Szegedi Tudományegyetem

Informatikai Intézet

Recipe hoarder webes alkalmazás

(Recipe hoarder web application)

Szakdolgozat

Készítette:
Vas Laura
gazdaságinformatika szakos
hallgató

Témavezető:
Dr. Bilicki Vilmos
egyetemi adjunktus

Szeged
2021

Tartalomjegyzék

Feladatkiírás

A szakdolgozat során egy Angular keretrendszerben kialakított webes alkalmazás létrehozása volt a feladatom. A projekt a Firebase felhőalapú szolgáltatásait használja. A fejlesztés során a legfőbb cél a recept importálás más honlapokról volt. Az importálás második legfontosabb lépése az alapanyagok szétválogatása, hogy később a bevásárlólistába helyezésnél a megegyező anyagokat össze lehessen adni.

Tartalmi összefoglaló

- Téma megnevezése:
  A szakdolgozat céljául kitűzött témám egy Angular-ban írt web applikáció, ami recept megjelenítésre és importálásra használható.

- Feladat megfogalmazása:
  Az importálás funkció lehetővé teszi, hogy a felhasználók egy helyen gyűjtsék a receptjeiket. Továbbá, hogy a receptek összetevőit egy bevásárló listába ki tudják menteni, ezzel is megkönnyítve a mindennapi életet. A felhasználók a többiek által létrehozott receptek között tudnak keresni, és a nekik tetsző recepteket ki tudják menteni a saját receptgyűjteményeikbe.

- Megoldási mód:
  Az applikáció egy weblap formájában lett megvalósítva, mivel így lehet a legtöbb eszközt elérni egyetlen kód bázissal. A megvalósításhoz a már említett Angular keretrendszert használtam, illetve a Firebase felhő alapú szolgáltatásait. Mivel mind a kettő (Angular, Firebase) a Google terméke, ezért várhatóan hosszútávon támogatva lesznek. A felhasználó a recept URL-je alapján tud receptet importálni, vagy manuálisan is lehet létrehozni újat. Ekkor az importáláshoz egy szerver oldali funkció fut le és próbálja értelmezni a megkapott URL-en lévő html fájlt. Ennek egy fontos lépése az, hogy az alapanyagok nevét, mértékegységét és mennyiségét az eredeti helyről kiolvassa. Miután a receptet sikeresen importáltuk, azokat a Firebase adatbázisában tároljuk.

- Alkalmazott eszközök, módszerek:
  Mind az importálás mind az egész projekt során törekedtem, hogy minél modulárisabb legyen a felépítés. A webapp fejlesztése során a PWA-t alkalmazva elérhető, hogy bizonyos funkciók offline is működjenek. Ezt a modern, könnyen kezelhető weblapot számítógépen és telefonon egyaránt lehet használni.

- Elért eredmény:
  A fejlesztés során sikerült egy modern telefonos és számítógépes környezetben is elérhető webes alkalmazást készíteni, ami bárki számára a regisztráció után elérhető és könnyen használható, ezzel egyszerűbbé téve a hétköznapokat.

- Kulcsszavak:
  Angular, Firebase, pipeline architektúra, PWA, telefonos nézet

Motiváció

Egyetemisták, mint én is egyre közelebb vagyunk ahhoz az életformához, ahol önellátók vagyunk, ennek fontos része a főzés és étkezés. Manapság nagyon egyszerű különböző recepteket, különböző országokból, kultúrákból találni, viszont ez temérdeknyi weblapot jelenthet. Ennek hátulütője, hogy egy idő után követhetetlen lesz, hogy egyáltalán hova regisztráltunk, valamint, hogy "melyik weblapon is volt az a bizonyos recept, amit egyszer már kipróbáltam, és tetszett". Személyes tapasztalatom ezzel kapcsolatba pedig, hogy én egy chat alkalmazásban gyűjtöttük a barátommal az URL címeket, hogy legközelebb is megtaláljuk, de már kezdett nagyon követhetetlen lenni.

Azért választottam ezt az ötletet a szakdolgozatom témájának, mert ez egy személyes problémám már hosszú ideje. Láttam már korábban próbálkozásokat, de egyik sem volt az én elképzelésemnek megfelelő. A célom az volt, hogy egy egyszerű URL cím másolással pillanatok alatt egy helyen lehessen a megtalálni mindent.

A továbbiakban részletesen kifejtem az általam tervezett és megvalósított webes applikáció felépítését és funkcióit. A bemutatót a konkurencia ismertetésével kezdem.

1. fejezet

Piackutatás

Már létező programokra öt példát hoztam, amik mind valamilyen szinten különböznek.

Felhasználókörük, funkcióik, előnyök és hátrányok az én tervemhez képest. Az én terveimhez képest egy rövis ismertetők az előnyeikről valamint hátrányaikról. Miért jobb vagy mivel tudnak többet, mint amit én terveztem egyenlőre kialakítani.

1.1. Grocy

A Grocy egy lokálisan hosztolható weblap, ez azt jelenti, hogy minden felhasználónak rendelkeznie kell egy szerver géppel,amin magát a weblapot tudja üzemeltetni. Maga a weblap kifejezetten sok funkcióval rendelkezik, de ezeknek a teljes kihasználásához sok időt és munkát kell befektetni. Sajnos mivel mindenki saját magának futtatja, ezért a beépített adatbázisa üresen kezd. Ellenben, ha valaki hosszú ideig használja, akkor a befektetett munka kifizetődő tud lenni.

A recept kezelő lapja csak manuálisan feltölthető, tehát nincs importálásra lehetőség. Rendelkezik bevásárlólista és “sufni” opciókkal is. Az otthon lévő alapanyagokat egyessével, tetszőleges részletességgel fel lehet venni a “sufniba”, ezzel leltározva, hogy milyen alapanyagok vannak otthon. Ezekről eltárolható adatok közé tartozik, hogy mennyi van belőle, meddig jók, képet, de akár a vonalkódját is. A bevásárló lista pedig egyértelműen a vásárlást segítő funkció, aminek a végén, egy kattintásra átrakható “sufniba”.

