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 a feladatom. A projekt a Firebase felhőalapú szolgáltatásait fogja használni. A fejlesztés során a legfőbb cél a más honlapokról való recept importálás megvalósítása. Az importálás egyik 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 étel receptek megjelenítésre és importálásra használható. Innentől kezdve a recept mindig étel receptet jelent.

- Feladat megfogalmazása:
  Az importálás funkció lehetővé teszi, hogy a felhasználók egy helyen gyűjtsék a receptjeiket. Ezen túl az is, 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 arra, 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 temérdek 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 kapcsolatban pedig, hogy 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 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

Sok hasonló jellegű weboldal létezik, mindnek kölünböző funkcióik, felhasználó körük, előnyeik és hátrányaik vannak. A jelen dolgozat projektje igyekszik a többi weblap hiányosságait elkerülni, ezzel szélesebb felhasználói kört magához vonzani.

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. A befektetett munka akkor tud kifizetődő lenni, ha valaki hosszú ideig használja.

A recept kezelő része csak manuálisan feltölthető, tehát nincs importálásra lehetőség. Rendelkezik bevásárlólista és “sufni” opciókkal is. Az otthon lévő alapanyagokat egyesével, tetszőleges részletességgel fel lehet venni a “sufniba”, ezzel leltározva, hogy milyen alapanyagok vannak otthon. Ezekről eltárolható adat az, 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

Ez a weblap azért szerepel, mert ez egy tökéletes példa egy átlagos, egyszerű receptes weblapra. A honlapon csak receptek és pár blog bejegyzés található regisztráció után is. Ez a weblap reprezentálja a legtöbb hasonló, csak blogként működő weboldalt.

Egy receptre kattintva láthatóak a hozzávalók, elkészítési javaslat, valamint az alapadatok, mint például az elkészítési 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 vesz esetleges allergiákat, vagy étrendeket ajánl 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. Mivel nincs importálási lehetőség, nem a legegyszerűbb használni, mert kézzel kell beírni az adatokat.

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 újakat létrehozni: manuálisan leírva; másolni egy már létező fájlból és beilleszteni; képről vagy scennelésről beolvasni; URL címmel importálni. Az importálás során nem tárolják el az egész receptet, ha más honlapról származik. Az "Our Pledge to Food Bloggers" leírja, hogy ez miért van így, viszont ez azt jelenti, hogy a teljes recept megtekintéséhez át kell navigálni az eredeti oldalra. Ezen túl a bevásárló listában nem adódnak össze az azonos nevű 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. 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 egyebet, beleértve a recept importálást, bevásárló listákat és étkezés tervezőt, csak az applikáción keresztül lehet elérni és szerkeszteni. A weben van lehetőség Google segítségével bejelentkezni, viszont az applikációnak nincs 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 lehetőség egy közös adatbázisból való keresésre. A felhasználónak mindent magának kell beszerezni.

Az importált recepteket könnyű módosítani, valamint rengeteg kis adatot megadni, hogy otthonosan lehessen használni a környezetet. Ám az applikációban nincs lehetőség közvetlenül a receptből a bevásárló listába rakni alapanyagokat, menüket összekészíteni vagy az étkezéstervezőt használni az ingyenes próbaverzió után.

1.6. Összefoglaló

A piackutatás során egyértelműen kiderült, melyek a legnépszerűbb funkciók, amiket minden program megvalósít. Érdekes eredmény például, hogy az emberek számára nem fontos, hogy maguknak tudják hosztolni a programot, hanem még akár fizetni is hajlandóak érte. Így amennyiben szükséges, a mi szoftverünket is lehet előfizetéshez kötni.

További megfigyelés, hogy a legtöbb weboldal nem támogatja a teljes körű recept importálást más forrásokból. Ennek lehet fejlesztési és jogi indoklása is, de biztos, hogy a felhasználók számára ez egy hasznos funkció lenne. Így biztosan megéri a projekt részeként megvalósítani. Érdekesség, hogy a BigOven tud kép alapján betölteni egy receptet, ez nagy segítséget jelent azon felhasználóknak, akiknek sok régi receptkönyvük van. Ehhez képfelismerési algoritmusokra van szükség, amik komplexitása igen nagy, igy a feljesztésük drága, ez jól magyarázza, miért csak egyetlen weblap támogatja a listából.

A piackutatás összes eredményét az 1.1 ábra foglalja össze.

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 piackutatásról

[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

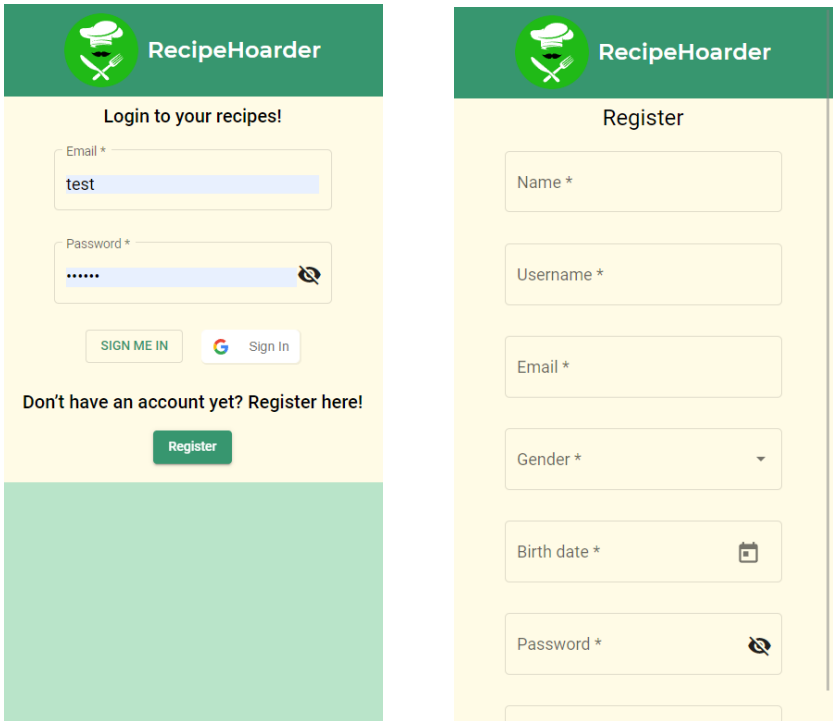
A következők összefoglalják a program főbb funkcionalitásait, azokat felhasználási esetekben (UseCase) csoportosítja. Ez alkotja a program funkcionális követelményét, azaz azokat a képességeket, amiket mindenféleképpen tudnia kell. Ezeket az elemeket a 2.1-es ábra grafikusan ábrázolja, majd a fejezet további része pontosítja és kifejti őket. Az ebben a fejezetben használt képek a már kész applikáció-ból származnak, és illusztrálják a megvalósított funkcionalitásokat.

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. Másrészt, 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").



(a) Bejelentkezés (b) Regisztráció

2.2. ábra. Bejelentkezés és regisztráció képernyőpépek

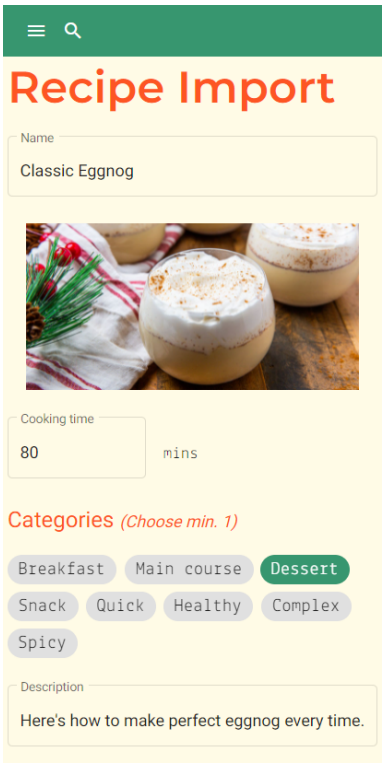
2.2. Receptek

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, például a legfontosabbnak megállapított recept importálást.

