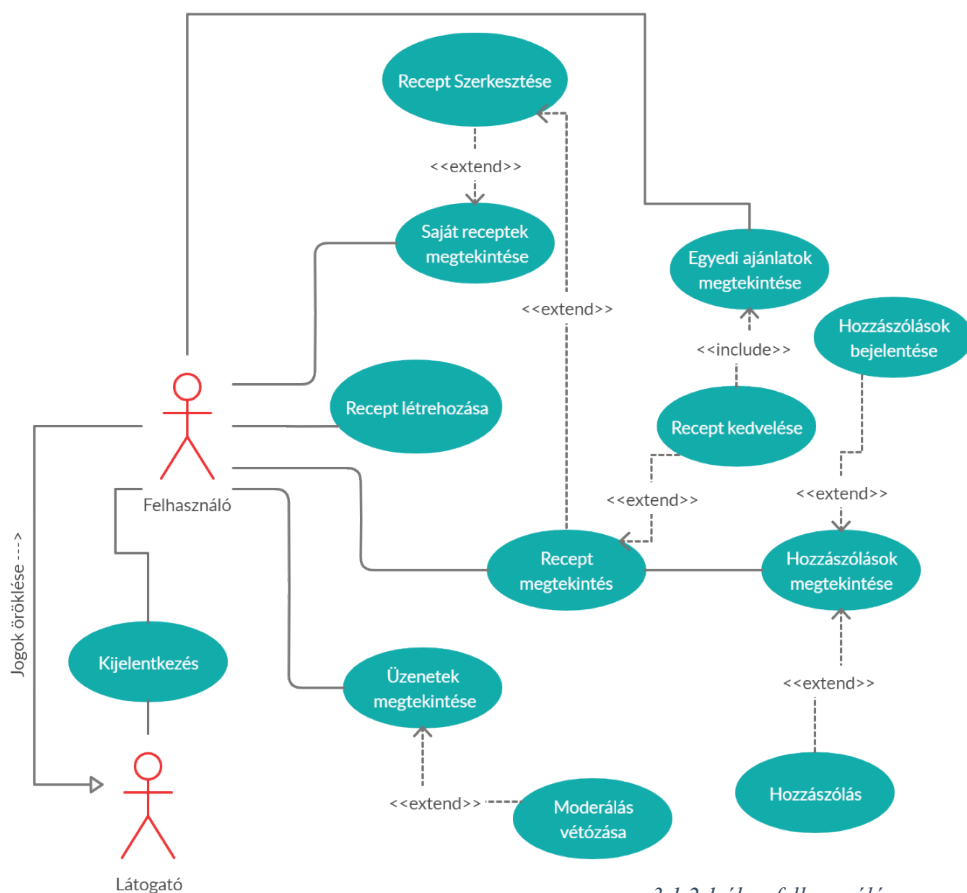


3.1.1.1 ábra látogatók use-case diagram

### 3.1.2 Felhasználó szerepkör

- Magában foglalja a látogató jogait.
- Kijelentkezés opció.
- Rendszerüzenetek megtekintése.
- Receptekhez hozzászólni.
- Hozzászólásokat jelenteni.

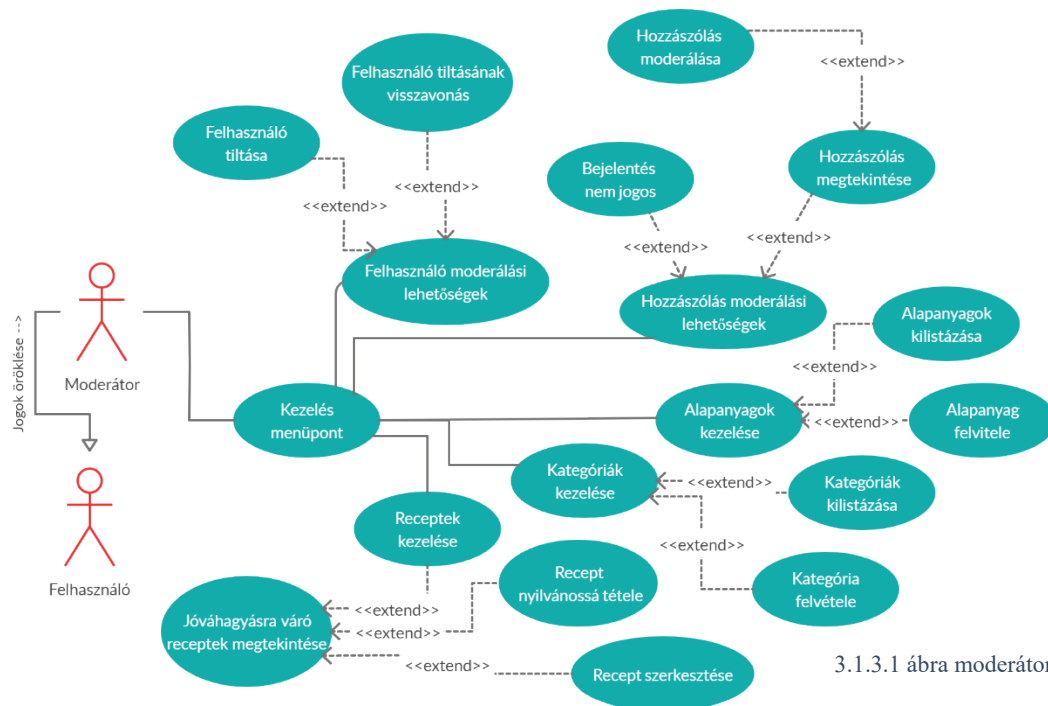
- letiltást/hozzászolás törlését vétózni.
- Receptek kedvelése.
- személyre szabott ajánlatok böngészése
- Recept létrehozása
- Saját recept szerkesztése



3.1.2.1 ábra felhasználó use-case diagram

### 3.1.3 Moderátor szerepkör

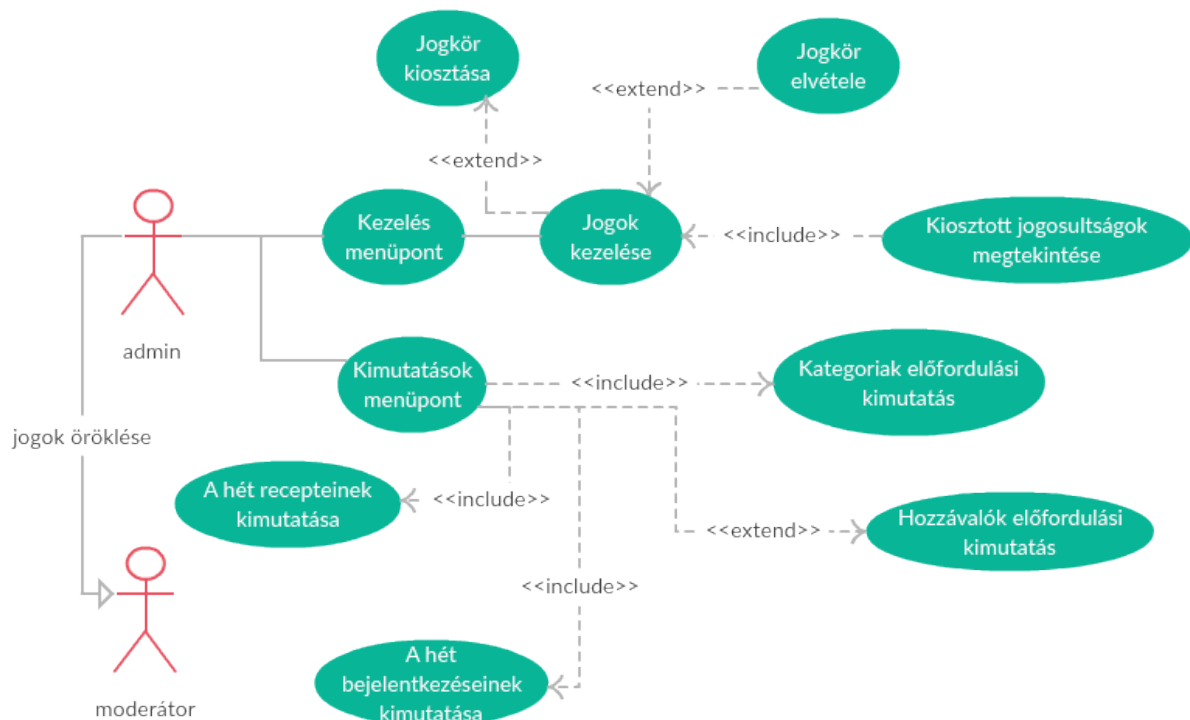
- Magában foglalja a felhasználó jogait.
- Elérhetővé válik a kezelés menüpont
- A moderátori jogaitól függően többféle kezelési opció áll rendelkezésre
- Letilthat felhasználókat és törölhet hozzászólásokat
- Publikussá és priváttá tehet recepteket.
- Alapanyagokat vehet fel.
- Kategóriák felvétele



3.1.3.1 ábra moderátor use-case diagram

### 3.1.4 Admin szerepkör

- Magában foglalja a moderátorok jogait.
- Elérhetővé válik a kimutatások menüpont
- A moderátor kezelési felületen túl moderátori jogokat oszthat ki a felhasználók között.



3.1.4.1 ábra moderátor use-case diagram

## 3.2 Az alkalmazás főbb funkciói

### 3.2.1 Autentikáció és autorizáció

Az oldalra látogató személynek lehetősége van az oldalon saját profilt létrehozni a regisztráció segítségével. Miután a felhasználó már egyszer sikeresen regisztrált, utána lesz lehetősége az alkalmazásba történő bejelentkezésre. A bejelentkezésre pedig azért lehet szüksége, mert az alkalmazásban bizonyos funkciókat csak bejelentkezett felhasználó vehet igénybe.