Már ezen leírás alapján is látszik, hogy ahhoz, hogy ez a rendszer használható legyen, egy komoly lokális adatbázist kell létrehozni az alapanyagokból és azok adatairól, valamint a receptekről. Ez a rendszer csak olyan emberek számára használható, akik rendelkeznek hardverrel és tudással, hogy maguknak futtassák a weblapot.

1.2. Delish

Ezt a weblapot azért választottam példaként, mert ez egy tökéletes példa egy átlagos, egyszerű receptes weblapokra. A honlapon csak recepteket és talán pár blog bejegyzés található regisztráció után is. Ez a weblap reprezentálja a legtöbb hasonló, csak blogként működő weboldalt.

Egy receptre kattintva látjuk az alapadatokat, hozzávalókat, elkészítési javaslatot valamint alap adatokat mint az elészítési idő. A weblapon találhatóak még hasonló recept ajánlások, de ezen felül több szolgáltatást nem nyújt.

1.3. Yummly

Ez egy fejlettebb verziója a korábban említett "átlagos" weblapoknak. Bejelentkezés nélkül kevesebb funkció érhető el, viszont utána már kifejezetten sok funkcionalitása van. A webes kinézeten felül applikációval is rendelkezik.

Az alap recept keresésen kívül, itt már lehetőségünk van azok elmentésére a sajátjaink közé. A weblap rendelkezik bevásárló lista funkcióval, valamint képes azonnal a receptből áthelyezni az alapanyagokat is. Egy kiemelkedő funkciója az étkezés tervező. Ez, figyelembe véve esetleges allergiákat, vagy étrendeket ajánj és segít tervezni a következő időszakra.

Ami hátrány az egész weblapon, hogy nem közösség bővíti a recept adatbázist, ezért limitált a receptek száma és nem lehet mindent megtalálni. Még akkor is, ha figyelembe vesszük a manuális recept készítést, nem feltétlenül a legegyszerűbb használni, mivel mindig egy külső helyről ki kell kikeressük amit akarunk, majd kézzel beírnunk.

1.4. BigOven

A legnagyobb különbség az eddigiekhez képest, hogy ez a weblap már rendelkezik recept importáló funkcióval is. Azon felül négy különböző módon lehet újjakat létrehozni, manuálisan leírva, másolni egy már létező fájlból és beilleszteni, képről vagy scennelésről beolvasni vagy URL címmel importálni. Az importálás során nem tárolják el az egész receptet, ha más honlapról származik. "Our Pledge to Food Bloggers" leírja, hogy miért, viszont ez azt jelenti, hogy a teljes receptet megtekintsük, át kell navigálni az eredeti oldalra. Ezen felül a bevásárló listában nem adódnak össze a termékek, valamint nincsenek kategóriák a receptekhez.

Egy nagy hátránya a weblapnak, hogy kissé régi stílusú. Az oldal nem használ túl sok modern stílust. A képek, beviteli mezők, lista nézetek mind úgy néznek ki, mint amin épp hogy van egy kis formázás. A weboldal navigációja nem túl felhasználóbarát.

Annak ellenére, hogy a weblapnak mennyire nem modern stílusa van, az applikáció igenis követhető. A funkciók szintén jól működnek. Elméletileg IOS-en is létezik, viszont arra nincs lehetőségem, hogy felmérjem milyen különbségek lehetnek a két verzió között. Egy nagy előny, hogy az ingyenes verzióban is használhatóak az alapfunkciók.

1.5. ChefTap

Az összes közül valószínűleg ez az applikáció, ami a legtöbb funkcióval rendelkezik. Technikailag van webes és telefonos verziója is, viszont a webes csak recept megtekintésre használható. Minden egyéb, beleértve a recept importálást, bevásárló listákat és étkezés tervezőt csak az applikációk keresztül lehet elérni és szerkeszteni. A weben volt lehetőség Google segítségével bejelentkezni, viszont az applikációnak nem volt ilyen lehetősége. Ennél az appnál az ingyenes verzió elég limitált, a recept importáláson kívül semmi sem működik a próbaidőszak lejárta után.

Ezen a felületen nincs mások által, vagy akár csak egy közös adatbázisból való keresésre és importálása lehetőség a recepteknél. A felhasználónak mindent magának kell beszerezni.

A beimportált recepteket könnyű módosítani, valamint rengeteg kis adatot megadni, hogy otthonosan lehessen használni a környezetet. Ebben az applikációban nincs lehe-

tőség közvetlenül a receptből a bevásárló listába rakni alapanyagokat, menüket összekészíteni vagy az étkezéstervezőt használni az ingyenes próbaverzió után.

1.6. Összefoglaló

Egy táblázat a korábban összefoglalt példákról és tudásaikról egy egyszerű összehasonlításhoz.

Összegzés

weblapok	Grocy	Delish	Yummly	BigOven	ChefTap	RecipeHoarder
open source	✓	✗	✗	✗	✗	✗[1]
fizetős	✗	✗	✗	✓	✓	✗[2]
recept importálás - URL	✗	✗	✗	✓	✓	✓
receptekhez vannak lépések	✗	✓	✓	✗	✓	✓
manuális recept hozzáadás	✓	✗	✗	✓	✓	✓
bevásárlólista	✓	✗	✓	✓	✓	✓
bevásárlólista ajánló	✓	✗	✗	✗	✗	✓
katgóriák használata receptekhez	✗	✓	✓	✗	✓	✓
étkzés tervező	✓	✗	✓	✗	✗	✗
bejelentkezés google fiókkal	✗	✓	✓	✓	✗	✓
reszponzív weblap	✓	✗	✗	✗	✗	✓

1.1. ábra. Összefoglaló táblázat a piackutatásból

[1] Technikailag az, de nem könnyű hosztolni.

[2] Jelenleg nem fizetős, de amennyiben fellendül az applikáció népszerűsége, akkor Firebase-el könnyen bevezethető.