2.2.1. Receptek hozzáadása

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

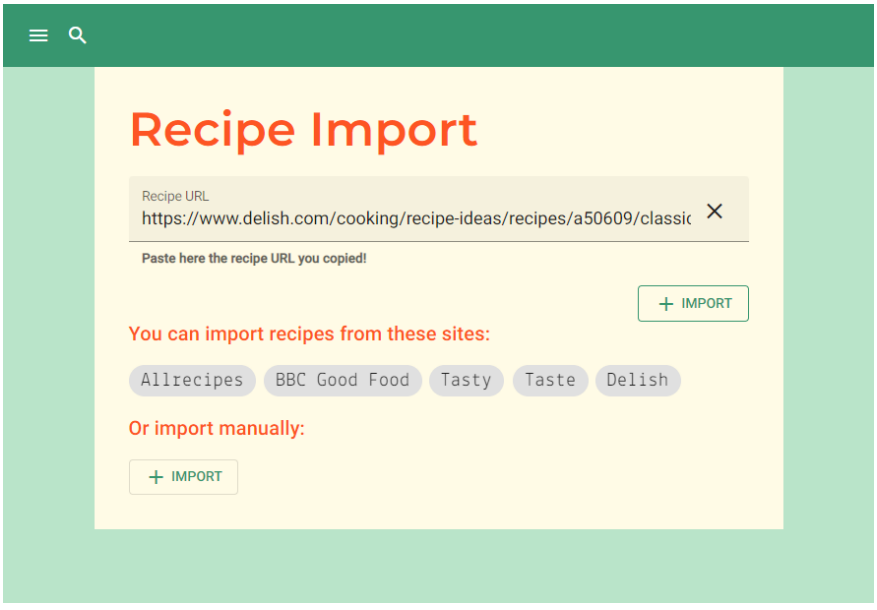
Egy recept feltöltésre két lehetőség is van. Az egyik az, hogy a felhasználó manuálisan tölti ki a recept paramétereit (lásd: 2.1-es ábra "Recept hozzáadás manuálisan"). Ez azokban az esetekben fontos a felhasználó számára, amikor a recept nem az internetről származik, hanem például nyomtatott receptkönyvből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



2.3. ábra. Egy recept a manuális recept feltöltés közben

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 kinentve hozzá adni a globális recept lisához.

Mindkét opció során a felhasználónak meg kell adnia, hogy a recept mely előre meghatározott csoportba tartozik (Pl.: Reggeli, Csípős, stb.). Majd a kimentés után a rendszer mind a két módszernél megpróbál kalória mennyiséget számolni a recepthez.



2.4. ábra. A desktopos verzióban a receptfelöltő oldal

2.2.2. Receptgyűjtemény kezelés

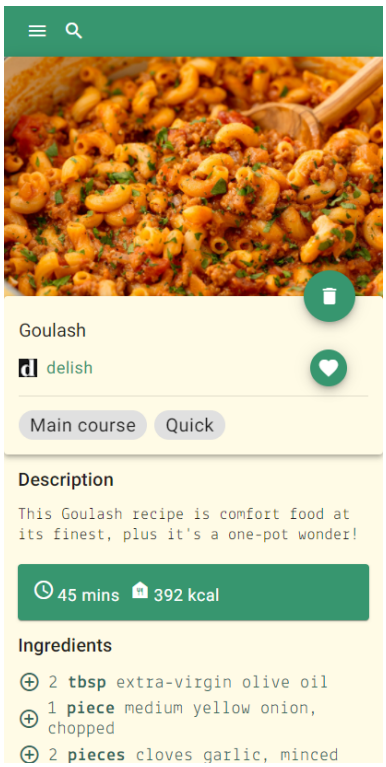
Ahhoz, hogy a weblap könnyen használható legyen, szükségünk van arra, hogy a felhasználó ki tudjon menteni recepteket egy saját gyűjteménybe. Ez nagyban megkönnyíti a recept későbbi megtalálását. Erre két különböző gyűjtemény áll a felhasználó rendelkezésére. Egyik a saját “receptkönyv”, amibe akárhány 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 először ki kell menteni a saját receptkönyvbe.

Természetesen 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”).

2.5. ábra. A desktopos verzióban a felhasználónak a "Receptes kézikönyv"-re képernyőkép

2.2.3. Recept megtekintés

Mivel ez egy recept gyűjtő webes alkalmazás, ezért egyértelműen lehetőséget kell adni 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”).

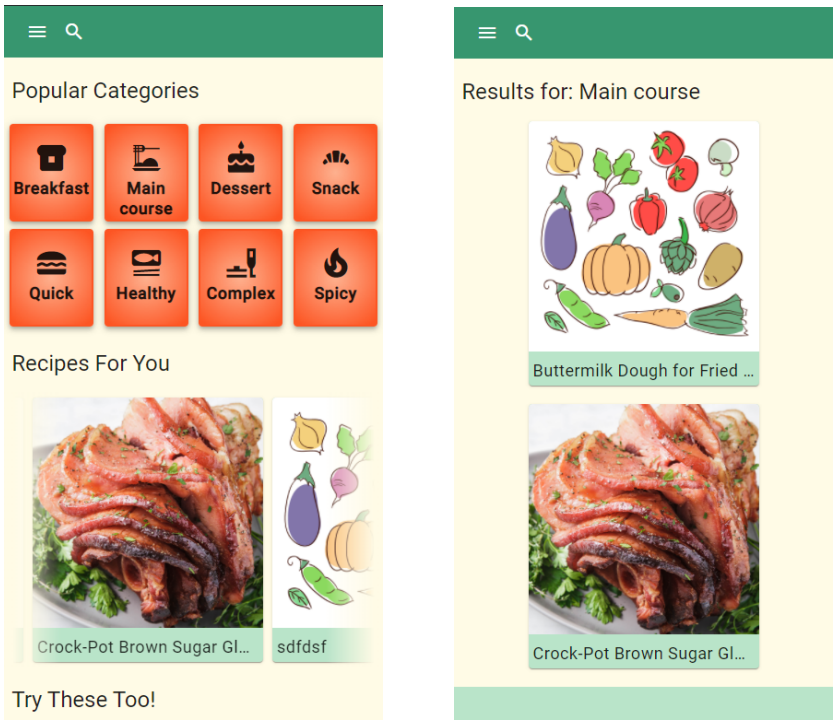


2.6. ábra. Egy recept képernyőképe

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, így egyszerűen kereshetők ez alapján.

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



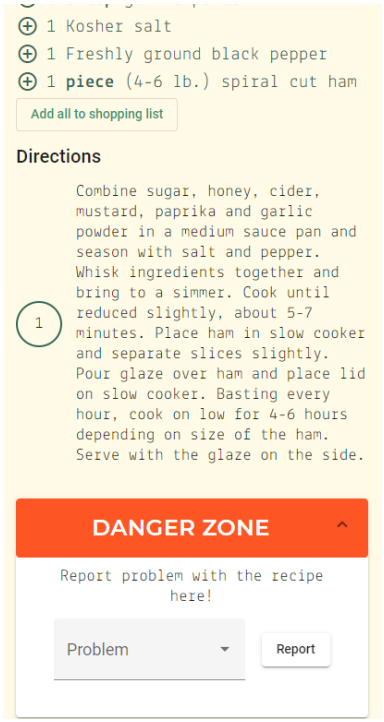
(a) Kereső opciók (b) Keresés eredmény

2.7. ábra. A kezdőoldalon található kategória szerinti szűrés

2.2.5. Hibás recept jelentése

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

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



2.8. ábra. Egy recept hibabejelentés képernyőkép

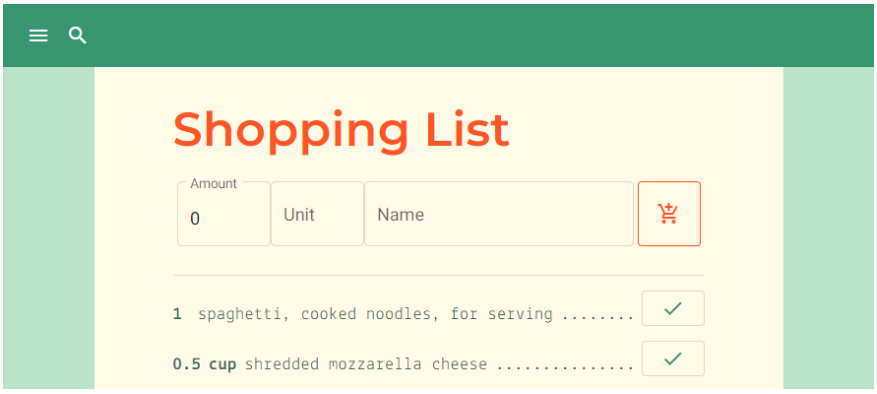
2.3. Bevásárló lista

Minden felhasználónak van saját bevásárlólistája, amibe össze tudja gyűjteni a számára szükséges alapanyagokat. Ez a lista 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 az adatokat, majd a listához adni. A másik módszer a recepteken belül, a felsorolt alapanyagokat is egyesével hozzá lehet adni, illetve a recepteknél lehetőség vagy az összes alapanyagot azonnal hozzáadni a listához.