Minden olyan felhasználó be tud jelentkezni, aki már regisztrált az oldalon. Sikeres bejelentkezéshez meg kell adnia a felhasználónevét (azonosítóját) valamint a hozzá tartozó jelszót. Ezután a 'Belépés' gombra kattintva elküldi az előbb megadott adatokat. Az adatok elküldése a \$\_POST változó segítségével történik.

### 3.2.2 Receptek és hozzászólások

Minden regisztrált és nem regisztrált felhasználónak lehetősége van az oldalra feltöltött és jóváhagyott receptek között böngészni és azokat megtekinteni, azonban a hozzászólások megtekintése és használata csak a teljeskörű, regisztrált felhasználóknak áll rendelkezésére. Azok a felhasználók, akik korábban már tiltva lettek, megtekinthetik a recepteket és a hozzászólásokat, de Ők maguk nem írhatnak oda.

A recept felépítése az elkészítési útmutatóból áll, amit jól láthatóan egységekre bontunk, valamint a hozzávalók és a tápértékek táblázatából.

### 3.2.3 Keresési lehetőségek

Az oldalon minden felhasználónak lehetősége van a receptek között keresni. A kereséshez három alapvető módszert biztosítunk:

- kategóriára való keresés dinamikus tagcloud segítségével
- egyszerű keresőmező a menüsávban, amely a recept nevére keres
- részletes kereső, ami a recept nevére, hozzávalóira és kategóriájára is tud keresni többszörös paramétermegadási lehetőséggel.

A többszűrős keresőmotor azt a célt szolgálja, hogy ha nem tudja a felhasználó mit akar elkészíteni otthon akkor felviszi a keresési paraméterekbe a nála megtalálható hozzávalókat és csökkenő sorrendben kilistázza a találatokat aszerint, hogy mennyi egyezést talált.



### 3.2.4 Toplista és profilok

A toplista, valamint a profilok megtekintése szintén egy mindenki számára elérhető funkció. A toplistán a felhasználók pontjaik összegzéseként vannak sorba rendezve, csökkenő sorrendben. Egy felhasználó összes receptjének az összes pontja beleszámít az értékelésbe. A toplista megnyitásakor kéri le az adatbázisból, tehát mindig a legfrissebb adatokkal dolgozik.

A toplistán és a receptek megtekintésekor a felhasználók neve egy hivatkozás, ami az Ő profiljukra visz. A profilon látható az adott felhasználó pontja és kategóriája. Ha a felhasználó már létrehozott akár egy receptet is (és azt jóváhagyták) akkor az is kilistázódik ott.

### 3.2.5 Saját receptek és profil

Minden regisztrált felhasználó képes megtekinteni a saját profilját. Ott megtalálható az Ő pontszáma és kategóriája. Ha van elfogadott recept az ő nevén akkor az is megjelenik. Amennyiben van moderált hozzászólása, akkor az is megtalálható lesz a profilján, amit csak Ő láthat és a hozzászólások moderálására kijelölt adminisztrátor.

A saját receptek menüpont megtalálható minden regisztrált felhasználónál. Itt kilistázódik az összes általa létrehozott recept időrendi sorrendben csökkenően. Annyiban tér el a profilján található listától, hogy itt a még nem jóváhagyott receptek is megjelennek, valamint itt megtalálható a szerkesztési hivatkozás minden receptnél.

### 3.2.6 Recept szerkesztés

Az általunk létrehozott receptet Mi magunk szerkeszthetjük, valamint az ezzel a jogkörrel rendelkező adminisztrátor az oldalon. A szerkesztésnél minden opción változtathatunk a recepten, amit felvitelekor megadtunk. Ha már egy aktív, jóváhagyott receptet szerkesztünk, akkor azt újra jóvá kell hagynia egy moderátornak. A szerkesztési felületen lehetőség van törölni a receptet.

### 3.2.7 Recept felvitele

Az új recept menüpont a regisztrált és nem tiltott felhasználóknak áll rendelkezésére. Itt a beviteli mezők bizonyos kritériumokon esnek át, amik biztosítanak minket a helyes adatokról. A felhasználó a megadott alapanyagokból és kategóriákból rakhatnak össze egy receptet. A kimaradt vagy nem talált hozzávalókat / kategóriákat az erre a célra fenntartott egyéb mezőbe kell írni, amit jóváhagyás előtt felvesz az erre a feladatra kijelölt adminisztrátor és rögzíti a receptben.

### 3.2.7 Moderálás

A fenti funkciókat egy oldalon tartják működésben az oldal üzemeltetői, a kezelés menüponttal. Itt minden emberi beavatkozást / megítélést igénylő folyamathoz hozzáfér a kellő hatáskörrel rendelkező felhasználó.

A moderálási feladatokat több szekcióra bontottuk és azokhoz jogot hoztunk létre. Security szempontból elengedhetetlen, hogy fölöslegesen ne adjunk ki jogokat, amiket nem használ az adott személy.

### 3.2.8 Vétózás

Ha a felhasználót letiltják vagy moderálják egy hozzászólását akkor a rendszerüzenet mellett megjelenik egy gombra, amivel vétózni tudja a műveletet. Ilyenkor egy folyamat indul el amit egy erre a feladatra kijelölt és adott jogkörrel rendelkező moderátor visz végig és dönt.

### 3.2.9 Üzenetek

A felhasználó értesítéseket kap minden fontosabb rendszerfolyamatról, ami érinti őt. ez az üzeneteknél tudja megtekinteni. A menüben egy jelző mutatja mennyi olvasatlan üzenete van a fiókjában.

## 3.3 Adatbázisterv

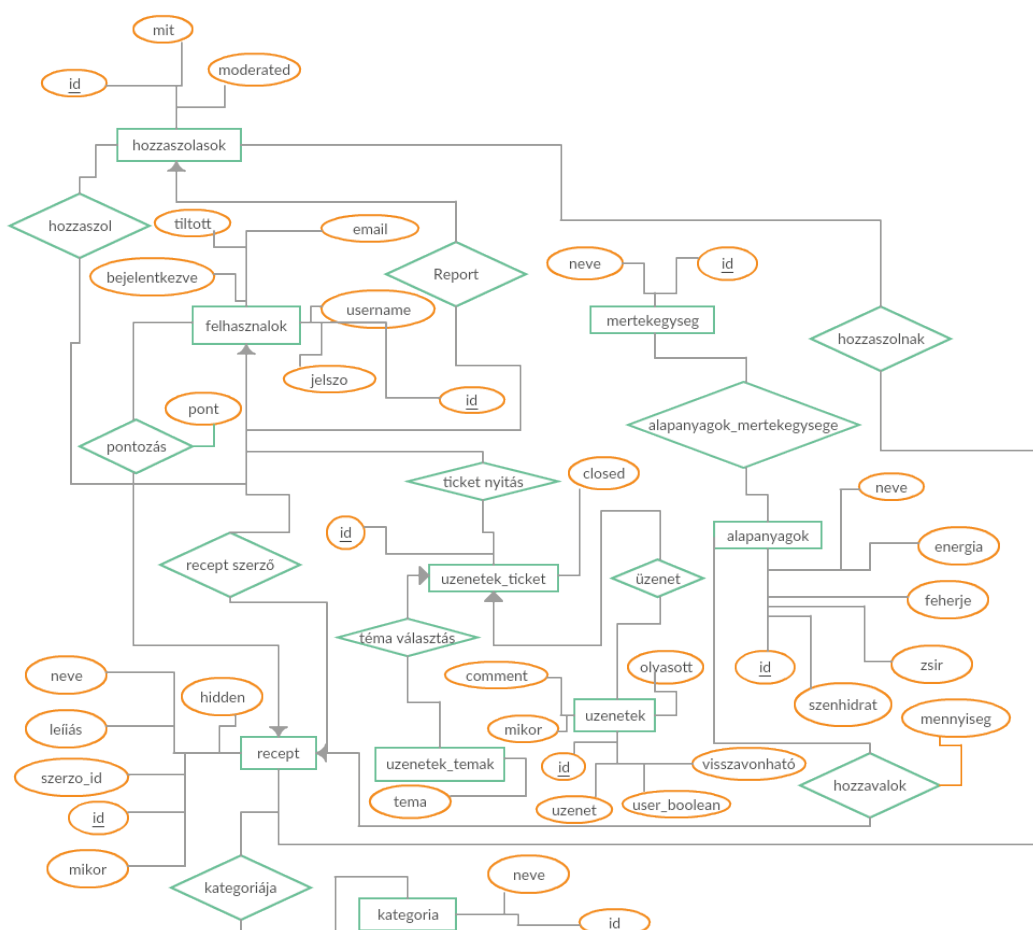
### 3.3.1 Szükséges táblák meghatározása

- Feltétlenül szükség lesz a **felhasználók** tárolásához egy felhasználók táblára.
- A **receptek** tárolásához is kell egy külön tábla, ahol a recept szerzője egy külső kulcs a felhasználók táblának az azonosító mezőjével összekapcsolva.
- A receptekhez tervezünk hozzászólásokat, tehát egy **hozzászólások** tábla is kell, ami a recept azonosítójával lesz összekapcsolva.
- A recepteknek van hozzávalója és kategóriája ezért kell **kategória** és **alapanyagok** tábla is. Fontos, hogy egy recepthez ezekből több is lehet.
- Ha vannak alapanyagain akkor abból kell bizonyos mennyiséget tárolnunk az egyes receptekhez. Ahhoz, hogy meghatározzuk milyen mértékegységgel mérünk kell egy **mértékegység** tábla.
- A mértékegységek táblát összekapcsoljuk a hozzávalókkal és egy külön táblába írjuk, hogy egy recepthez miből mennyi kell. tehát kell egy **hozzávalók** tábla.
- A recepteket értékelni is akarjuk ezért kell egy gyűjtőtábla a recept azonosító és a felhasználó id-t külső kulcsként megadva egy pont mezővel kiegészítve létre kell hoznunk a **pontozás** táblát
- A felhasználóknak rendszerüzeneteket akarunk küldeni ezért ezzel is számolnunk kell tervezéskor. Ha több témában akarunk nekik írni akkor kell egy **üzenet témája** tábla.
- ha van az üzenetnek témája akkor ahhoz a témához nyithatunk egy új beszélgetést, ahol eltároljuk a felhasználót, a dátumot, a témát. Tehát kell egy **beszélgetés** tábla

- már mindenünk megvan az üzenetek lekezeléséhez, csak azokat nem tudjuk tárolni. Kell egy **üzenetek** tábla ahol eltároljuk szövegesen a tényleges üzenetet, majd azt egy beszélgetés azonosítóhoz kötjük, így kiszelektálhatjuk a különböző beszélgetéseket.