2. fejezet

Funkcionális specifikáció

A következők összefoglalják a program főbb funkcionalitásai, azokat felhasználási esetekben (UseCase) csoportosítja. Ez alkotja program funkcionális követelményeit, azaz azokat a képességeket, amiket mindenféleképpen tudnia kell. Ezeket az elemeket a 2.1-es ábra grafikusan ábrázolja, majd a fejezet további része pontosítja és kifejti őket.

2.1. Bejelentkezés/Regisztráció

A weblap használatához be kell jelentkezni vagy regisztrálni kell. Ameddig ez nem történik meg,a felhasználó vendég státuszban van (lásd: 2.1-es ábra “Vendég”).

A regisztrációra két lehetőség van. Egyrész a regisztrációs oldalon a mezők (pl.: név, születési év, email, stb) kitöltésével tud manuálisan regisztrálni (lásd: 2.1-es ábra “Regisztráció manuálisan”). Ekkor a rendszer az azonosításra a megadott email címet használja és ezt kell megadni a jelszavával együtt a belépéskor. Másrész, ha a vendégnek van Google fiókja, akkor egyszerűen egy lépéssel azonnal regisztrálhat annak felhasználásával (lásd: 2.1-es ábra “Regisztráció Google fiókkal”).

A bejelentkezésnél a regisztrációhoz hasonlóan két lehetőség áll rendelkezésre. Attól függően, hogy a felhasználó mely módszerrel regisztrált, az annak megfelelő módszerrel kell belépnie. Tehát, ha manuálisan regisztrált, akkor meg kell adni a regisztrációkor használt email címet és jelszót (lásd: 2.1-es ábra “Bejelentkezés manuálisan”). Amennyiben Google fiókot használt a regisztrációhoz, akkor egyszerűen a “Sign In with Google” gombra kattintva tud belépni (lásd: 2.1-es ábra “Bejelentkezés Google fiókkal”).

2.2. Receptek

A következőkben a receptekhez kapcsolódó főbb funkciókról lesz szó. Ez az ábrán a “Receptek” csoportban található felhasználási eseteket jelenti. ezen funkcionalitások alkotják az alkalmazás legfontosabb képességeit. A piackutatásban megállapított fontos funkciókat tartalmazza.

2.2.1. Receptek hozzáadása

A weblap egyik jellegzetes alapfunkciója, hogy a közösség tölti fel az adatbázis receptekkel. Ez azért fontos, mert így nem az adminisztrátoroknak kell folyamatosan dolgozni, hogy minél több recept legyen elérhető a felhasználók számára.

Egy recept feltöltésre két lehetőség is van. Az egyik az, hogy a felhasználó manuálisan tölti ki a recept paramétereit (lásd: 2.1-es ábra “Recept hozzáadás manuálisan”). Ez azokban az esetekben fontos a felhasználó számára,amikor a recept nem az internetről származik, hanem például nyomtatott receptkönyvből vagy családi hagyományból.

A másik opció az, hogy egy másik recepteket tartalmazó weboldal URL címét kimásolja a böngészőből és beilleszti a linket a RecipeHoarder importáló felületére (lásd: 2.1-es ábra “Recept hozzáadás URL-ből”). Itt, amennyiben a forrás weblap támogatott, a program betölti a recept adatait. Ezután a felhasználónak lehetősége van módosítani a receptet, majd azt kimérve hozzá adni a globális recept lisához. //TODO: calória számlálás // TODO: category

2.2.2. Receptgyűjtemény kezelés

Ahhoz, hogy a weblap könnyen használható legyen szükségünk van arra, hogy a felhasználó ki tudjon menteni recepteket egy saját gyűjteménybe. Ez nagyban megkönnyíti a recept későbbi megtalálását. Erre két különböző gyűjtemény áll a felhasználó rendelkezésére. Egyik a sajt “recept könyve”, amibe akárhány receptet kimethet, illetve a kedvencnek ítélt receptek gyűjteménye. A felhasználó bármely megtekintett vagy frissen beimportált receptet ki tud menteni a recept könyvébe (lásd: 2.1-es ábra “Gyűjteményhez adás”), majd innen amennyiben tetszett neki hozzá tudja adni a kedvencekhez. Ez azt jelenti, hogy a kedvencekhez való hozzáadáshoz szükséges kimenteni a saját receptkönyvbe először.

Természetese a felhasználó bármikor meg tudja tekinteni az összes receptet amit a saját receptkönyvbe vagy kedvencek közé mentett ki (lásd: 2.1-es ábra “Gyűjtemény megtekintése”). Majd amennyiben egy receptet már nem szeretne az adott listában látni, akkor képes a receptet kivenni a kedvencek/saját receptkönyvből, de ezzel nem törlődik a globális receptlistából, így a többi felhasználó számára elérhető marad (lásd: 2.1-es ábra “Gyűjteményből törlés”).

2.2.3. Recept megtekintés

Mivel ez egy recept gyűjtő webes alkalmazás, ezért egyértelműen lehetőség kell arra, hogy a receptek leírását meg lehessen tekinteni. Itt már megtalálható az összes eltárolt adat a receptről, mint például a elkészítési idő (ha van), kalória (ha van), hozzávalók vagy elkészítési lépések (lásd: 2.1-es ábra “Recept megtekintés”).

2.2.4. Receptre szűrés

A weblapon van lehetőség keresni a receptek között, ezen belül is két mód van erre. Egy általános névre való keresés, valamint kategória szerint is lehet szűrni. Mivel egy recept létrehozásakor kötelező megadni legalább egy kategóriát. Ezzel biztosítva van, hogy minden recept tratozik legalább egy előre megszabott kategóriába és így egyszerűen kereshetők ez alapján.

Ez a 2.1-es ábra “Receptek” csoport alatt, a “Receptre szűrés”-nek felel meg a két lehetséges verziójával, “név szerinti” és “kategória szerinti” szűrés.