Ez a 2.1-es ábrán a “Bevásárlólista” rész alatt, a “Bevásárló listához adás” résznek felel meg.

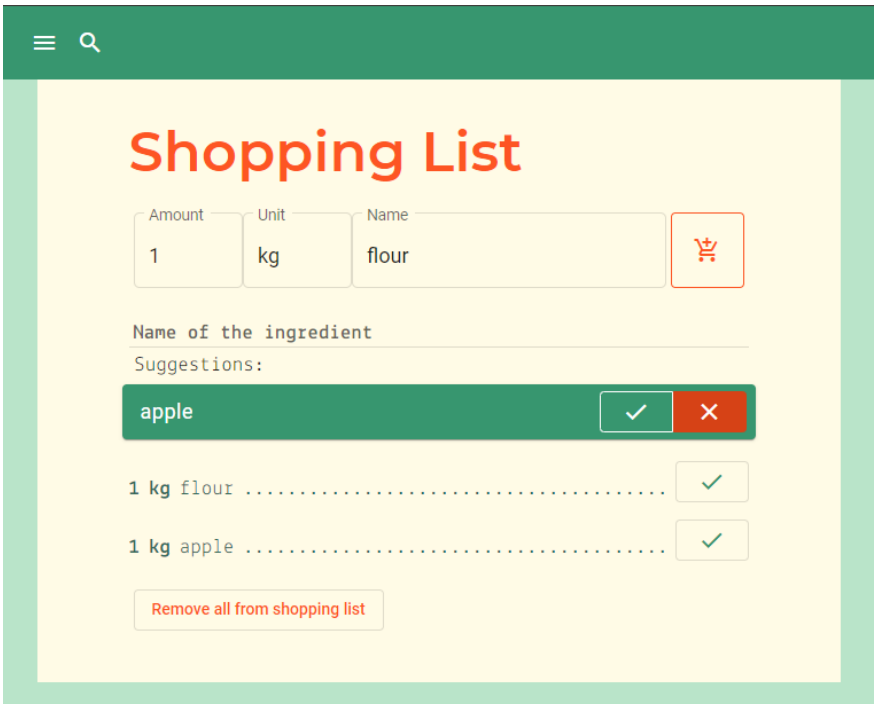


2.9. ábra. A bevásárló listához adás mezői

2.3.2. Bevásárló lista megtekintés

A bevásárlólista megjelenítésénél a korábban hozzáadott alapanyagok vannak felsorolva. A bevásárlólista az alapanyagokat a hozzáadásuk dátuma szerint jeleníti meg. Igy könnyen követhető, mikor egy új tétel 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.

Ez a 2.1-es ábrán a “Bevásárlólista” rész alatt lévő “Bevásárló lista megtekintése” résznek felel meg.

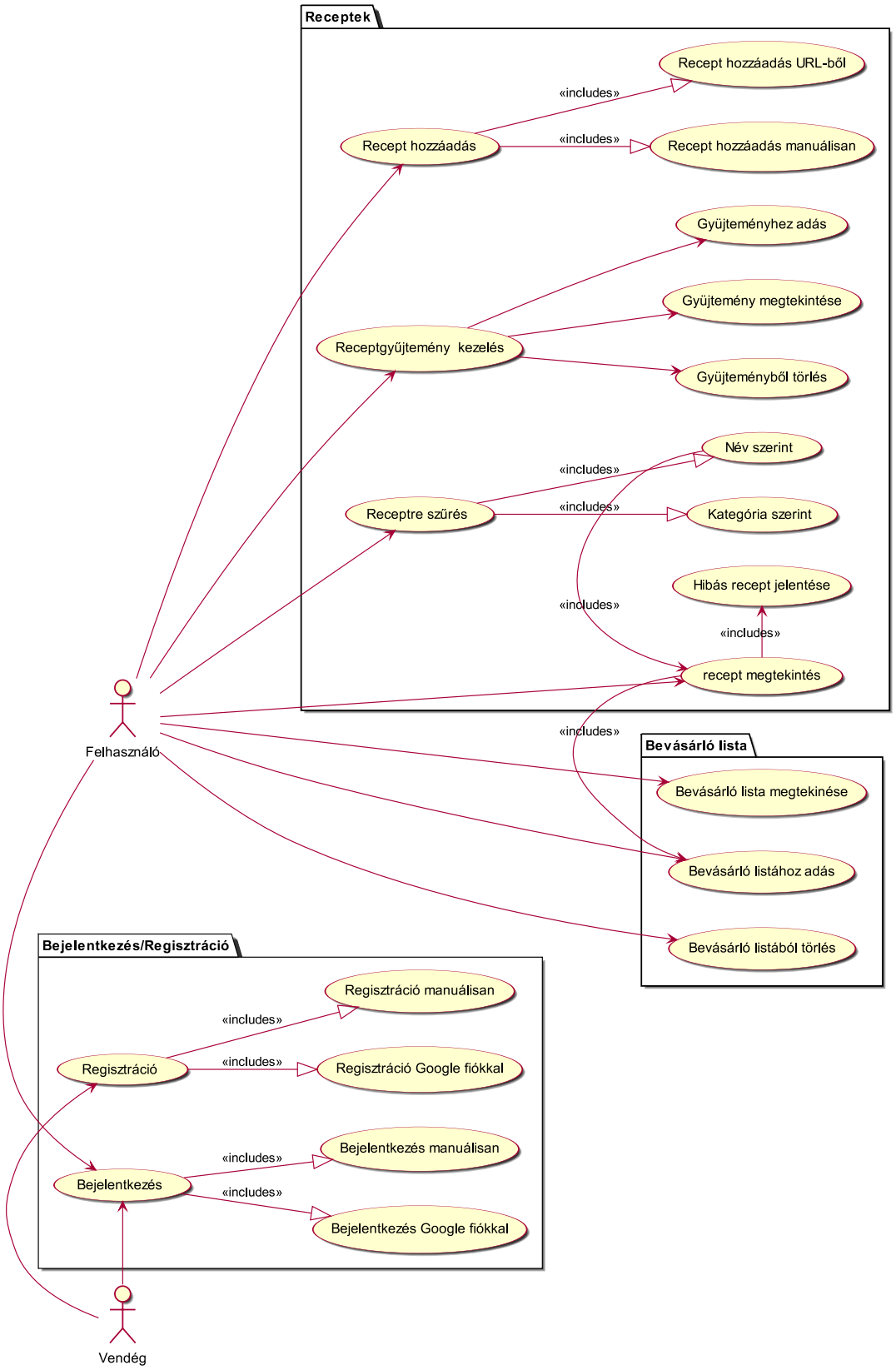


2.10. ábra. A bevásárlólistáról képernyőkép

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 tüntetni a listából. Másik verzió a lista ürítésére 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.

Ez a 2.1-es ábrán a “Bevásárlólista” rész alatt lévő “Bevásárló listából törlés” résznek felel meg.



2.1. ábra. UseCase diagram

3. fejezet

Felhasznált technológiák

Ebben a fejezetben a projekt által felhasznált számos technológiák kerülnek bemutatásra. A projekt legfontosabb technológiái az Angular és Firebase. Ezeknek a Google a fő fejlesztője, így várhatóan sokáig lesznek támogatva. A projekt az itt felsoroltakon túl, számos kisebb program könyvtárat is felhasznál, de ezek nem kerülnek részletezésre, mivel a projekt szempontjából nem kiemelt fontosságúak. A felhasznált csomagok teljes listáját a projekt forrásfáljai között lévő kettő `packag.json` file tartalmazza.