### 3.3.2 ER modell

Az ER, azaz Entity-Relationship modell három alapelemből áll: egyed (entity), kapcsolat (relation), és az egyedek tulajdonságai (attributes). Népszerűségét annak köszönheti, hogy egyszerű, átlátható, könnyen megérthető, mégis jól szemlélteti az adatszerkezetet, és ezen felül könnyen átalakítható relációs adatmodellé. A modellben téglalappal jelöljük, az egyed nevét. Az egyednek lehetnek tulajdonságai, ezek jellemzik az egyedet. Jelölése ellipszis, benne a tulajdonság neve. A kulcs tulajdonság aláhúzva, ez az egyed egyértelmű azonosítója. Az egyedek között kapcsolatokat is le tudjuk írni a modellel. Jelölésük rombuszsal és azokból induló nyilakkal történik, a nyíl típusa függ a kapcsolat jellegétől. A kapcsolatok típusai a következők:



#### 3.3.2.1 entity relationship diagram

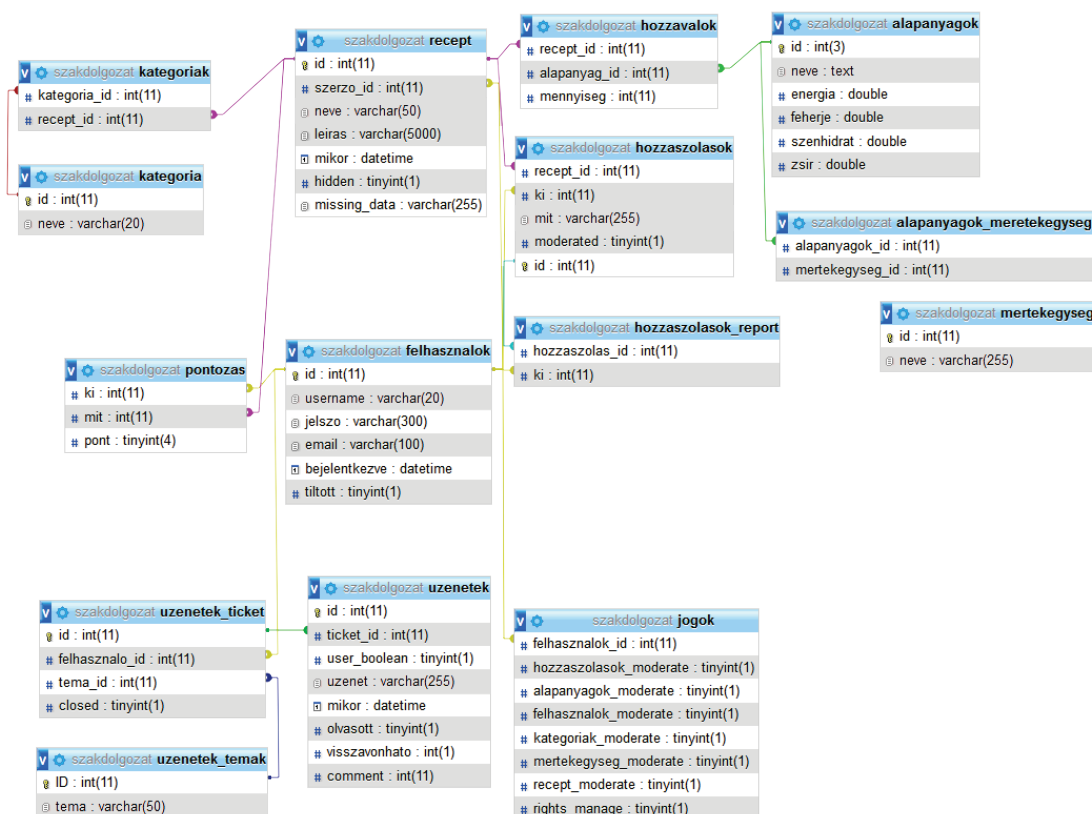
- Az 1:1, azaz egy az egyhez kapcsolatnál egy egyed előforduláshoz csak egy másik egyed előfordulás tartozhat, és ugyanígy fordítva. Jelölése rombuszból, és abból mindkét egyed felé kiinduló szimpla végű nyílból áll.

- Az 1:N, azaz egy a többhöz kapcsolatban egy X egyed előforduláshoz több Y egyed előfordulás is tartozhat, viszont minden Y egyed előforduláshoz csak egy X egyed előfordulás tartozik. Jelölése a rombuszból, az X egyed felé mutató szimpla végű, a Y egyed felé pedig dupla végű nyíllal történik.
- Az N:M, azaz több a többhöz kapcsolatban X és Y egyed előforduláshoz is több tartozhat. Jelölése a rombuszból kiinduló, mindkét irányba mutató dupla végű nyíl.

A kapcsolatoknak is lehet nevük és tulajdonságaik.

### 3.3.2 Relációs adatmodell

Ebben a modellben az adatokat táblákba szervezzük, és a kapcsolat kialakítása egy közös mező tárolásával történik. A modellt 1970-ben mutatta be E.F Codd, és azóta ez a legszélesebb körben használt adatbázis-modell. A relációs modell adatainak alapvető szerkezetei a táblázatok. Az adott típushoz tartozó összes információt az adott táblázat soraiban tárolja. Ezért a táblákat relációknak is nevezzük. Egyszerűségének, könnyen tanulhatóságának és rugalmasságának is köszönheti népszerűségét.



## 4. Implementáció

### 4.1 Adatbázis implementációja

Az adatbázis implementációjánál segítségünkre van a XAMPP egyik grafikus felületű adatbázis-kezelő modulja, a phpMyAdmin. A phpMyAdmin grafikus felületének köszönhetően lehetőségünk van űrlapok segítségével létrehozni az egész adatbázist. Azonban az SQL parancsok nagyobb átláthatóságot és ráhatást biztosítanak. Ha kézzel szeretnénk az SQL parancsokat megadni, a phpMyAdmin automatikus tartalom kiegészítéssel segíti a munkánkat.

Mint ahogy egy korábbi fejezetben kifejtettem, az általam alkalmazott programozási nyelvek, eszközök, technikák a MySQL, PHP, JavaScript (jQuery), CSS (Bootstrap) és a HTML lesznek. Mivel nagyon rugalmasan együtt tudnak működni, ezért viszonylag egyszerű dolgom volt velük, ugyanis a különböző PHP fájlokban el tudtam helyezni minden technológiát.

#### 4.1.1. Táblák létrehozása

A táblák létrehozásakor mindig azokat kell elsőnek hozzáadni, amelyekben nem szerepel más tábla elsődleges kulcsa, azaz nincs idegen kulcs mezője. A felhasználók, a alapanyagok, uzenetek\_temak, kategoriak és amértékegyseg tábláknak nincs idegen kulcsa, így ezeket adjuk hozzá először az adatbázishoz. Minden tábla azonosítója, azaz elsődleges kulcsa AUTO\_INCREMENT funkcióval lesz ellátva, ami azt jelenti, hogy az azonosító értéke mindig automatikusan eggyel nő a rekord hozzáadásakor az előző rekord azonosítójához képest. A létrehozást SQL kód formájában szoktam, mivel így nagyobb átláthatóságot érzek mint a felületen erre a célra készített opcióknál. A felhasználók tábla létrehozása:

```
CREATE TABLE felhasznalok (  
  id int NOT NULL AUTO_INCREMENT,  
  username varchar(20) NOT NULL,  
  jelszo varchar(300) NOT NULL,  
  email varchar(100) NOT NULL,  
  bejelentkezve datetime,  
  tiltott tinyint(1),  
  
  PRIMARY KEY (id)  
);
```

### 4.1.2. Idegen kulcs és további megszorítások

Ha már csak az idegen kulccsal rendelkező táblák maradtak akkor kezdjük azokat is létrehozni hasonló módon:

```
CREATE TABLE recept (
  id int NOT NULL AUTO_INCREMENT,
  szerzo_id varchar(20) NOT NULL,
  neve varchar(50) NOT NULL,
  leiras varchar(5000) NOT NULL,
  mikor datetime,
  hidden tinyint(1),

  PRIMARY KEY (id),
  FOREIGN KEY (szerzo_id) REFERENCES felhasznalok(id)
);
```

### 4.1.3. felhasznalok

A felhasznalok táblában található a regisztrált felhasználók.

id : int(11)
username : varchar(20)
jelszo : varchar(300)
email : varchar(100)
bejelentkezve : datetime
tiltott : tinyint(1)

4.2.1.1 ábra felhasznalok tábla

Az itt található mezők:

- id – ez az elsődleges kulcs a felhasználóknak, auto increment
- username – itt tároljuk a megadott felhasználónevet
- jelszo – ide kerül hashelés után a jelszó regisztráláskor
- email – itt tároljuk a megadott emailcímet
- bejelentkezve – minden bejelentkezéskor frissülő dátum
- tiltott – itt tároljuk hogy a felhasználó rendelkezik-e

megszorításokkal. Ha '1' akkor igen ha '0' akkor nem.