2.2.5. Hibás recept jelentése

Korábban említetten, az oldal adatbázisa a közösség által feltöltött receptek alapján nő. Ezért van lehetőség, hogy akár az importálás során különböző hibák keletkeznek amit a felhasználó nem vesz észre vagy nincs kedve kijavítani. Akár arra is van lehetőség, hogy támadás érje a weblapot, és keletkezzenek olyan receptek, amik hibásak. Ennek javítása érdekében minden recepthez tartozik egy hiba bejelentő rész, ahol előre megaszabott lehetséges hibák listájából választva jelezhet a weblap karbantartóinak, hogy ha valami nem jó a recepten. Ezek a lehetséges hibák közé tartozik például, ha hiányos az alapanyagok listája vagy trágár szavakat tartalmaz.

Ez a 2.1-es ábra “Receptek” rész alatt, a “Recept megtekintés”-hez tartozik “Hibás recept jelentés” néven.

2.3. Bevásárló lista

A bevásárlólista egy olyan kollekció, ami minden felhasználónak lehetséges alapanyagokat tartalmazó gyűjtemény. Ez szintén személyre szabott és csak a felhasználó látja, vagy módosíthatja.

2.3.1. Bevásárló listához adás

A listához adásra két lehetőség van. A szokásos bevásárlólista módszert követve manuálisan is ki lehet tölteni a mezőket, majd a gombra kattintva a listához adni. A másik módszer a recepteken belül, a felsorolt alapanyagokat is egyesével hozzá lehet adni az alapanyag előtt lévő + gombbal. Végül a recepteknél lehetőség nagy mennyiségű alapanyagot azonnal hozzáadni a listához. Mivel itt nem látjuk, hogy tényleg sikerült-e a bevásárló listához adás, ezért a sikeres feltöltés a receptes felületen jelezve van.

Ezen a diagramon a “Bevásárlólista” rész alatt, a “Bevásárló listához adás” résznek felel meg.

2.3.2. Bevásárló lista megtekintés

A bevásárlólista megjelenítésénél, a korábban hozzáadott alapanyagok vannak felsorolva. A bevásárlólista az alapanyagok hozzáadás dátuma szerinti csökkenő sorrendben jelenik meg, hogy könnyen követhető legyen, mikor egy új anyag lett a listához adva. Amennyiben az alapanyagnak nincs mértékegysége vagy mennyisége, akkor azok az adatok nem lesznek megjelenítve, hogy elkerüljük, hogy például “0 pasta“ legyen megjelenítve.

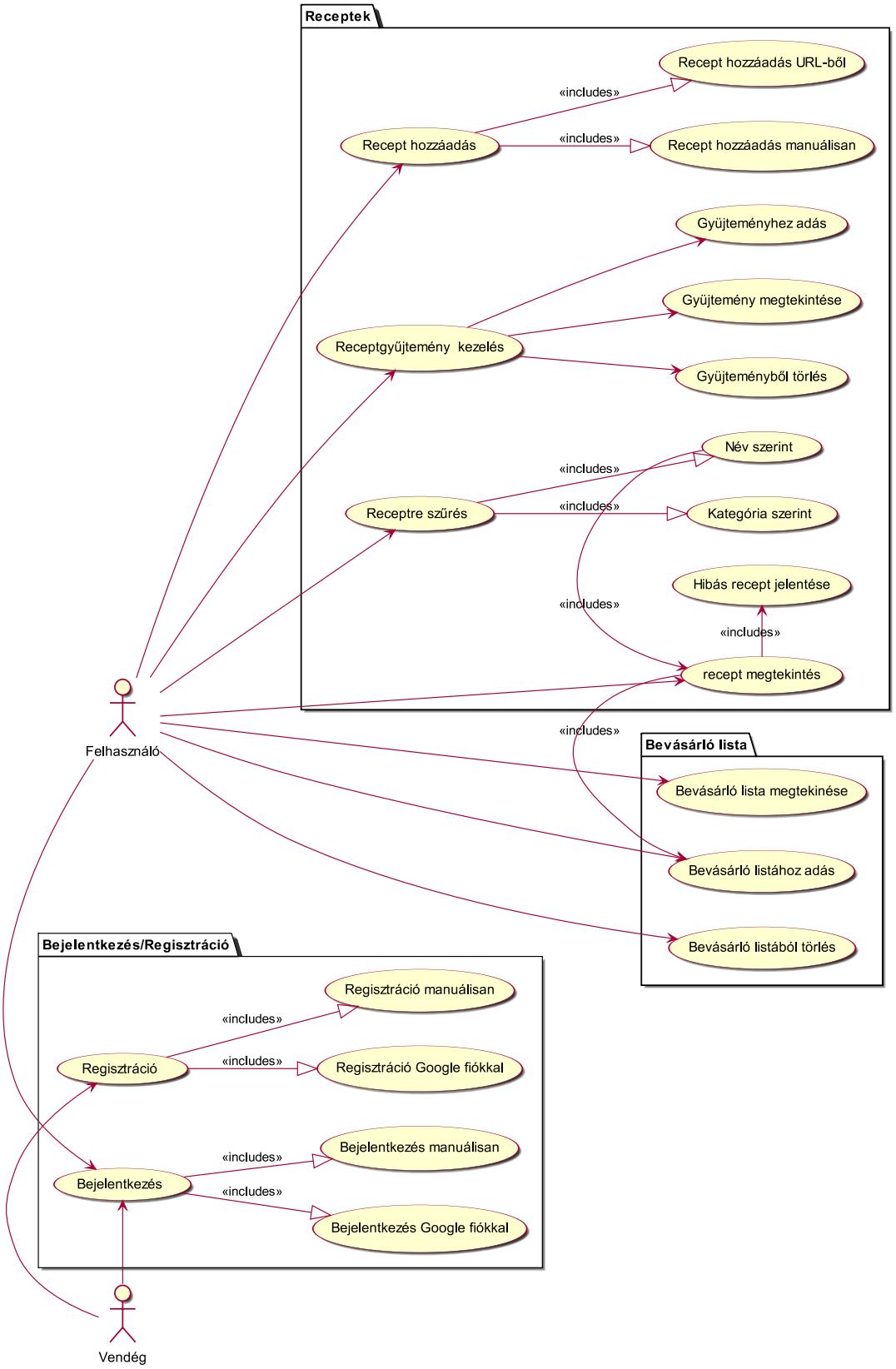
Ezen az ábrán a “Bevásárlólista” rész alatt lévő “Bevásárló lista megtekintése” résznek felel meg.