3.1. Angular

Az Angular egy TypeScript alapú Framework, amit eredetileg a Google fejlesztett ki, de most már nyílt forráskódú. Az Angular teljes újraírása a régi AngularJS-nek. Egyik legnagyobb előnye, hogy komponens alapú keretrendszer, ezért ennek 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 tervezése során is igyekeztem komponenseket kialakítani mind telefonos, mind desktopos kinézetben. A komponensekkel való munka megkönnyíti egy projekt fejlesztését, mivel akár saját fejlesztésből, akár külső könyvtárból szerzett elemekből 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 kellett kiegészíteni a már létező verziót, hogy illeszkedjenek 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

Eredetileg független cég által fejlesztett, de most már 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 miatt, arról nem is beszélve, hogy rengeteg dokumentáció létezik a fejlesztők számára.

Ebben a projektben elég sok helyen használom a Firebase funkcióit. Az egész weblap a Firebase Hosting platformon működik, ami egy statikus weblap hosztolási felület. Ez a GitHub-bal össze van kötve, ha a Main Branch-re feltöltés történik, akkor a Firebase-re automatikusan feltöltődik az új production verzió.

A FireStore Database-t használom a weblap adatainak tárolásához. Itt vannak eltárolva receptek, felhasználók, a felhasználókhoz különböző kollekciók és bevásárló listájuk illetve Metaadatok. A Cloud FireStore-on belül lehet beállítani az adatbázis Rule-okat. Ezt azt jelenti, hogy be lehet állítani, hogy különböző kollekciókat kik érhetnek el, milyen tevékenységek vannak engedélyezve nekik, valamint, hogy milyen jogokkal milyen adatbázis műveleteket lehet csinálni.

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

A legnagyobb biztonság érdekében a Firebase Auth-ot használom a weblap bejelentkezés és regisztrációs műveleteihez. Így lehetséges email és jelszó párossal 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 a szerver oldalon fut. Ez nem csak olcsóbb, mivel szerver oldali adatbázis lekérdezések ingyenesek a kliens oldalival szemben, hanem az importálás során felmerült problémát is megoldotta. Ez a probléma az, hogy egy honlap nem kérdezhet le adatot egy másikról kliens oldalon keresztül, mivel a cél weblapok CORS beállítása ezt nem engedi. Azon kívül, mikor a bevásárló listába új alapanyag kerül, akkor egy trigger lefut a szerver oldalon, aminek segítségével állítjuk elő az ajánlóhoz a metaadatokat.

3.4. PWA

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

A projekt esetében az Angular-t ki kell kiegészíteni JavaScript kóddal. Ha az eszköznek nincs internet elérése, akkor a script segítségével a weblapnak lokálisan elmentett változatát használja a böngésző. Ezzel elérhető, hogy valamilyen szinten használható legyen a weblap offline környezetben is. Szerencsére mind az Angular, mind an Firebase úgy lett kifejlesztve, hogy csak egy-egy helyen kell megadni 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 csak a már egyszer letöltött receptekre fog működni, de a bevásárló listához adás során, mikor az eszközön újra lesz internet, az egész szinkronizálódik, hogy ne legyen adatvesztés.

3.5. Schema.org

A schema.org egy kereső cégek által kezdett indítvány aminek része a Google, Yahoo és Bing!. A céljuk az, hogy az interneten való keresést megkönnyítsék az által, hogy a weblapok egy a számítógépek által könnyen értelmezhető formátumban is elérhetővé teszik a tartalmukat. Az indítvány összefoglalja, hogy milyen tartalmakat tudnak az egyes weblapok leírni és ezeknek milyen konkrét tulajdonságaik lehetnek. A weblapokban ezeket az adatokat több féle tényleges formátumban lehet eltárolni úgy, hogy azt a megjelenítés során nem rajzolja ki. Ezek közül a legelterjedtebb a JSON-LD, ami a HTML head részében elhelyezett JSON formátumú adatot jelent.

A projektben a legfontosabb a "Recipe" séma, ami leírja hogy a receptnek milyen tulajdonságait lehet megtalálni a JSON-LD kódban. Sajnos a standardizáció ellenére a különböző weblapok sok adatot eltérő módokon adnak meg, igy ez az adat további feldolgozásra szorul. A helyzetet tovább rontja az, hogy nem minden adatot kötelező megadni a sémában, így például elképzelhető, hogy egy receptnek hiányoznak a lépései.

3.6. Figma

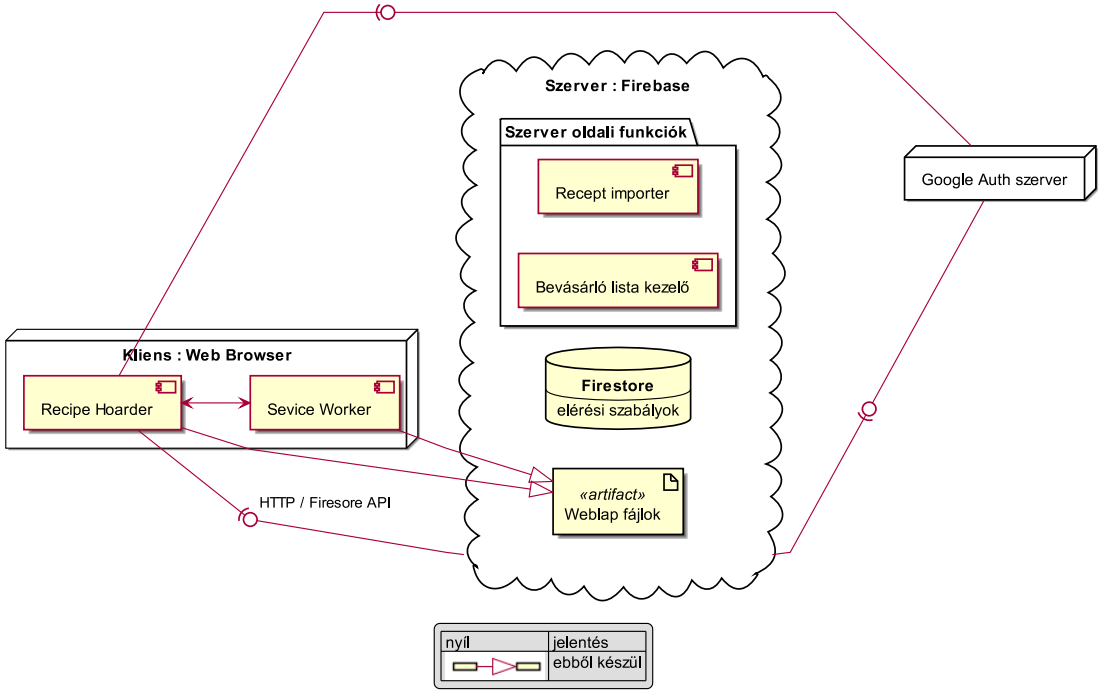
Egy vektorgrafikus szerkesztő, ahol felhasználói felület prototípusokat lehet tervezni. A program legfőképp webalapú, de letölthető desktopos verziója is. A program képes a tervezett lapok élő tükrözésére akár saját telefonon az applikáción keresztül, vagy csak az emuláló felületen keresztül. A Figma egy modern dizájn tervező app, amiben egyszerűen fel lehet használni a material design stílust és alap komponenseket, ezzel is segítve az Angular-ban is használt komponens alapú gondolkodást. Másik nagy előnye, hogy interaktívvá lehet tenni a tervezés során a gombokat, komponenseket, így akár a teljes felhasználói élmény kipróbálható. Ezen túl meg lehet nézni a kész dizájnoknak a pontos CSS stílus kódját is, ezzel is segítve a konkrét kódolás folyamatát.

Ez a program sokat segített a tervezésben, mivel a kész material design komponenseket tudtam próbálgatni anélkül, hogy először mélyebben kellet volna tanulmányoznom az Angular felépítését. Az egységes dizájn és téma tervezésében pedig segített konzisztensnek maradnom. Ez igaz mint a színekre, mind a tipográfiára.