### 4.1.4. recept

A recept táblában a receptek adatait tároljuk. A kategóriákon és a hozzávalókon kívül minden adat itt van ami megjelenítésre releváns.

id : int(11)
# szerzo_id : int(11)
neve : varchar(50)
leiras : varchar(5000)
mikor : datetime
# hidden : tinyint(1)
missing_data : varchar(255)

4.2.2.1 ábra recept tábla

Az itt található mezők:

- id – ez az elsődleges kulcs a recepteknek, auto increment
- szerzo\_id – ez egy idegen kulcs a felhasznalok.id mezőjére hivatkozik.
- neve – a recept neve
- leiras – a recept leírása
- mikor – a recept létrehozásának ideje

- hidden – ez egy boolean adat. Itt tároljuk hogy a recept nyilvános-e vagy még jóváhagyásra vár.
- missing\_data – itt tároljuk a recept létrehozásakor, vagy szerkesztésekor hiányolt választási opciókat a hozzávalók és kategóriák terén, amit a moderátorok tudnak javítani.

#### 4.1.5. pontozas



szakdolgozat pontozas
# ki : int(11)
# mit : int(11)
# pont : tinyint(4)

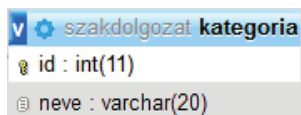
4.2.3.1 ábra pontozas tábla

Egyszerű összekötő tábla ami a recepteket és a felhasználókat köti össze és egy pontozás mező segítségével rögzítjük a felhasználók értékelését.

Az itt található mezők:

- ki – idegen kulcs a felhasznalok.id-re
- mit – idegen kulcs a recept.id-re
- pont – a felhasználó értékelését rögzítjük itt

#### 4.1.6. categoria



szakdolgozat categoria
id : int(11)
neve : varchar(20)

Az itt található mezők:

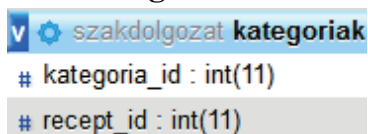
4.2.4.1 ábra categoria tábla

A jogkörrel rendelkező moderátorok által felvett kategóriák ide kerülnek rögzítésre.

id – elsődleges kulcs, auto increment

neve – categoria neve

#### 4.1.7 kategoriak



szakdolgozat kategoriak
# categoria_id : int(11)
# recept_id : int(11)

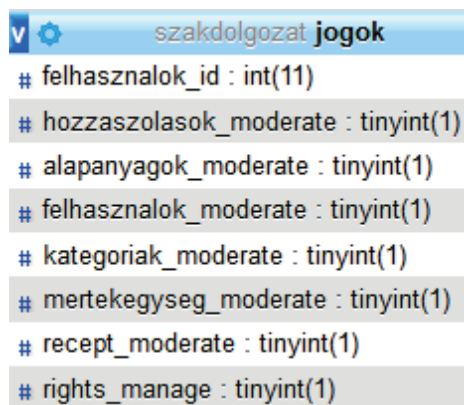
4.2.5.1 ábra kategoriak tábla

Összekötő tábla a recept és categoria között. Itt kapcsoljuk össze a recepthez fűződő kategóriákat.

Az itt található mezők:

- kategoriak\_id – idegen kulcs a categoria.id- re
- recept\_id – idegen kulcs a recept.id-re

### 4.1.8 jogok



# felhasználok_id : int(11)
# hozzaszolasok_moderate : tinyint(1)
# alapanyagok_moderate : tinyint(1)
# felhasználok_moderate : tinyint(1)
# kategoriak_moderate : tinyint(1)
# meritekegyseg_moderate : tinyint(1)
# recept_moderate : tinyint(1)
# rights_manage : tinyint(1)

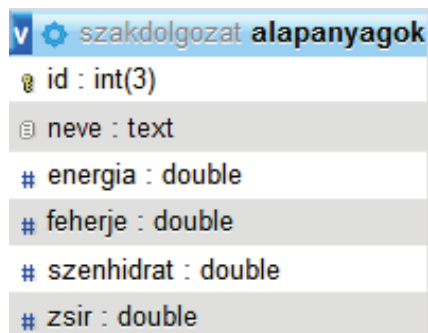
4.2.6.1 ábra jogok tábla

Az alábbi tábla határozza meg ki férhet hozzá a kezelés menüponthoz és ,hogy mit láthat ott. A jogok táblát az admin felhasználó üzemelteti a siteon.

Az itt található mezők:

- felhasználó\_id – külső kulcs a felhasználok.id-re
- hozzaszolasok\_moderate – boolean érték. Azt tároljuk el itt, hogy a felhasználó képes e a hozzászólásokat moderálni.
- alapanyagok\_moderate – boolean érték. Azt tároljuk el itt, hogy a felhasználó képes e a alapanyagokat felvenni.
- felhasználok\_moderate – boolean érték. Azt tároljuk el itt, hogy a felhasználó képes e a másokat letiltani, tiltást feloldani.
- kategoriak\_moderate – boolean érték. Azt tároljuk el itt, hogy a felhasználó képes e a kategóriákat felvenni.
- meritekegyseg\_moderate – boolean érték. Azt tároljuk el itt, hogy a felhasználó képes e a mértékegységeket felvenni.
- recept\_moderate – boolean érték. Azt tároljuk el itt, hogy a felhasználó képes e a recepteket jóváhagyni, levenni, feltenni, szerkeszteni.
- rights\_moderate – boolean érték. Csak az adminnak van ez a jog fenntartva. Ezzel a jogkörrel rendelkező felhasználó képes másoktól jogot elvenni és adni is.

### 4.1.9 alapanyagok



id : int(3)
neve : text
# energia : double
# feherje : double
# szenhidrat : double
# zsir : double

4.2.7.1 ábra alapanyagok tábla

Ide kerül rögzítésre az adott jogkörrel rendelkező adminisztrátor által felvett alapanyag.

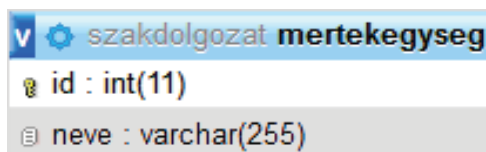
Az itt található mezők:

- id – elsődleges kulcs, auto increment
- neve – az alapanyag neve
- energia – az alapanyagban található energia 1 grammra számítva
- feherje – az alapanyagban található fehérje 1 grammra számítva
- szenhidrat – az alapanyagban található szénhidrát 1 grammra számítva



- zsir – az alapanyagban található zsír 1 grammra számítva

#### 4.1.10. mertekegyseg



```

szakdolgozat mertekegyseg
id : int(11)
neve : varchar(255)

```

4.2.8.1 ábra mertekegyseg tábla

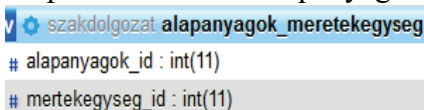
Itt lényegében azt rögzítjük jelenleg, hogy az alapanyag folyékony vagy szilárd, esetleg darabra mérjük. Fejleszthetőség szempontjából külön táblázatban tároljuk el ezt. Jelenleg három adat található itt: 'g', 'ml', 'db'.

Az itt található mezők:

- id – elsődleges kulcs, auto increment
- neve – a mértékegység neve

#### 4.1.11. alapanyagok\_mertekegyseg

Kapcsolattábla. Az alapanyagot köti össze a mértékegységgel. .



```

szakdolgozat alapanyagok_mertekegyseg
# alapanyag_id : int(11)
# mertekegyseg_id : int(11)

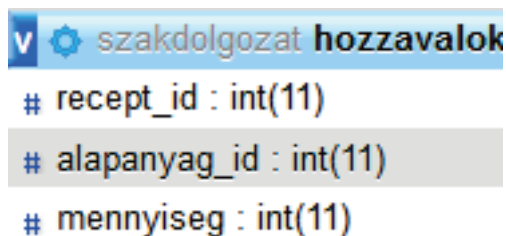
```

4.2.9.1 ábra alapanyagok\_mertekegyseg tábla

Az itt található mezők:

- alapanyag\_id – idegen kulcs az alapanyagok.id-re
- mertekegyseg\_id idegen kulcs amertekegyseg.id-re

#### 4.1.12. hozzavalok



```

szakdolgozat hozzavalok
# recept_id : int(11)
# alapanyag_id : int(11)
# mennyiseg : int(11)

```

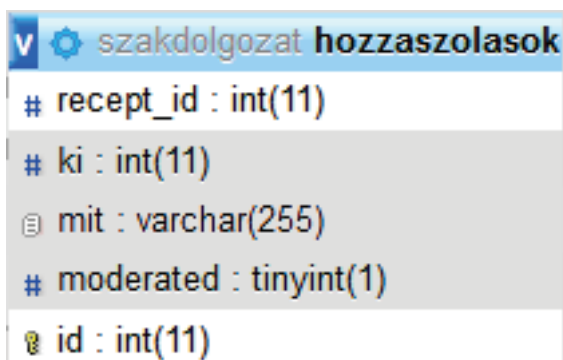
4.2.10.1 ábra hozzavalok tábla

Ide rögzítjük, hogy miből mennyi kell egy receptbe. Mivel számításokhoz a legcélszerűbb volt a legkisebb mértékegységben eltárolni, ezért nem került a mertekegyseg táblába több adat (azokat a site-on alakítjuk át).

Az itt található mezők:

- recept\_id – idegen kulcs a recept.id-ra
- alapanyag\_id – idegen kulcs az alapanyag.id-ra
- mennyiseg – egy szám. Ennyi gramm/ml/darab található az adott alapanyagból a receptben

#### 4.1.13. hozzaszolasok



szakdolgozat hozzaszolasok
# recept_id : int(11)
# ki : int(11)
# mit : varchar(255)
# moderated : tinyint(1)
# id : int(11)

4.2.11.1 ábra hozzaszolasok tábla

Itt tároljuk a receptekhez hozzáfűzendő hozzászólásokat.