2.3.3. Bevásárló Listából törlés

Ha a felhasználónak már nem kell egy adott alapanyag a listából, legyen ez azért, mert már megvette vagy csak nem aktuális már, akkor könnyen egy kattintással el lehet tün-

tetni a listából. Másik verzió a lista ürítésére, pedig a lista alatt lévő gomb megnyomásával, az egész lista törlése. Így gyorsan és egyszerűen lehet újrakezdeni a lista használatát anélkül, hogy az összes elemet egyesével kellett volna törölni.

Ezen az ábrán a “Bevásárlólista” rész alatt lévő “Bevásárló listából törlés” résznek felel meg.



2.1. ábra. UseCase diagram

3. fejezet

Felhasznált technológiák

Itt fogom részletezni a kiemeltebb felhasznált technológiákról alapvető információkat, valamint szerepüket ebben a projektben.

3.1. Angular

Egy TypeScript alapú Framework, ami a eredetileg a Google fejlesztette ki, de most már nyílt forráskódú. Az Angular egy teljes újraírása a régi AngularJS-nek. Egyik legnagyobb előnye, hogy komponens alapú keretrendszer, ami segítségével újra felhasználható kódot lehet írni. Támogatja a Single Page alkalmazás készítését. Ezeken a weblapokon a weblap újratöltése nélkül tud a felhasználó navigálni, ezzel is gyorsabbá téve azt.

A felhasználói felület tervezés során is igyekeztem komponenseket kialakítani mind telefonos mind desktopos kinézetre. A komponensekkel való munka megkönnyíti egy projekt fejlesztését, mivel akár saját fejlesztésből, akár külső könyvtárból szerzett elemeketből lehet egyszerűen felépíteni a weblapokat. Ezeknek az elemeknek másik nagy előnye, hogy elég csak az eredetit módosítani ahhoz, hogy mindenhol megváltozzon, ezzel is egységesítve az oldal szerkezetét, valamint segítve a fejlesztők dolgát.

3.2. Angular Material

Az Angular Material egy a Google által fejlesztett komponens csomag. Ez segít a fejlesztőknek már kész és modern dizájnos komponenseket egyszerűen felhasználni a projektekben. Ez elsősorban egy UI komponens dizájn könyvtár, ami importálás után egyszerűen használható. A weboldalán pedig példákkal és részletes dokumentációval fel vannak sorolva a létező komponensek és ezek különböző változatai.

Az én projektem is nagy részben ennek a könyvtárnak a komponenseit használja fel. Csak pár helyen lett kiegészítve a már létező verzió, hogy illeszkedjek az eredeti tervekhez. Ezen felül a projekt témája is az Angular Material alapján lett felépítve, hogy a komponenseknek könnyen meglehessen adni az alap színeket és tipográfia szabályokat.

3.3. FireBase

Mostmár a Google által fejlesztett webes és mobilos alkalmazások készítéséhez létrehozott platform. Manapság nagyon népszerű az egyszerűsített adatbázis felépítése

valamint a könnyű és könnyen értelmezhető kezelő felülete, arról nem beszélve, hogy rengeteg dokumentáció létezik a fejlesztők számára.

Ebben a projektben elég sok helyen használom a Firebase funkcióit. Az egész weblap a Firebase Hosting felületéről működik. A GitHub-bal összekötve, ha a Main Branch-re van feltöltés, akkor a Firebase-en automatikusan deploy-oldódik az új production verzió.

A FireStore Database-t használom a weblap adatok tárolásához. Itt vannak eltárolva receptek, felhasználók, felhasználókhoz különböző kollekciók és bevásárló lista meta adatok. A Cloud FireStore-on belül lehet beállítani az adatbázis Rule-okat.Ezt azt jelenti, hogy be lehet állítani, hogy különböző kollekciókat kik érhetik el, milyen tevékenységek vannak engedélyezve nekik, valamit, hogy milyen jogokkal milyen adatbázis műveleteket lehet csinálni. Amíg nem volt autentikáció beállítva és a development folyamat volt, akkor csak egy dátumot frissítettem, hogy meddig lehet az összes felhasználónak bármit csinálni. Viszont most már le van tiltva a receptek törése, valamint bejelentkezés nélküli lekérdezések is.

A FireBase Storage-ban tárolódnak a receptekből kimentett képek, amik az import során kerülnek az adatbázisba.

A legnagyobb biztonság érdekében a Firebase Auth-ot használom a weblap bejelentkezés és regisztrációs műveletekhez. Így lehetséges manuálisan regisztrálni majd bejelentkezni, viszont, akinek van Gmail fiókja annak lehetősége van egy lépésben azonnal regisztrálni és belépni annak segítségével.

A recept importálás és a bevásárlólista egy része a Firebase Cloud Function keresztül szerver oldalon fut. Ez nem csak olcsóbb, mivel szerver oldali lekérdezések ingyenesek a kliens oldalival szemben, hanem az importálás során felmerült problémát is megoldotta. Az importálásnál felmerűlt, hogy egy honlap nem lérdezhez le adatot egy másikról kliens oldalon keresztül, valamit a CORS okozta nehézségek elkerülése érdekében is hasznos volt. Azon kívül, mikor a bevásárló listába új alakanyag kerül, akkor egy trigger leful a szerver oldalon, ami segítségével állítjuk elő az ajánlóhoz a metaadatokat.

3.4. PWA

PWA azaz Progressive Web Application egy koncepció, amit követve lehetséges megbízható, alkalmazható és telepíthető webes applikációkat fejleszteni. Az alap szabályait követve a weblap platformspecifikus alkalmazássá alakítható. A PWA lehetővé teszi az alkalmazások használatát bárhol, bármikor, bármilyen eszközön.