4. fejezet

A rendszer magas szintű áttekintése

Mint minden más webes alkalmazás, ez a projekt is egyértelműen felbontható két fő egységre a szerver és a kliens oldal mentén. A kliens oldal azt a kódot és egyéb fájlokat jeleni, amiket az interneten keresztűl a felhasználó böngészője tölt le és futtat. Míg a szerver oldali kód az egy központi helyen fut és a hosszabb időt igénybe vevő számításokat, illetve az adatbázis elérését végzi.



4.1. ábra. Diagram a kilens és szerver oldali összeköttetésről

4.1. Szerver oldal

A projekt esetében a szerveroldali funkcionalitások a Firebase segítségével lettek megvalósítva (lásd: 4.1. ábra Firebase eleme). Ez nagyban megkönnyítette a munkát, mivel a szerver tényleges működtetését a Firebase végzi. Az adatbázis szerepét a Firebase Firestore tölti be (lásd: 4.1. ábra Firestore eleme), ami egy NoSQL alapú adatbázis (lásd: 6. fejezetben részletesebben). Ez egy biztonságos, kliens oldalról is használható API-val rendelkezik, aminek segítségével az adatbázis műveletek elvégzéséhez nincs

szükség saját szerver oldali kód írására. A weblap statikus fájljainak kiszolgálására a FireBase Hosting van használva, aminek segítségével nincs szükség saját Apache vagy NGINX szerver üzemeltetésére. Ahol szükséges volt a szerver oldalon saját kódot futtatni, ott a FireStore által nyújtott Cloud Functions megoldást használtam.

4.1.1. Firebase functions

A Firebase functions egy olyan felhő alapú megoldás, ahol a fejlesztőnek csak a ténylegesen futtatni kívánt kódot kell megadnia. A virtuális gép, az azon futó operációs rendszer és a program elindítása mind a platform által van kezelve. Ez nagyban megkönnyíti a fejlesztést, mivel nem kell plusz időt fordítani a szerver karbantartására. A Firebase JavaScript/ TypeScript -ben írt kódot tud futtatni. Mivel a kliens oldal is TypeScript-ben készűlt, ezért bizonyos funkciókat, illetve adatstruktúrákat a két komponens között meg lehet osztani.

A fejlesztő álltal megírt programokat többféle események is elindíthatják. Leggyakoribb példák erre a bejövő HTTP kérések vagy az adatbázisban történt változások. Ezek közül a projekt mind a kettőt használja.

A szerver oldalon két fő funkcionalitást kellett megvalósítani. Első és legfontosabb a receptek importálásáért felelős Cloud Function. Ez a kliens által küldött speciális HTTP üzenetre indul el és végzi el a receptek letöltését és értelmezését. A második funkcionalitás a bevásárló lista ajánlóhoz metaadatok előállítását végzi, amit minden bevásárlólistához adáskor elvégez.

4.1.2. Google Auth

A Google felhasználói fiókkal való belépéshez természetesen a Google által üzemeltetett azonosító szerverrel kell kommunikálni a mind a kliensnek, mind a szervernek. A kliens a kommunikáció során elvégzi a felhasználói adatok megszerzését, majd ezeket a szerver is hitelesíti. Mindezt a Firebase Auth szolgáltatása autómaikusan elintézi. Az Auth szolgáltatás előnye, hogy sok más platformot is támogat például Facebook és Twitter.

4.2. Kliens oldal

A kliens a projekt esetében a felhasználó web böngészőjében futó kódokat jelenti. ez két fő egységre választható szét. Az egyik a ténylegesen megjelenő weblap kódja, amely tartalmazza a megjelenítéshez szükséges HTML, CSS és JavaScript fájlokat. A másik komponens a weblap offline elérését biztosító Service Worker. Ami a webböngészőben, mint háttérfolyamat fut.

4.2.1. Weboldal

A weboldalt az Angular fordítási folyamat eredményeképp létrejövő fájlok alkotják. Ezek a fájlok tartalmazzák a lefordított TypeScript kódot, ami egy JavaScript fájlba lett generálva, illetve tartalmazza még a hasonló módon összesített stílusokat egy CSS fájlban. Ezeket a fájlokat a kliens a Firebase Hosting szerveréről tölti le, ami egy statikus fájlkiszolgáló rendszer. Azért lett ez választva, mivel a Google szervereit használja, így magas szintű megbízhatóságot biztosít. Továbbá automatikusan jól skálázódik, ha

a weblap forgalma hirtelen megnövekszik. Ezen tulajdonságai miatt, ez egy olyan platformot biztosít, ami bármekkora felhasználói bázis mellett jól használható.

A fordítás kimeneti fájljai autómatikusan feltöltésre kerülnek, amikor a GitHubrara egy új verzió kerül fel. Ezzel biztosított, hogy a feltöltés közben emberi hiba nem történhet.

4.2.2. Service Worker

A Service Worker egy olyan relatív új funkciója a webböngészőknek, ami régi böngészőkön például Internet Explorer egyáltalán nem támogatott. A funkció lényege az, hogy megadhatók egy olyan JavaScript kód, ami tudja kezelni a weblap által indított HTTP lekérdezéseket. Ennek segítségével amennyiben nincsen nincsen internet kapcsolat, a Service Worker képes mégis visszaadni adatokat. Így, amennyiben a lekérdezések eredménye a LocalStorage-ba ki lett mentve, akkor azokat internet hiányában is vissza lehet kapni.

Ezt a funkcionalitást az Angular által szolgáltatott ServiceWorker megvalósítja, így ehhez a fejlesztőnek plusz munkát nem kell végeznie. Amennyiben az applikáció komplexebb adatok tárolását igényli, szükség lehet ennek az újraírására. Ezen projekt számára a beépített Service Worker megfelel, így nincs szükség saját kód írására. Mivel a Service Worker csak a második újratöltésre szolgáltat új kóddal, ezért egy figyelmeztető ablak lett implementálva, ami jelzi a felhasználónak, ha új verzió érhető el és lehetőséget ad az új verzió betöltésére.

5. fejezet

Architektúra

Ebben a fejezetben a 4-es fejezetben leírt komponensek kerülnek részletes leírásra.

5.1. Kliens

A kliens oldal felépítését nagyban meghatározza az, hogy Angular keretrendszer van használva a projektben, ami sok programozási mintát megkövetel. Ezek segítenek az újrafelhasználható, illetve megbízható kódot írni.

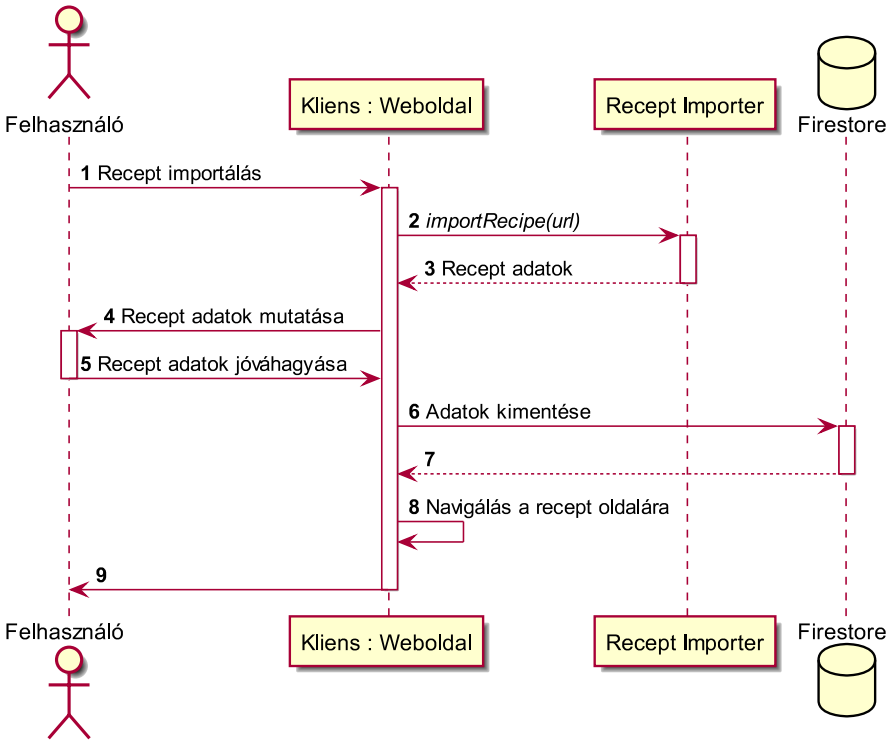
5.1.1. Angular által nyújtott szolgáltatások

Az Angular által előírt minták közül a legfontosabb a komponens minta. Mivel így az általunk készített grafikus komponenseket az oldalon akárhány helyen fel tudjuk használni. Ezek a komponensek a fejlesztés alatt úgy viselkednek, mintha tényleges HTML elemek lennének, majd futtatáskor az Angular rendszere helyettesíti be az általunk megadott HTML szegmenseket.