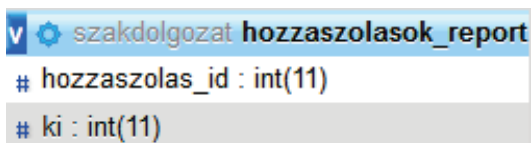
Az itt található mezők:

- recept\_id – idegen kulcs a recept.id-re
- ki – idegen kulcs a felhasznalok.id-re
- mit – a hozzászólás
- moderated – boolean, ez határozza meg, hogy a hozzászólás lett-e moderálva. ha

moderáltak egy hozzászólást akkor az nem jelenik meg a recepteknél.

- id – elsődleges kulcs (tervezéskor ez csak egy gyűjtőtábla lett volna, de később elengedhetlenné vált a pontos hivatkozás a hozzászólásokra, így lett ez később hozzáadva), auto increment

#### 4.1.14. hozzaszolasok\_report



szakdolgozat hozzaszolasok_report
# hozzaszolas_id : int(11)
# ki : int(11)


4.2.12.1 ábra hozzaszolasok\_report tábla

Gyűjtőtábla. Összefűzi a hozzászólásokat és a felhasználókat. ha valaki bejelent egy hozzászólást akkor ide kerül a bejelentő és a bejelentett hozzászólás.

Az itt található mezők:

- hozzaszolas\_id – idegen kulcs a hozzaszolasok.id-re
- ki – idegen kulcs a felhasznalok.id-re

#### 4.1.15. uzenetek\_temak



szakdolgozat uzenetek_temak
# ID : int(11)
# tema : varchar(50)

4.2.13.1 ábra uzenetek\_temak tábla

A rendszerüzenetek témái vannak ide rögzítve. Jelenleg három darab van : rendszerüzenet, hozzászólás moderálás, tiltás.

Az itt található mezők:

- id – elsődleges kulcs, auto increment
- tema – a téma neve

#### 4.1.16. uzenetek\_ticket

id	felhasznalo_id	tema_id	closed
id : int(11)	# felhasznalo_id : int(11)	# tema_id : int(11)	# closed : tinyint(1)

4.2.14.1 ábra uzenetek\_ticket tábla

Alapértelmezetten mindenkinek nyílik egy ticket az első bejelentkezésakor, ez egy rendszerüzenet témájú ticket ami arra a célra szolgál hogy üdvözzölje a felhasználót a weboldalon. A rendszerüzenet témájú ticketet nem zárjuk, de a többi a probléma megoldásával igen.

Az itt található mezők:

- id – elsődleges kulcs, auto increment
- felhasznalo\_id – idegen kulcs a felhasznalo.id-re
- tema\_id – idegen kulcs az uzenetek\_tema.id-re
- closed – boolean, azt mutatja hogy megoldódott e már a ticket (felhasználó tiltása vagy hozzászólásának moderálása vétója kapcsán nyitott jegyek esetén)

#### 4.1.17. uzenetek

id	ticket_id	user_boolean	uzenet	mikor	olvasott	visszavonhato
id : int(11)	# ticket_id : int(11)	# user_boolean : tinyint(1)	uzenet : varchar(255)	mikor : datetime	# olvasott : tinyint(1)	# visszavonhato : int(1)
						# comment : int(11)

4.2.15.1 ábra uzenetek tábla

A tényleges üzeneteket itt tároljuk. Minden üzenet egy adott jegyhez van kötve, így kiíratáskor tudjuk hogy csak a kívánt beszélgetést adjuk vissza.

Az itt található mezők:

- id – elsődleges kulcs, auto increment
- ticket\_id – idegen kulcs az uzenetek\_ticket.id-ra
- user\_boolean – boolean. Mivel a beszélgetés két oldali is lehet ezért így különböztetjük meg ki írta az adott üzenetet.

Ha ez a mező 1 (az-az true) akkor az ticketetben (jegyben) érintett felhasználó írta azt.  
Ha a mező 0 (az-az false) akkor az rendszerüzenet.

- mikor – a küldéskor rögzített idő
- olvasott – boolean. azt mutatja, hogy látta-e már a felhasználó
- visszavonhato – ha itt nullánál nagyobb szám szerepel akkor a felhasználónak van lehetősége vétózni a rendszerüzenetben foglalt műveletet. A szám itt a ticket témája.
- comment – ez egy feldolgozási folyamatot megkönnyítő mező. itt tároljuk a hozzászólás azonosítóját ha az üzenet témája a hozzászólás moderálása.

## 4.2. Főbb funkciók megvalósítása

Mivel huszonhárom php fájlból áll a projektem, ezért csak az előzőekben megírt főbb funkciókat mutatom be ebben a szekcióban. Az ismétlések elkerülése miatt a hasonló megoldással működő funkciókat az első leírásánál és bemutatásánál felsorolás jellegem említem.

### 4.2.1. db.con.php

Ahogy a fájl neve is sugallja itt valósítottam meg a kapcsolatot az adatbázissal. Mivel az adatbázisomat a XAMPP programcsomag keretein belül készítettem ezért a szerver a gépen fut, így a localhost-tal el lehet érni azt (helyben fut) .

```
$dbServername = "localhost";
$dbUsername = "root";
$dbPassword = "";
$dbName = "szakdolgozat";

$conn = mysqli_connect($dbServername, $dbUsername, $dbPassword, $dbName) or
die("Nem sikerült csatlakozni az adatbázishoz");
$conn->set_charset("utf8");
$conn->query("SET lc_time_names = 'hu_HU'");
session_start();
```

A szakdolgozat adatbázishoz csatlakozunk az alapértelmezett fiókkal. Ezt a fájlt include-olom az-az meghívom a menu.php-ba.

### 4.2.2. menu.php

Az adatbázis csatlakozást megvalósító db.con.php fájlt itt hívjuk meg.

```
<?php
include_once 'include/db.con.php'
?>
```

Az előző fejezetben említett bootstrap style-t is itt hívjuk meg, valamint az egyéni stylet itt állítom be.

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<style> ...
```

Mint ahogy a fájl neve is sugallja, itt valósítjuk meg a menüt. A menüben, amit jobban taglalnék az az egyes menüpontok láthatóságának vizsgálata.

```
<li><?php if (isset($_SESSION["user"])) {
    echo "<a href='sajat.php'>Recepteim</a>";
} ?></li>
```

Megnézzük, hogy a session-ben van-e user beállítva. Amennyiben van user az azt jelenti, hogy regisztrált felhasználóról beszélünk, tehát látható lesz. (a session-ről bővebben majd a login.php-ban lesz szó).

Vannak olyan menüpontok, ahol bonyolultabb műveleteket alkalmazunk:

```
<li><?php if (isset($_SESSION["user"])) {

    $felhasznalo = $_SESSION["id"];
    $sql = "select count(*) as db FROM uzenetek,uzenetek_ticket where
uzenetek_ticket.id=uzenetek.ticket_id and user_boolean !=1 and olvasott!=1 and
felhasznalo_id=$felhasznalo ";
    $result = $conn->query($sql);
    $ertesites = array();

    if ($result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {

            $ertesites = array_merge($ertesites, array('db' => $row["db"]));

        }
    } else {
        $ertesites["db"] = 0;
    }
    echo "<a href='uzenetek.php'>Üzenetek";
    if ($ertesites["db"] > 0) {
        echo "<span class='badge badge-primary badge-pill'>" . $ertesites["db"] . "
</span>";
    }
    echo "</a>";
} ?></li>
```

itt megvizsgáljuk, hogy az adott felhasználónak van-e olvasatlan üzenete. Amit a lekérdezésünk ad értéket, azt egy badge-be tesszük, ami a felületen jelezni fogja a felhasználó felé, mennyi értesítése jött. A menu.php-t minden fájlban meghívjuk, ezzel továbbadva az adatbázis és a bootstrap style-t egyaránt.

### 4.2.3. Autentikáció és autorizáció

A funkció a login.php-ban van megvalósítva. Regisztrációkor és bejelentkezéskor is inputszűrést használunk, hogy lecsökkentsük a támadási felületet.

```
$username = test_input($_POST["username"]);
$email = test_input($_POST["email"]);
$password = test_input($_POST["password"]);
$hash = hash('sha256',$password);
```

A POST metódust használva adjuk át az adatokat a test\_input függvényünknek, ami a következő képen néz ki:

```
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
```

A trim függvény a fehér szóközök levágását végzi (az-az a fölösleges szóközöket a szöveg mindkét oldalán). A stripslashes leveszi az idézőjeleket. A htmlspecialchars speciális karaktereket HTML megfelelőire konvertálja át.

Regisztrációnál még meg kell néznünk, hogy a jelszó elég biztonságos:

```
$supercase = preg_match('@[A-Z]@', $password);
$lowercase = preg_match('@[a-z]@', $password);
$number = preg_match('@[0-9]@', $password);
$specialChars = preg_match('@[^\w]@', $password);
$specialChars2 = preg_match('@[^\w]@', $username);

if(!$supercase || !$lowercase || !$number || !$specialChars || strlen($password) < 8) {
    echo '<script language="javascript"> alert("A jelszónak legalább 8 karakter hosszúnak kell lennie kis és nagy betűkkel, valamint legalább egy speciális karakterrel");</script>';
}
```

Ha minden feltételnek megfelel és a felhasználónév még nem létezik, akkor felvesszük a felhasználót:

```
$sql = "INSERT INTO felhasznalok (username, jelszo, email)
VALUES ('$username', '$hash', '$email')"; ...
if ($conn->query($sql) === TRUE) {
    $regisztralt_id=0;
    $sql = "select id from felhasznalok where username='$username' and '$email'=email";
    $x = $conn->query($sql);
    if ($x->num_rows > 0) {
        while($row = $x->fetch_assoc()) {
```

```

$regisztralt_id=$row['id'];
$sq2 = "insert into uzenetek_ticket (felhasznalo_id) values ($regisztralt_id)";
$y = $conn->query($sq2); }
$sq2 = "select id from uzenetek_ticket where felhasznalo_id=$regisztralt_id and
tema_id=0";
$y = $conn->query($sq2);
if ($y ->num_rows > 0) {
    while($sor = $y ->fetch_assoc()) {
        $ticket_id=$sor['id'];
        $ido=date("Y-m-d H:i:s");
        $sq3 = "insert into uzenetek (ticket_id,user_boolean,uzenet,mikor) values
($ticket_id,0,'Üdv a ChefBot-on!', '$ido')";
        $z = $conn->query($sq3); } } }
echo '<script language="javascript">';
echo 'alert("Sikeres regisztráció!");';
echo '</script>'; }

```

Mint ahogy a fenti kód mutatja, a felhasználó felvitele után rögtön nyitunk egy üzenet ticketet és kiküldünk neki egy rendszerüzenetet, amiben üdvözljük a weboldalon.

Bejelentkezéskor is hasheljük a bevitt kódot, mivel mi eleve azt tároljuk az adatbázisban biztonsági okokból. Amikor a felhasználó bejelentkezik, akkor az adatait összevetjük az adatbázisban tároltakkal és ha minden rendben van akkor a session-be felvisszük az adatokat:

```

$_SESSION["tiltott"] = $row["tiltott"];
$_SESSION["user"] = $row["username"];
$_SESSION["id"] = $row["id"];

```

A session a böngészőbe menti el ezeket az adatokat, így ha másik oldalra lép a felhasználó akkor is tudjuk azonosítani. Fontos hogy kijelentkezéskor a sessiont törölnünk kell:

```

session_start();
session_unset();
session_destroy();

```

#### 4.2.4. Receptek és hozzászólások

A receptek megjelenítésénél egyszerű a dolgunk, mivel az adatbázisunkat úgy terveztük, hogy könnyen azonosítható és egybeköthető legyen minden adat a megadott receptnél. Két sql lekérdezésben megkapjuk a kellő adatokat:

```

$id = $_GET['id'];
$sql = "SELECT recept.id,recept.szerzo_id,felhasznalok.username as
felhasznalo,recept.neve,recept.leiras,recept.mikor,recept.hidden as hidden,
TRUNCATE(sum((energia*mennyiseg)),2)as
energia,TRUNCATE(sum((feherje*mennyiseg)),2)as
feherje,TRUNCATE(sum((szenhidrat*mennyiseg)),2)as szenhidrat
,TRUNCATE(sum((zsir*mennyiseg)),2)as zsir,missing_data as egyeb FROM hozzavalok,
alapanyagok, recept,felhasznalok where recept_id=$id and alapanyagok.id=alapanyag_id
and recept_id=recept.id and felhasznalok.id=recept.szerzo_id";
$result = $conn->query($sql);

```

Az első sql lekérdezésben megkapjuk a kért recept nevét, szerzőjét, leírását, dátumát és

Energia	Fehérje	Zsír	Szénhidrát
2215.14	163.92	159.24	142.88

Hozzávaló	Mennyiség
erős paprika	1 db
szalonna (füstölt)	150 g
kolbász	120 g
vöröshagyma	500 g
paprika	2000 g
paradicsom	1000 g

#### 4.2.4.1 recept tápértéke és hozzávalók listája

kiszámoljuk a teljes tápértéket. Ha változtatunk a recepten, vagy az alapanyagok tápértékén akkor az itt kapott adatok is változnak, mivel a megnyitás pillanatakor értékelőik ki. Az említett adatok mellé szükséges még a hozzávalók listája is amit egy másik sql lekérdezés keretein belül valósítottam meg:

```

$sql = "select alapanyagok.neve as alapanyag,hozzavalok.mennyiseg as mennyi,
mertekegyseg.id as mertekegyseg_id,mertekegyseg.neve as mertekegyseg_neve from
hozzavalok, alapanyagok, alapanyagok_mertekegyseg,mertekegyseg where
alapanyag_id=alapanyagok.id and recept_id=$id and
alapanyagok_mertekegyseg.alapanyagok_id=alapanyagok.id and
mertekegyseg.id=alapanyagok_mertekegyseg.mertekegyseg_id ";
$hozzavalok = $conn->query($sql);

```

Mindkét lekérdezésben és a recept.php egészében a \$id-t használom., melynek az értéke a \$\_GET[„id”]. Azért alkalmaztam a get műveletet mivel így a böngészőben hivatkozás formájában lehetséges megosztani másokkal is receptet.



A hozzászólásoknál lekérdezem az összes hozzászólást az aktuális recepthez, aminek a hidden értéke nulla (az-az false). Amennyiben a felhasználó le van tiltva, nem jelenik meg a hozzászólás input mezője. Az input mezőt szűrjük a fentihez hasonló módszerekkel.

Minden felhasználónál megjelenik a hozzászólásoknál a report gomb amivel jelezhet az üzemeltetők felé ha sértő vagy nem helyénvaló a hozzászólás. Amennyiben az aktuális felhasználó rendelkezik hozzászólások moderálási jogával akkor nála az előbb említett gomb egy törlés gomb lesz amivel egyből törli (az-az elrejt) a hozzászólást és a rendszer kiküldi az értesítő üzenetet a felhasználónak.

A recept moderátoroknak két egyéni opciójuk is van. A recept szerkesztése gomb bal ők is szerkeszthetik a receptet, nem csak a készítője. A recept felvitele vagy levétele gombbal tehetik nyilvánossá vagy nem publikussá a receptet.

#### 4.2.5. Keresési lehetőségek

Az oldalon három keresési lehetőség áll rendelkezésünkre. A legegyszerűbb a menüben

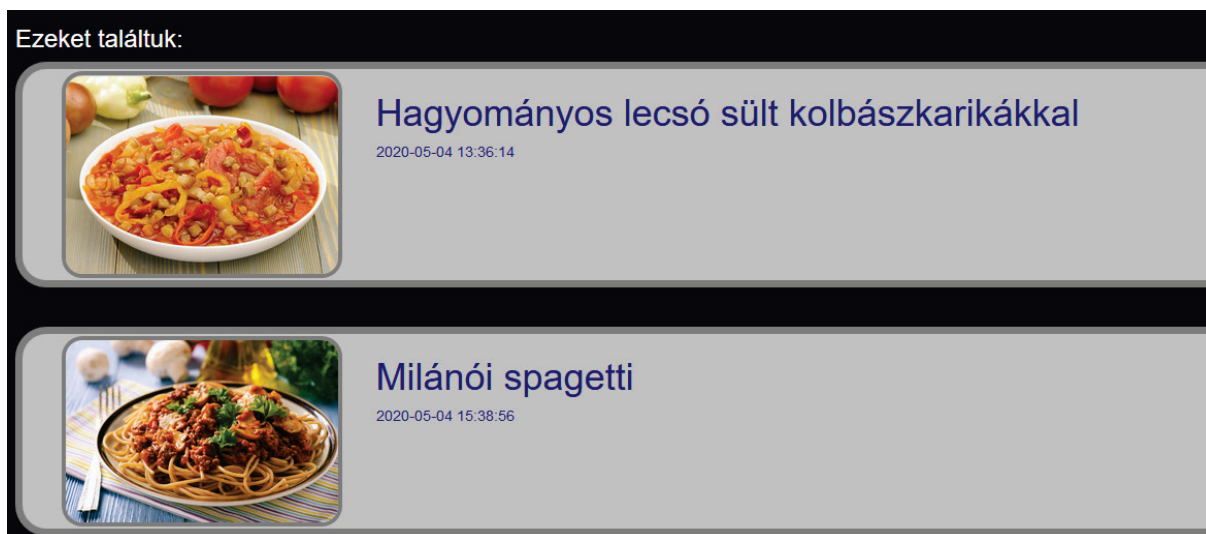


4.2.5.1 katalógus -tag cloud

található keresési mező. Itt a recept nevére szűrünk rá, ami egy egyszerű szöveges bevitellel történik. Az ennél

érdekesebb

katalógus menüpontnál azonban egy tag cloud-al van lehetőségünk a kategóriákra bontottan böngészni.



4.2.5.2 katalógus -tag cloud találatok

A kategóriák előfordulásától függően az egyes kategória név mérete változhat. minél nagyobb a kategória annál több recept használja.

A harmadik és egyben a legbonyolultabb keresési mezőnk a részletes keresés. Itt szűrhetünk névre, alapanyagra és kategóriára egyaránt. Az egyes keresési paramétereket halmozhatjuk és így többszörös szűréssel szűkíthetjük a találatokat.

A találatoknál egy badge jelzi mennyi egyezést talált a



4.2.5.3 katalógus -részletes keresés találat

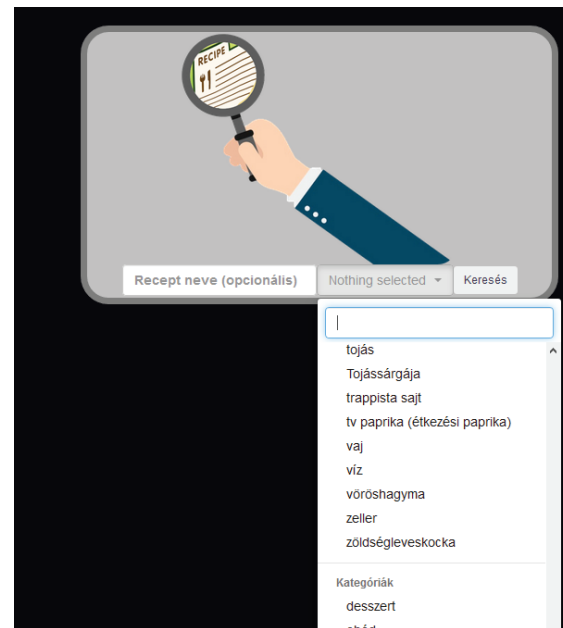
rendszer az alábbi módon:

Mint ahogy az előző fejezetekben kifejtettem, az itt végbemenő keresés egy megengedő kereső, az-az, ha már egy feltételt talál a receptben akkor visszaadja felénk. Ha a felhasználó nem tudja mit akar elkészíteni, csak beviszi a keresésbe az otthon megtalálható hozzávalókat akkor az oldal egyezés szerint csökkenő sorrendben adja vissza a találatokat, így a legtetején található az a recept, amihez a legkevesebb dolgot kell beszereznie elkészítéshez.

A részletes keresésnél logaritmikus (bináris) keresési algoritmust alkalmaztam rekurzívan az alábbi módon:

```
function binaryKereses(Array $arr, $elso, $utolso, $x)
{
    if ($utolso < $elso)
        return -1;

    $mid = floor(($utolso + $elso) / 2);
    if ($arr[$mid][1] == $x) {
        return $mid;
    } elseif ($arr[$mid][1] > $x) {
        return binaryKereses($arr, $elso, $mid - 1, $x);
    } else {
```



4.2.5.3 katalógus -részletes keresés

```

        return binaryKereses($arr, $mid + 1, $utolso, $x);
    }
}

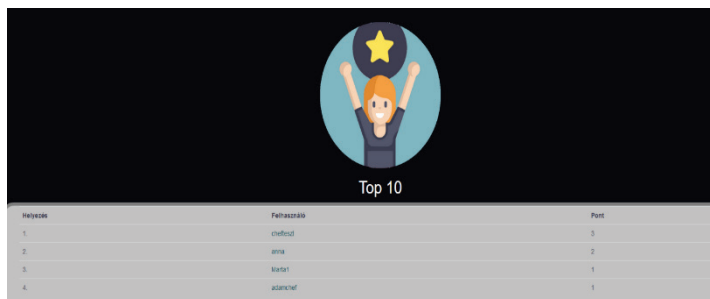
```

Az SQL lekérdezésben eleve rendezett formában kérem le az adatokat, de a keresési adatokat php rendezési függvényekkel valósítom meg.

A site-on található select egy bootstrap-select jquery bővítmény eredménye. A pluginnak köszönhetően több opció is választható az options-ben, valamint lehet az elemekre aktívan keresni és a kilistázott adatokat lehet kategorizálni. Mivel a kategóriák és a hozzávalók más id-vel rendelkeznek ezért a post metódussal nem a kiválasztott elemazonosítóját hanem a nevét adjuk vissza, amihez id-t párosítunk sql lekérdezéssel.

#### 4.2.6. Toplista és profilok

A toplista egy egyszerű lekérdezéssel megy végig, ahol az eredményt egy tömbbe mentjük és kiíratáskor foreach ciklussal végig megyünk rajta.



Top 10		
Rangsor	Felhasználó	Pont
1.	csabai	5
2.	anna	2
3.	boris	1
4.	adrian	1

4.2.6.1 toplista



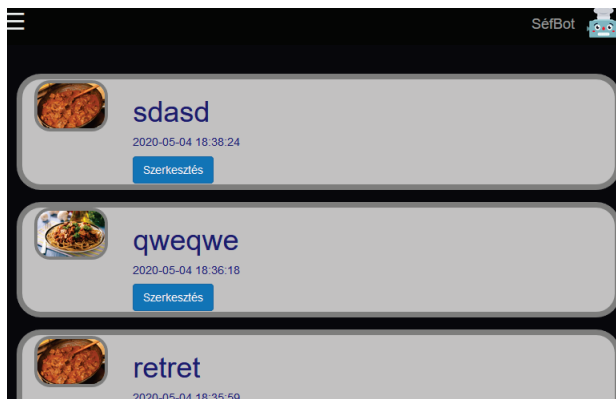
4.2.6.2 profil recepttel

A profilok megtekintését a tervezetben leírtak alapján teljes egészében sikerült implementálnom. A profilon a publikus adatok a felhasználó neve, pontja, kategóriáj és (amennyiben van) publikus receptei. Saját profilunkon megjelenik a moderált hozzászólásaink, valamint ha a kellő jogkörrel rendelkezünk akkor másoknál is.

## 4.2.7. Saját receptek és profil

### 4.2.6.3 profil moderált hozzászólással

Az előző pontban bemutattam a profilok nézetét ami belefoglalta a saját profil megtekintését, ezért csak a saját receptek nézetét fejteném ki.



A saját receptek nézetnél minden recept kilistázódik amit létrehoztunk, beleértve a nem publikusakat is (lásd. 4.2.7.1 ábra). A receptek időrendi sorrendbe vannak rendezve, fentről csökkenően. A szerkesztés gomb egy másik oldalra viszi a felhasználót amit a következő pontban veszünk át.

### 4.2.7.1 saját receptek

## 4.2.8. Recept szerkesztés

A receptek szerkesztése az adott recept szerzőjének és a jogkörrel rendelkező moderátornak áll rendelkezésére. Ilyenkor az új recept létrehozásához hasonló oldal kerül betöltésre, azzal a különbséggel, hogy itt az adatok már ki vannak töltve. Az adatokat kedvünkre módosíthatjuk és végül felülírhatjuk a mentéssel az eredetit, annyi következménnyel, hogy így újra jóvá kell hagynia egy moderátornak.

## 4.2.9. Recept felvitele

### 4.2.9.1 új recept létrehozása felület

Az új recept létrehozás kulcsfontosságú eleme alkalmazásunknak, ezért is okozhatta a legnagyobb fejfájást készítéskor. A végeredményként kapott felület php, javascript és egy jquery bootstrap plugin ötvözetét alkotja. Az egyes komponensek

közötti kommunikálás volt a legnehezebb megoldani.

**Hozzávalók:**

Búza finomliszt  dkg

Tehéntej (1,5%)  ml

**Kategóriák:**

- ml
- cl
- dl
- l

4.2.9.2 hozzávaló mértékegysége

hozzávalót lehessen megadni dinamikusan. A hozzávaló kiválasztásakor automatikusan a hozzátartozó mértékegységek jelennek meg.

#### 4.2.10. Moderálás

Amennyiben a felhasználó bármely speciális jogkör birtokában van, megjelenik a kezelés menüpont a weboldalon. A kezelés.php-n az adott jogkör hatáskörét lefedő műveletekhez szükséges felület jelenik meg.

Felhasználó	Hozzászólások moderálása	Alapanyagok hozzáadása	Felhasználók moderálása	Kategorizálási jog	Recept szerkesztő	Jogok kezelése
admin	✓	✓	✓	✓	✓	✓
bela32	✓	✗	✗	✗	✗	✗

Felhasználó	Moderálási előzmény	Hozzászólás	Bejelentés
teszt	2	Ez nem jó. Nekem nem ízlet	1

[Téves bejelentés](#)

[A kiválasztott felhasználót itt lehet tiltani.](#)

Felhasználó:

[A kiválasztott felhasználó tiltásának visszavonása.](#)

Felhasználó:

**Alapanyag felvétele:**

Neve:  Mértékegysége:

Energia értéke ( kcal )  Fehérje ( gramm )  Zsír ( gramm )  Szénhidrát ( gramm )

**Alapanyagok:**

Neve	energia	fehérje	szénhidrát	zsír
Asztali só	0	0	0	0

A hozzávalók kiválasztásához és a leírás megadásához alkalmaztam javascriptet, mivel így lehetett megoldani, hogy tetszőleges számú lépést és

- A felhasználók moderálása jogkörrel rendelkezők itt tilthatják le a felhasználókat, valamint itt vehetik le a tiltást is.
- a hozzászólások moderálása jogkörrel rendelkezők itt láthatják a beérkezett bejelentéseket és dönthetnek róluk
- a kategóriák vagy alapanyagok moderálása jogkörrel rendelkezők itt vehetnek fel kategóriát és alapanyagot.