Ez eseten az Angular-t kellene kiegészíteni egy kevés plusz JavaScript kóddal, hogy ha az eszköznek nincs internet elérése, akkor a script alapján a localstorage-ba elmentett, valamint cache adatokat használja az indításhoz. Ezzel elérhető, hogy valamilyen szinten használható a weblap offline környezetben is. Szerencsére mind az Angular, mind an Angular Fire úgy lett kifejlesztve, hogy csak egy-egy helyen megadtam neki, hogy az Angular Service Worker-t használja, ezért nekem nem kellett saját kódot írni ahhoz, hogy internet nélkül is használható legyen a weblap. Ez persze limitált arra, ami cachelve lett, például a keresés nem nagyon fog működni az összes receptre, de a bevásárló listához adás során, mikor az eszközön újra lesz internet, az egész szinkronizálódik, hogy ne legyen adatvesztés.

3.5. Schema.org

A shema.org egy weblap, ami azért készűlt, hogy a közösség a központosított sémákat használja különböző weblapokon, e-mailekben, modellekben, adatbázisok tervezésében vagy sok más helyen. Sok különböző kódolással használható, de a legnépszerűbbek a JSON-LD, Microdata valamint a RDFa. Ezek a szótárak olyan entitásokat és kapcsolatokat írnak le, amik segítik a fejlesztők számára a megfelelő sémát kiválasztani. Tehát összességében, ez egy tervgyűjtemény.

A projektben fontos szerepet játszott a shema.org által biztosított "Recipe" séma feldolgozása és megértése, mivel a recept importálás más weblapokról ezen alapul. A jól felépített weblapok manapság tartalmaznak egy vagy több JSON-LD meta információt, amiben a weblapon jelenleg megjelenő adatok vannak eltárólva JSON formátumban a Shema.org szerinti felépítés alapján.

3.6. Figma

Egy vektorgrafikus szerkesztő, ahol web prototípusokat lehet tervezni. Ez legfőképp webalapú, de letölthető desktopos verziója is. A program képes a tervezett lapok élő tükrözésére akár saját telefonon az applikáción keresztül, vagy csak az emuláló felületen keresztül is. A figma egy modern dizájn tervező felület, ahova egyszerűen integrálni lehet a material design stílust és alap komponenseket, ezzel is segítve az Angular component szerű gondolkodást. Másik nagy előnye, hogy interaktívvá lehet tenni a tervezés során a gombokat, komponenseket. Valamint a kész dizájnoknak meglehet nézni a pontos CSS stílus kódját is, ezzel is segítve a konkrét kódolás folyamatát.

Sokat segített a tervezésben, mivel a kész material design komponenseket tudtam próbálgatni anélkül, hogy először mélyebben tanulmányoznám az Angular felépítését. Az egységes dijájn és téma felépítésében pedig segített konzisztensnek maradnom. Ez igaz mint a színekre, mind a tipográfiára.

4. fejezet

A rendszer magas szintű áttekintése

Ebben a fejezetben fogom leírni a különböző fontosabb funkciókról a részletesebb felépítését és tudnivalókat.

4.1. Recept importálás

A moduláris felépítés miatt ezt több al részre szedem.

4.1.1. FetchUrlData

asd

4.1.2. JsonLdExtractor

asd

4.1.3. SeparateIngredients

asd

4.1.4. ImageDownloader

asd

4.1.5. CalorieCalculator

asd

4.2. Kliens oldali adatbázis műveletek

Services...

4.3. Bevásárlólista ajánló

trigger és működés

5. fejezet

Architektúra

5.1. asd

6. fejezet

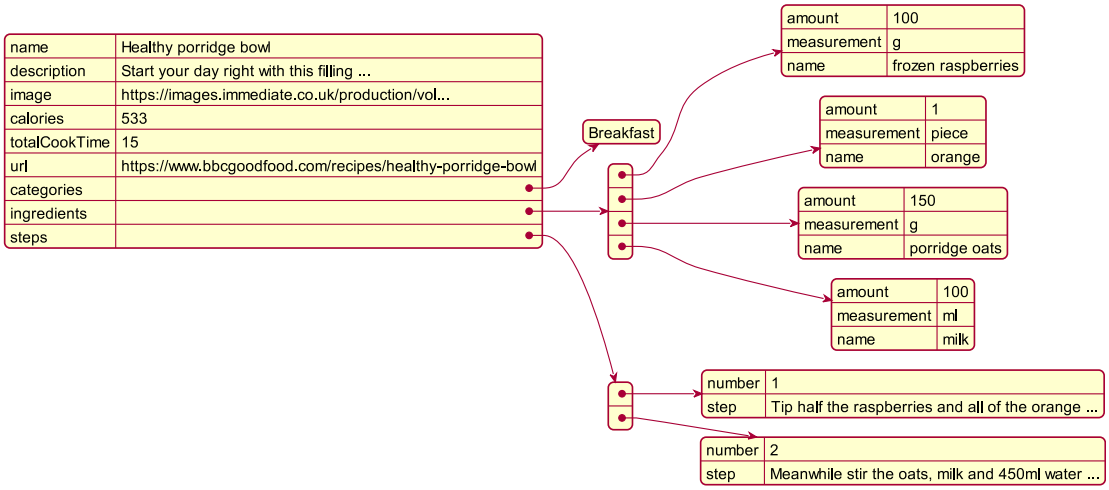
Adatmodellek

Az adatbázis felépítése különbözik a megszokott táblás módszertől, mivel a Firebase egy saját kifejlesztett módszerrel tárolja az adatokat, amivel gyorsabb a keresés és írás az adatok között. A szerkezete nagyon egyszerű, mivel csak egyszerű kollekciónkból és dokumentumokból áll, felépítése hasonlít a JSON fájltípusokra. Minden adat egy-egy dokumentumban tárolódik, és ezek a dokumentumok vannak elrendezve kollekcióban. Ennek a felépítésnek a hátránya viszont, hogy nehéz az adatbázis struktúrát frissíteni, mert csak az új adatoknál fog megjelenni, ha nem indítunk rá egy külön funkciót. Ezért is macerásabb a régi adatokhoz változtatások után az új adattagba értéket adni.