Továbbá a keretrendszer lehetőséget biztosít service-k írására és használatára. Ezek olyan speciális osztályok, amelyeket a komponensekből el tudunk érni és komponenshez szorosan nem köthető funkcionalitásokat lehet bennük megvalósítani. Az elérésükhöz az Angular dependency injection-t biztosít. Ennek a lényege az, hogy a felhasználni kívánt service-t mint bemeneti paraméter van feltéve az osztály konstruktorában. Így, amikor az Angular-nak létre kell hoznia az osztályt, automatikusan átadja neki a service egy példányát. Ezt a megoldást sok más környezetben például JAVA-ban is szokás használni.

5.1.2. Recept importálás

A recept importálás folyamatot kliens oldalról mindig a felhasználó kezdeményezi. Ezt a link bemásolásával és az import gombra való rányomással teszi meg. Ekkor a kliens a megadott URL címmel meghívja a felhőben futó Cloud function-t, majd vár amíg az visszaküldi az elkészült recept adatokat. Ezután a felhasználónak lehetősége van azt javítani. Majd, ha azt jóváhagyta, akkor a kliens ismét a szerver oldalhoz fordul. Ezúttal az adatbázisba tölti fel a receptet. Ezt a folyamatot az 5.1. ábra mutatja be. A szerver oldali folyamatokat később az 5.2.1. szekcióban vannak részletezve.



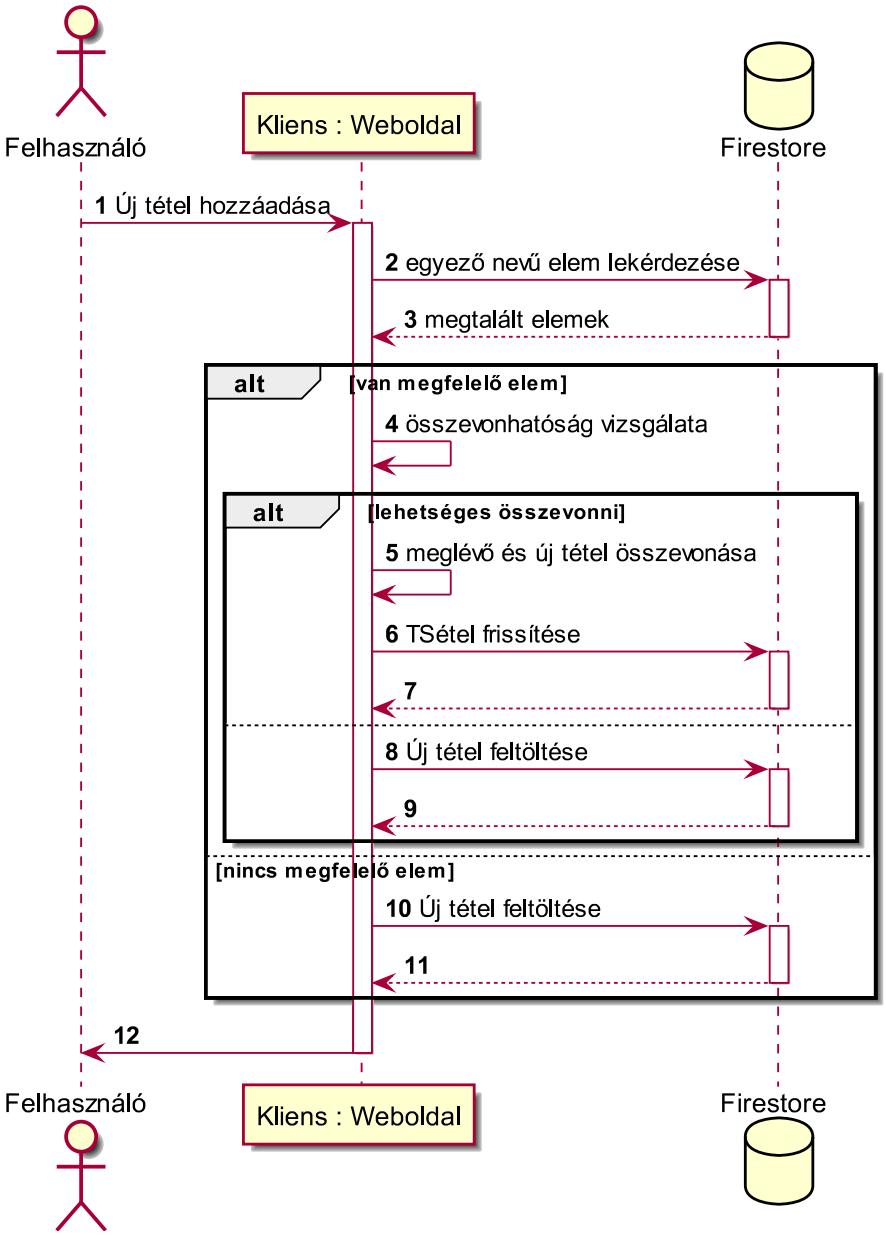
5.1. ábra. A recept importálás a kliens szemszögéből

5.1.3. Saját kollekcióba mentés

A receptek saját kollekcióba való mentése egy egyszerű folyamat. Amikor a felhasználó rányom a megfelelő gombra, a kliens a felhasználó adatbázisában lévő kollekcióba tölti fel a kiválasztott recept címét.

5.1.4. Bevásárló listához adás

A bevásárlólistához adáskor a kliensnek sok feladata van, ezt a folyamatot a 5.2. ábra mutatja be. Ezt az esemény sorozatot is a felhasználó indítja el azzal, hogy a megfelelő gombra rányom (`1`. átmenet az ábrán). Ezután a kliens az adatbázisból lekérdezi az ugyanilyen nevű tételeket. Amennyiben van ugyanilyen nevű tétel, akkor a kliens megpróbálja ezeket összevonni. A tételek mértékegységét is átváltja, ha szükséges. (`4`) Amennyiben az összevonás lehetséges, akkor az adatbázisban frissül a már meglévő tétel (`6`). Azonban, ha nincs megfelelő nevű tétel még kimentve vagy nem lehet őket összevonni, akkor új tételként kerülnek feltöltésre (`8` és `10`).



5.2. ábra. A recept importálás a kliens szemszögéből

5.2. Szerver

5.2.1. Recept importálás

A recept importálás nagy része a szerveren történik. A szerver pipeline (csővezeték) architektúrát alkalmaz. Ennek a lényege az, hogy a kérést lépésekben dolgozzuk fel. Minden egyes lépés a csővezeték egy egy állomása. A Pipeline osztály egy példányába bele kell rakni a modulokat, amiket egy-egy futtatás során a felépített sorrendben egymás után fog futtatni. Ennek a felépítésnek az előnye, hogy kevés a modulok közötti függőség így azok egymástól függetlenül is tesztelhetők. Azon kívül még akár külön külön is futtathatóak.

Amikor a kliens elindít egy lekérdezést, a cloud function belépési pontja rögtön meghívja a `recipesPipeline`-t ami sorban lefuttatja a modulokat.

- **FetchUrlData**
  Ez a modul fut le legelsőnek és ez felelős a HTML letöltéséért a megadott URL címről. Ehhez kettő külső csomagot használ fel. Elsőként az `axios` csomagot használja a tényleges HTTP lekérdezés futtatására. Majd a `cheerio` csomag segítségével a megkapott HTML fájlt értelmezi. A `cheerio` nagy előnye, hogy a CSS-ben megszokott módon tudunk kiválasztani elemeket a megadott HTML-ből.

- **JsonLdExtractor**
  Ebben a modulba kerülnek a már említett JSON-LD adatok feldolgozásra. Itt megpróbálja a modul megtalálni azt a JSON szekciót, ami a Schema.org-os `Recipe`-t ír le. Majd, ha ezt megtalálta, akkor ezt a különböző weblapok által használt változtatásokat figyelembe véve dolgozza azt fel.