- a recept moderálása jogkörrel rendelkezők itt látják az elfogadásra váró recepteket és a hivatkozáson keresztül szerkeszthetik, valamint publikussá tehetik azokat.
- az admin felhasználó itt oszthat vagy vonhat meg jogokat.

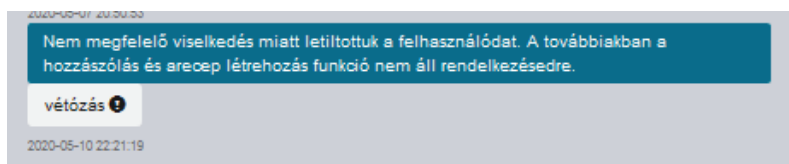


#### 4.2.11. Üzenetek

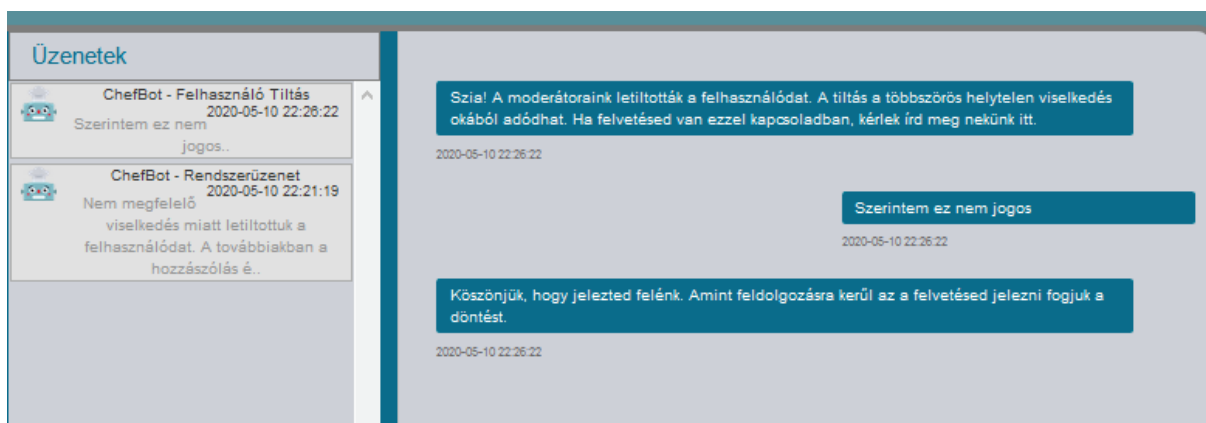
A regisztrációnál már említettük, hogy az új felhasználók üdvözlő üzenetet kapnak, amit az üzenetek menüpontnál megtekinthetnek. Az értesítések alapesetben rendszerüzenetként fordulnak elő, de a következő funkciónál bemutatásra kerülnek a kivételek is.

#### 4.2.12. Vétózás

Amikor a moderátorok letiltják vagy egy hozzászólását moderálják a felhasználónak akkor a kiküldött rendszerüzenet mellett megjelenik egy vétózás gomb amivel megpróbálhatják a felhasználók megindokolni viselkedésüket. Ez a funkció az alább módon néz ki:



4.2.11.1 vétózás gomb



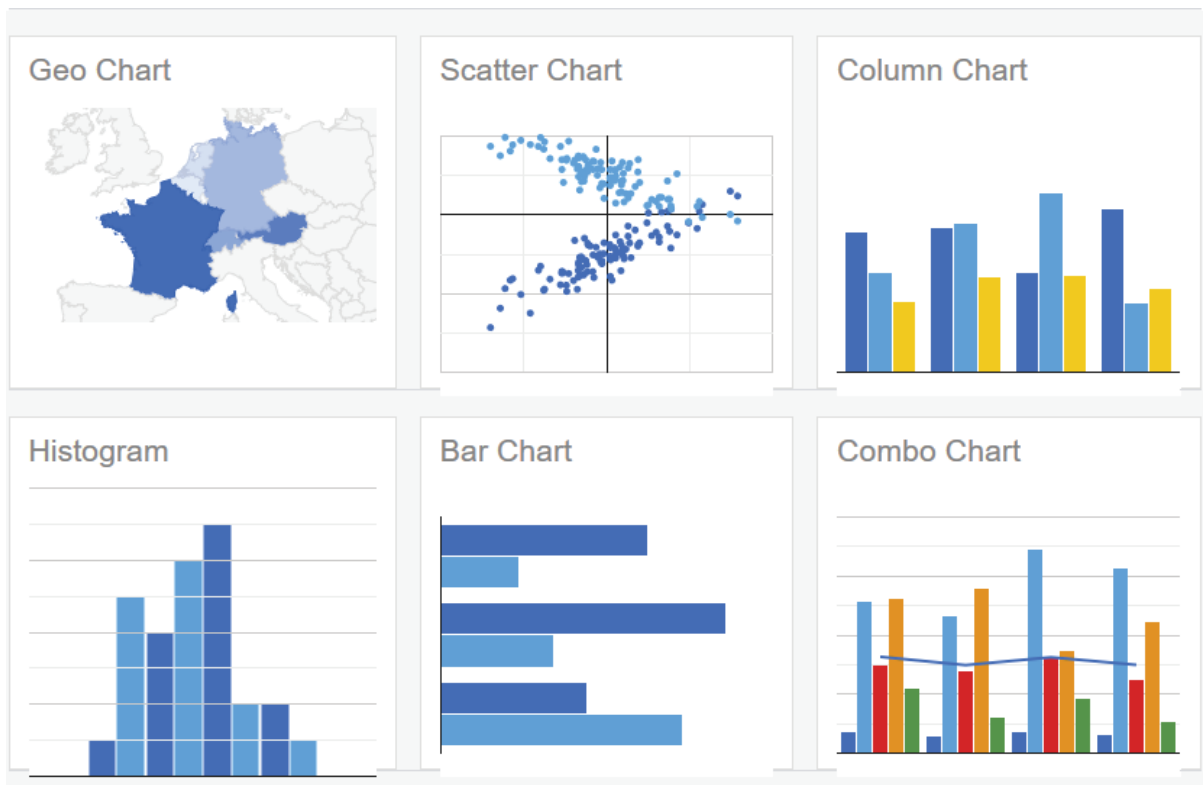
4.2.11.2 vétó elküldése és automatikus üzenetek

A gomb megnyomásakor annak függvényében hogy milyen témában történt, a felhasználó automatikus üzenetet kap amiben megkérlik, hogy lehetőleg röviden írja le indokát. A felhasználó üzenete elküldése után kap még egy automatikus üzenetet melyben tájékoztatjuk hogy moderátoraink kivizsgálják az ügyet.

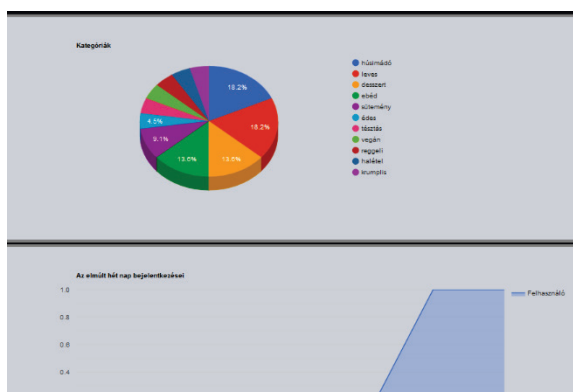
Adatbázis oldalon csak a felhasználó üzenetét tároljuk az ezzel a folyamattal megnyílt új üzenet ticket keretein belül.

### 4.2.13. Kimutatások

A kimutatások menüpontban foglalt funkciókat a tervezésben nem említettem, mivel utólagosan került bele a Google Charts API megismerése után. A projektem végé felé egyre többször kényszerültem rá a javascript használatára, így találtam rá az egyszerűen implementálható Google Charts-ra.



4.2.13.1 Google Charts <https://developers.google.com/chart/interactive/docs/gallery>



4.2.13.2 kimutatások menüpont

A kimutatások menüpontban alkalmaztam, amit csak az admin felhasználó érhet el. Megnyitáskor lekéri a kimutatásokhoz szükséges adatokat az adatbázistól és ezeket különböző dinamikus grafikonokon ábrázolja. Az adminnak jól átlátható módon bemutatja, a leggyakrabban használt kategóriákat, alapanyagokat valamint az elmúlt hét aktív felhasználóit és új recepteit.

## Irodalomjegyzék

1. PHP dokumentáció: <https://www.php.net/> (2020.02.12.)
2. CSS dokumentáció: <https://devdocs.io/css/> (2020.01.10.)
3. JavaScript dokumentáció: <https://devdocs.io/javascript/> (2020.03.21.)
4. Bootstrap dokumentáció: <https://getbootstrap.com/docs/4.4/getting-started/introduction/> (2020.02.10.)
5. jQuery dokumentáció: <https://api.jquery.com/> (2020.02.10.)
6. Apache: <https://www.apachefriends.org/hu/index.html> (2019.08.17.),  
[https://hu.wikipedia.org/wiki/Apache\\_HTTP\\_Server](https://hu.wikipedia.org/wiki/Apache_HTTP_Server) (2020.01.30.),
7. MySQL dokumentáció: <https://dev.mysql.com/doc/> (2019.02.25.),
8. SZTE Dr. Holló Csaba webtervezés kurzus előadási tananyaga
9. Webes alkalmazások és architektúrák: [https://ade.web.elte.hu/wabp/lecke1\\_lap1.html](https://ade.web.elte.hu/wabp/lecke1_lap1.html) (2020.02.07.),
10. Németh Gábor adatbázisok gyakorlat: <https://www.inf.u-szeged.hu/~gnemeth/kurzusok/adatbazisok.html> (2019.04.20.),
11. Xampp letöltés: <https://www.apachefriends.org/hu/download.html> (2019.05.20.)
12. Google Charts dokumentációja: <https://developers.google.com/chart/interactive/docs> (2020.05.05)



## Nyilatkozat

Alulírott Kocsor Levente Ferenc, programtervező informatikus szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet Számítógépes Algoritmusok és Mesterséges Intelligencia Tanszékén készítettem, BSc diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat / diplomamunkámat a Szegedi Tudományegyetem Informatikai Intézet könyvtárában, a helyben olvasható könyvek között helyezik el.

2020.05.11

Kocsor Levente Ferenc

## Mellékletek

### **chefbot** mappa

Tartalma:

- weboldal forrása  
forráskód, táblákat, teszt adatok létrehozó sql fájl
- img mappa a szükséges képekkel
- Xampp telepítő csomag

### **dolgozat** mappa:

Tartalma:

- dolgozatom pdf formátumban
- a szakdolgozatban használt fontosabb ábrák, grafikonok png formátumban

## **Köszönetnyilvánítás**

Ezúton szeretnék köszönetet mondani mindenkinek, aki hozzásegített diplomamunkám elkészítéséhez.

Köszönettel tartozom témavezetőmnek, Dr. Németh Tamásnak, aki lehetőséget adott a projektem létrehozásához és szabad kezet biztosított a technológiák használatánál.

Hálás köszönetem fejezem ki továbbá mindazon barátomnak, kollégámnak, akikhez a dolgozat készítése folyamán kérdéseimmel fordulhattam, és tapasztalataikkal, javaslataikkal támogatták munkámat.