6.1. Recipes

Az adatbázisban az egyik legfontosabb adat, mivel erre épül az egész weblap témája. A receptek a feltöltés során automatikus azonosító ID értéket kapnak, ami a weblapon az URL címben az azonosításért is is felelős. Ezen belül tároljuk az adott recept adatait, ezek közé tartozik például: kalória, teljes főzési idő (percben mérve), recept neve, eredeti URL, stb. Ezek az alap adatok, csak simán kulcs érték párok.

Jelenleg még kulcs-tömb értékben tárolva az alapanyagok és lépések, viszont a jövőben ez lehet megváltozik subcollection-re a gyorsabb betöltés idő eléréséért. Ezek a kulcs-tömb párosok hasonlóak a korábbi típushoz, viszont itt egy azonosító értékhez egy tömbbnyi belső adat tartozik. Ez lehetővé teszi, hogy az alapanyagokon belül eltárolhassul egyesével a mennyiséget, mértékegységet és az alapanyag nevét. Valamint a recept elkészítésének lépésére el van tárolva a sorrendbeli szám és maga a lépés.



6.1. ábra. Receptek JSON felépítéséről diagram

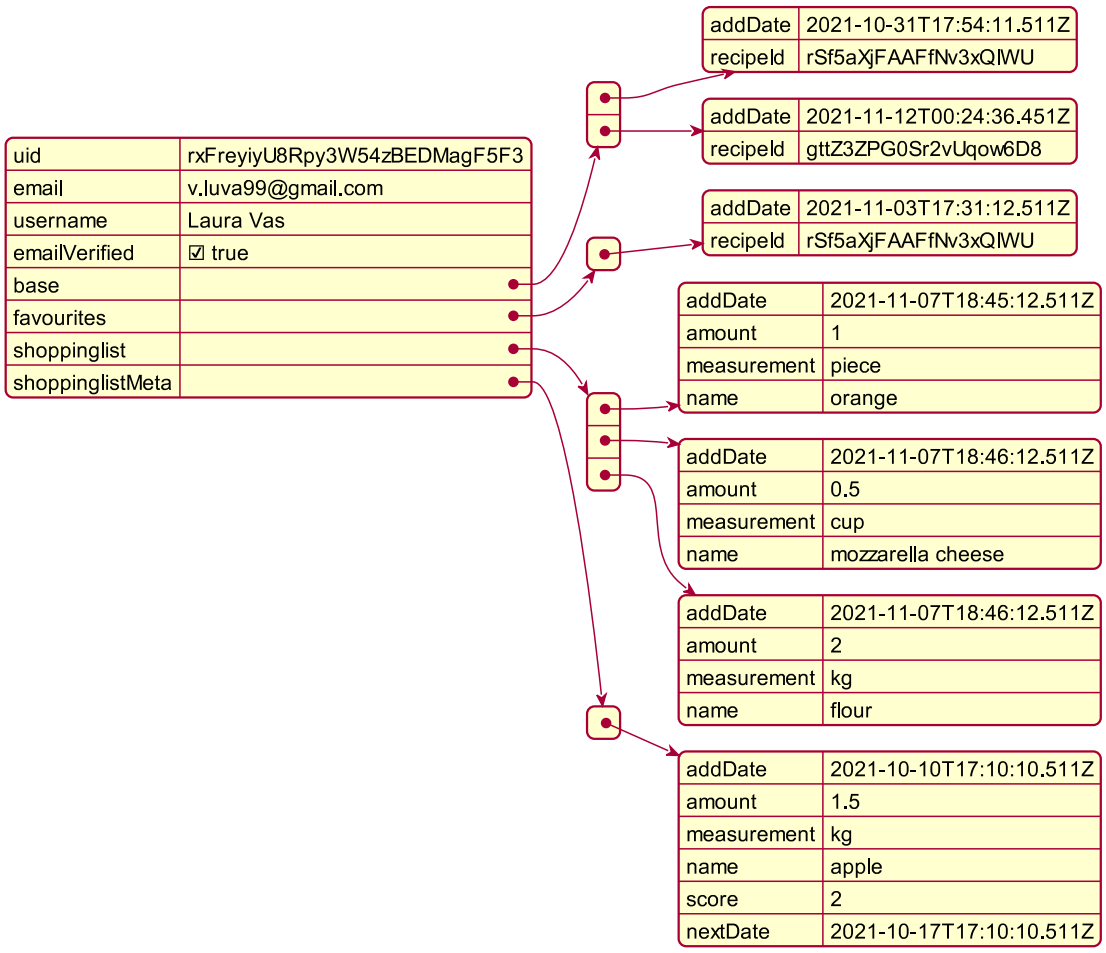
6.2. Users

Az adatbázisban a következő adathalmaz a felhasználók kollekciója. A regisztráció során létrejön egy új dokumentum az adatbázis felületen is. Itt tároljuk el az egyéb adatokat, amit a felhasználó az alap e-mail és jelszón kívül megad. Az egyéb adatok segíthetnek a jövőben személyre szabott recept ajánlót készíteni.

Itt a korábbi recepthez tartozó alapanyagok tömbhöz képest már egy subcollectionben vannak eltárolva az egyéb gyűjtemények, amit a felhasználó a böngészés során elmentett. Ezek közé tartozik a kimentett receptek listája, kedvenc receptek, bevásárló lista valamint a bevásárló listához tartozó ajánlónak a metaadatai.