- **SeparateIngredients**
  Ez a module az előzőleg megszerzett összető neveket választja szét. Erre több RegEx kifejezést használ. Ezek a sokféle emberek számára könnyen érthető formátumokat dolgozzák fel. Például külön eset dolgozza fel azt, amikor az összetevő `<szám> <mértékegység> <megnevezés>` (pl.: 2 kg apples) alakban van, a `(<tól> - <ig>) <mértékegység> <megnevezés>` (pl.: 1 - 2 kg apples) alakhoz képest. Ez egy kifejezetten összetett lépés mivel fel kell tudni dolgozni a Unicode Vulgar Fraction karaktereit is. Ez azért fontos, mert elég sok weblap előszeretettel használja ezeket a tört szám jelöléseket.

- **CalorieCalculator**
  A kalória számlálást valósítja meg, ehhez az USDA által szolgáltatott API-t használja. Az API-ról visszakapott adatokat fel kell dolgozni és kiszámítani az alapján, hogy mennyi kalória van a receptben megadott mennyiségekhez. Ehhez a modulnak át kell tudni váltania a recept által megadott mértékegységek és az API által szolgáltatott mennyiségek között. Ez jelenleg a legerősebb modul, mivel minden alapanyaghoz web lekérdezést kell elindítani. Ezt a későbbiekben lokális adatbázisban eltárolt cache segítségével lehetne gyorsítani.

5.2.2. Bevásárlólista metaadat kezelés

A bevásárló listában az ajánlások funkcióhoz el kell tárolni azt, hogy a felhasználó milyen gyakorisággal vásárol egy adott terméket. Ehhez egy metaadat kollekció van fenntartva (Lásd részletesebben Felhasználók eltárolása). Ennek a listának a fenntartása egy cloud function feladata. A funkció minden alkalommal lefut, ha úgy tétel felvételre került a bevásárlólistába vagy egy a listában lévő tétel került módosításra. Ezekben az esetekben a program kiszámítja az előző hozzáadástól eltelt időt, majd ezt átlagolva eltárolja a `deltaTime` mezőben. Ezen érték alapján kiszámolja a következő alkalmat, hogy mikor kell a következő alapanyag javaslatot feldobni a felhasználónak.

6. fejezet

Adatmodellek

Az adatbázis felépítése különbözik a megszokott táblás módszertől, mivel a Firebase a NoSQL módszerrel tárolja az adatokat, amivel gyorsabb a keresés 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. A szokványos SQL adatbázisoktól eltérően a FireStore-nak nincs fix sémája. Ez azt jelenti, hogy egy kollekción belül a dokumentumoknak a felépítése nem rögzített. Így előfordulhat az, hogy egy dokumentumnak a többihez képes több vagy kevesebb adata van. Emiatt a dokumentumok frissítésére külön energiát és figyelmet kell fordítani. Ezt a legtöbb esetben Cloud Functions futtatásával szokás megoldani.

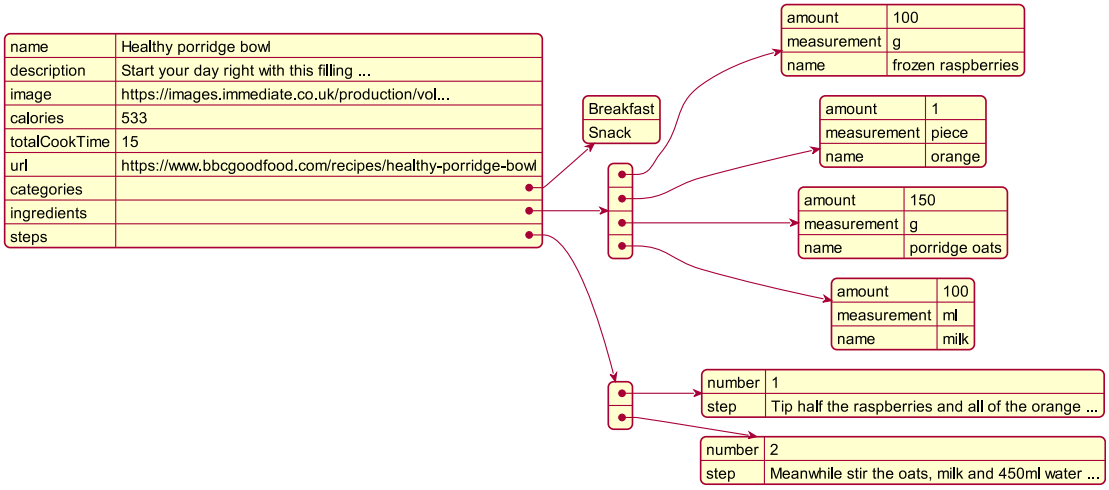
A FireStore-ban a dokumentumok olyan felépítésű adattárolók, amelyekben egy tetszőleges szöveges kulcshoz tudunk valamilyen típusú adatot rendelni. Ezek az adatok lehetnek:

- **Szöveg** (string), ami UTF-8 kódolású és maximum 1MiB méretű lehet
- **Szám** (number), tetszőleges pontosságú számérték
- **Igaz/Hamis** (boolean), igaz vagy hamis
- **Tetszőleges adatstruktúra** (map), ez egy a JSON formátumhoz nagyban hasonlító kötetlen adatmező
- **Üres** (null)
- **Időpont** (timestamp), ez egy dátumot és másodpercre pontos időpontot jelez
- **Földrajzi koordináta** (geopoint), hosszúsági és szélességi fokot egy mezőben tároló típus
- **Referencia** (reference), másik dokumentumra mutató elérési út
- **Tömb** (array), egy különleges tároló, ami az eddig felsorolt összes típusból képes sokat eltárolni

6.1. A receptek eltárolása

Az adatbázisban a legfontosabb adatok a receptek, mivel ezekre 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. A recepteket egy kollekcióban tároljuk és ezen belül a dokumentumok az adott recept adatait tartalmazzák, ezek közé tartozik például: kalória, teljes főzési idő (percben mérve), recept neve, eredeti URL, stb. Ezek az alap adatok csak simán kulcs érték párok.

Alapanyagokon belül eltároljuk az alapanyag nevét, mennyiségét és mértékegységét. A recept elkészítésének lépéseihez tároljuk a sorrendbeli számát és a lépés leírását. Jelenleg még kulcs-tömb értékben vannak tárolva az alapanyagok és lépések, viszont a jövőben megváltoztathatjuk alkollekcióvá a gyorsabb betöltési idő eléréséért.