A subcollection azért fontos ebben az esetben, mert mikor le akarjuk kérdezni a felhasználó kedvenc receptjei listát, akkor nem kell mindig az egész felhasználó adatot lekérdezni az összes többi kollekcióval együtt, hanem lekérdezhető ugyanúgy, mint a többi kollekció egyszerűen. Ezzel gyorsabb lesz a weblap, valamint spórolunk a fölösleges adatok kezelésével is. Amikor a felhasználó először elment a könyvtárába egy receptet, akkor a saját ID-ja alatt létrejön egy “base” kollekció, ami eltárolja, hogy mikor lett a listához adva a recept, valamit, hogy melyik lett kimentve a recept Id-ja alapján. Ugyanez igaz a “favourites” kollekcióra. Amint a felhasználó kimentett egy receptet a saját listájába, akkor van lehetősége ezt tovább elmenteni a kedvencek közé. Ez ugyanúgy működik mint a sima listába mentés. Amennyiben viszont a saját listájából törli a receptet, akkor a kedvencek közül is eltűnik.

A bevásárló listába a mentés dátumát és a pontos adatokat is elmentjük az alapanyagról,hogy utána az egész könnyen kilistázható legyen. A létrehozás dátuma a sorrendbe rendezés és a metaadat számítás miatt fontos. Ha a felhasználónak nem kell többé az adat, akkor törlődik a listából. Végűl a metaadat. ami kettő listához adás értékéből számolunk ki. Az ajánló jelenleg csak az alapanyag nevét ajánlja fel, viszont a mértékegység és mennyiség is el van tárolva, hátha a jövőben pontosabb számítást szeretnénk csinálni. Ezen kívül, mikor az adat létrejön, a “score” értéke 2. Ez egy érték, ami segít az ajánlás sikerességét követni, max értéke 5, minimum értéke 0. Végül a “nextDate” egy számot TimeStamp érték, ami a eredeti hozzáadás dátumából és az újjonnan hozzáadás dátumádól számolunk ki.

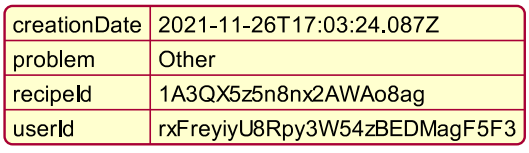


6.2. ábra. Felhasználók JSON felépítéséről diagram

6.3. Reports

Mivel a weblap úgy lett tervezve, hogy a közösség építi, és mindenki ugyanazokat a recepteket látja, ezért lennie kell egy lehetőségnek, ahol a felhasználók hibát tudnak jelezni. Tehát minden recept végén van egy felület, ahol jelzést lehet küldeni, ha hibás receptet találtak. A hiba bejelentés opciók előre be vannak állítva, így könnyebb lehet a jövőben megkeresni, hogy mi lehet a hiba az adott recepttel.

A “creationDate” a bejelentés elküldését jelzi. A “problem” pedig az opciókból választott a problémához legközelebbi lehetőség,amit a felhasználó választ. Valamint végül feljegyezzük, hogy melyik receptnél volt ez és hogy ki küldte. A “userId” megegyezik a korábban említett “uid” értékkel a “users” kollekcióból,mivel ez az érték, ami a bejelentkezés során az auth adott a felhasználónak.



6.3. ábra. Hibák bejelentése JSON felépítéséről diagram

7. fejezet

Tesztelés

A fontosabb funkciók, amik meghatározzák az alap témáját a weblapnak tesztelve lettek. A szerver oldali funkciók mind, az alap működések közé tartoznak,ezért a cloud function moduloknak be lett állítva automatikus tesztelés. Ennek segítségével, ha valami nagyobb változtatást csináltunk,mint például egy rendszer szintű frissítés a FireBase csomagnak, akkor egyszerűen le lehet tesztelni, hogy még minden jól működik-e. Ezt egyszerű unit tesztekkel oldottam meg. A teszteket az npm-ről szedett Jasmine csomaggal készítettem.

7.1. Recept importálás URL-en keresztül

Mivel ez a folyamat modulárisan lett felépítve, ezért minden egyes lépését egyszerű tesztelni. A modulok egyesével és a végeredmény is külön tesztelve van.

7.1.1. FetchUrlData

7.1.2. JsonLdExtractor

7.1.3. SeparateIngredients

Az alapanyag szétválasztás az egyik legkényesebb része a importálás folyamatnak, mivel ebben a részben egy string adatot kell megvizsgálni és különböző adattagokra osztani. Ehhez különböző regex formulák lettek kifejlesztve, hogy a különböző eseteket minél jobban és pontosabban tudjuk feldolgozni. Ehhez, egy egyszerű alap és egy komplexebb teszt készűlt.

Az első célja, hogy le ellenőrizze, hogy a tört szám karakter és egyéb tört szám változatok helyesen vannak átkonvertálva float értékre. A második teszt pedig egy tömböt kap értéknek, amiben rengeteg különböző eset van felsorolva olyan példákkal, amit élő importálás során kaphatunk és szélsőséges probléma lehet. Ezek közé tartozik a három alapeset, ahol vagy szépen visszaadja a mennyiséget, mértékegységet és az alapanyag megnevezését, vagy ha nem talál mértékegységet, de van szám, akkor berendezi, hogy mennyi datab alapanyagot talált, vgay végűl, ha nincs szám se, akkor csak simán visszaadja alapértelmezett módon. Ezen kívül le kell ellenőrizni azokat az eseteket, amik komplexé tették a regex teszteket például, a korábban említett tört karakter helyes észlelése és formázása.

7.1.4. ImageDownloader

7.1.5. CalorieCalculator

7.2. GetData

8. fejezet

Továbbfejlesztési lehetőségek

8.1. asd

Nyilatkozat

Alulírott ..... szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet ..... Tanszékén készítettem, ... 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.

Szeged, 2021. november 27. .....
aláírás

Alulírott ..... szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet ..... Tanszékén készítettem, ... 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 TVSZ 4. sz. mellékletében leírtak szerint kezelik.

Szeged, 2021. november 27. .....
aláírás

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani **X. Y-nak** ezért és ezért . . .

Irodalomjegyzék

[1] J. L. Gischer, The equational theory of pomsets. *Theoret. Comput. Sci.*, **61**(1988), 199–224.

[2] J.-E. Pin, *Varieties of Formal Languages*, Plenum Publishing Corp., New York, 1986.