6.1. ábra. Minta a receptek adatbázis felépítéséről

6.2. Felhasználók eltárolása

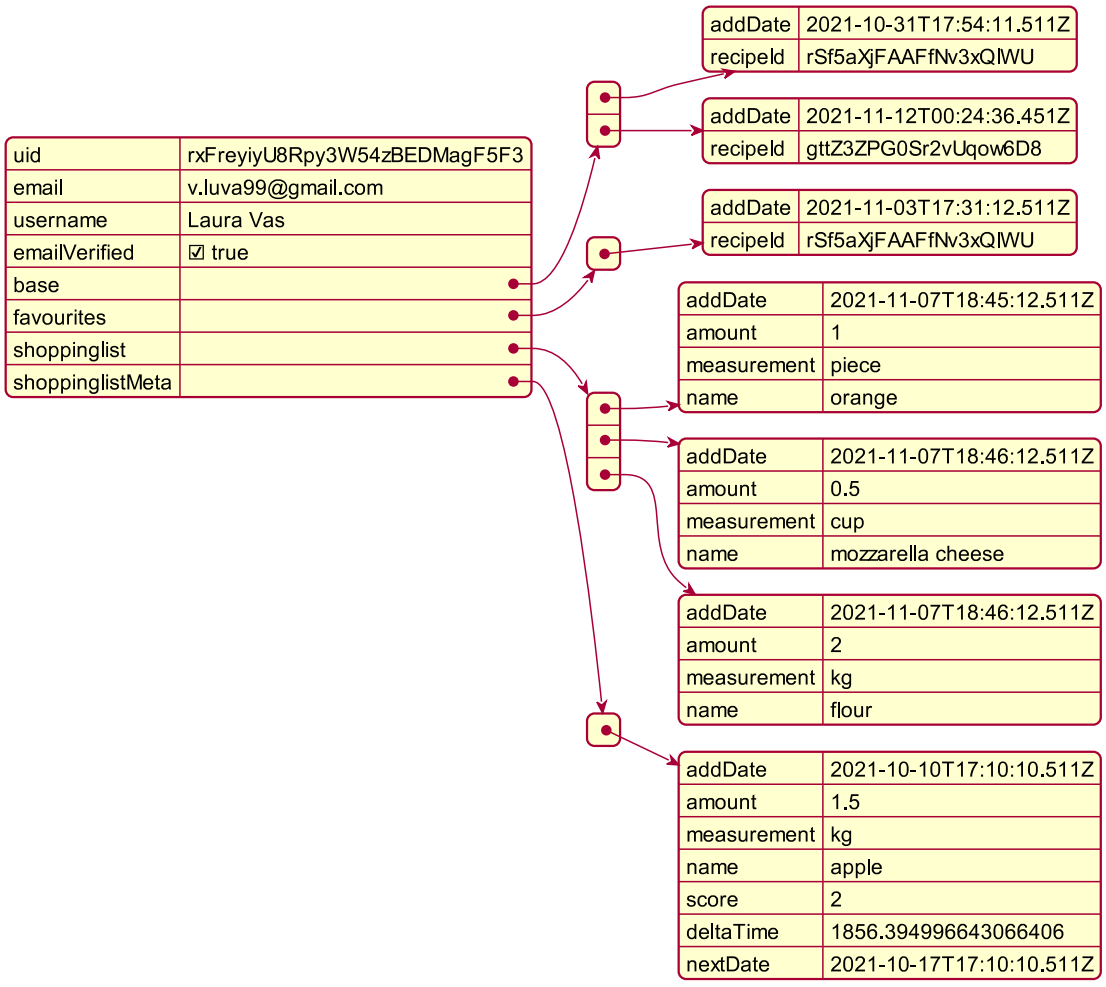
Az adatbázisban a felhasználókat egy kollekcióban tároljuk. A regisztráció során létrejön egy új dokumentum az adatbázisban. Itt tároljuk el az egyéb adatokat, amit a felhasználó az alap e-mail és jelszón kívül megad. Az egyéb adatok segíthetnek a jövőben személyre szabott recept ajánlót készíteni.

Minden felhasználóhoz továbbá eltároljuk a Firebase Auth által adott “UID” azonosítót, ami garantáltan egyedi. Itt a korábban említett kulcs-tömb típushoz képest már alkollekciók lettek bevezetve. Ezekben az alkollekciókban tároljuk el a felhasználó által elmentett recepteket, illetve hasonló módon tároljuk el a bevásárló lista adatait is.

Az alkollekció azért fontos ebben az esetben, mert ha le akarjuk kérdezni a felhasználó kedvenc receptjeit, akkor nem kell mindig az egész felhasználói adatot lekérdezni, hanem lekérdezhető ugyanúgy, mint egy egyedülálló kollekció. Ezzel gyorsabb lesz a weblap, valamint spórolunk a fölösleges adatok kezelésével is. A felhasználó által a könyvtárába elmentett receptek, a felhasználó dokumentuma alatt lévő “base” alkollekcióban kerülnek tárolásra. Ezek a recept ID-jét és a kimentési időpontot tartalmazó dokumentumokkal vannak eltárolva. Ugyanez igaz a kedvencek listára, ami a “favourites” alkollekcióban van tárolva. Amint a felhasználó kimentett egy receptet a saját listájába, akkor van lehetősége ezt tovább elmenteni a kedvencek közé. Ez ugyanúgy

működik, mint az egyszerű listába mentés. Amennyiben viszont a saját listájából törli a receptet, akkor a kedvencek közül is eltűnik.

A bevásárló listába az alapanyagokról a mentés dátumát és a pontos adatokat is elmentjük. A létrehozás dátuma a sorrendbe rendezés és a metaadat számítás miatt fontos. Ha a felhasználó úgy dönt, hogy a lista elem már nem aktuális, akkor az törölhető a listából. Továbbá a bevásárló listáról el vannak tárolva meta adatok, amiket egy-egy termék kettő listához adása között eltelt időből számolunk ki. Az ajánló jelenleg csak az alapanyag nevét ajánlja fel, viszont a mértékegység és mennyiség is el van tárolva, ezt a jövőben pontosabb számításokra lehet felhasználni. 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 számolt TimeStamp érték, ami a eredeti hozzáadás dátumából és az újonnan hozzáadott alapanyag dátumából számolunk ki.



6.2. ábra. Minta egy példa felhasználó lehetséges adatairól

6.3. Recept hibajelentések tárolása

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 és ezeket a hibákat az adatbázisban el kell tudni tárolni. Tehát minden recept végén van lehetőség visszajelzést küldeni, ha hibás receptet találnak. A bejelenthető

hiba típusai fixek, így könnyebben lehet a jövőben megkeresni, hogy mi lehet a hiba az adott recepttel.

Ezeket a bejelentett hibákat egy kollekcióban tároljuk, ahol minden dokumentum egy-egy bejelentett hibát tárol. A “creationDate” a bejelentés idejét jelzi. A “problem” pedig az opciókból választott problémát legjobban leíró lehetőség, amit a felhasználó választ. 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.

creationDate	2021-11-26T17:03:24.087Z
problem	Other
recipeId	1A3QX5z5n8nx2AWAo8ag
userId	rxFreyiyU8Rpy3W54zBEDMagF5F3

6.3. ábra. Egy minta a hiba bejelentésre

7. fejezet

Tesztelés

A fontosabb funkciók, amik meghatározzák a témáját a weblapnak tesztelve lettek. A szerver oldali funkciók mind, az alap működések közé tartoznak, ezért a cloud function moduloknak automatikusan futtatható tesztek lettek készítve. Ezeknek a segítségével könnyen észrevehető, ha egy változtatás a kódban hibás működéshez vezet. Ennek a megvalósítására unit és integráció tesztek lettek írva. Ezek különböző modulokat és az egész importáló pipeline-t tesztelik.

A tesztelés a Jasmine könyvtárat használja. Ez egy elterjedt az Angular által is javasolt teszt környezet. Az előnye az, hogy gyors és könnyedén hozzáadható a projekthez.

7.1. Import modulok

Mivel a recept importálás folyamat modulárisan lett felépítve, ezért minden egyes lépését egyszerű tesztelni. A modulokból a lényegesebbek egyesével és a teljes funkció külön tesztelve van.

7.1.1. SeparateIngredients

Az alapanyag szétválasztás az egyik legkényesebb része a importálás folyamatnak, mivel ebben a részben egy szöveges adatot kell megvizsgálni és különböző adattagokra osztani. Ehhez különböző RegEx formulák lettek kifejlesztve, hogy a különböző eseteket minél jobban és pontosabban lehessen feldolgozni.

Ezekhez egy egyszerű alap és egy komplexebb teszt készűlt. Az első célja, hogy le ellenőrizze a tört szám karakter és egyéb törtszám változatok helyesen vannak átkonvertálva float értékre. A második teszt pedig egy tömböt kap értéknek, amiben rengeteg különböző alapanyag és eredmény páros van felsorolva. Ezek olyan példák, amik élő importálás során kaphatóak és szélsőséges eseteket illusztrálnak. Ezek közé tartozik a három alapeset, ahol az első és legjobb eset amikor szépen visszaadja a mennyiséget, mértékegységet és az alapanyag megnevezését. Második lehetőség lehet az, amikor nem talál mértékegységet, de számot igen, tehát úgy kell visszaadnia mintha darabokban lenne mérve. Végül, ha számot sem talál a feldolgozandó szövegben, akkor csak simán visszaadja azt mintha az egész leírás a neve lenne. Ezen kívül le kell ellenőrizni azokat az eseteket, amik komplexé tették a RegEx teszteket például, a korábban említett tört karakter helyes észlelése és formázása.

7.1.2. CalorieCalculator

Egy másik fontos teszt a modulok között a kalória számláló. Mivel ez külső rendszereket is használ, ezért ennek a tesztelése fontos. Itt egy pár ismerten jó és egy pár biztosan sikertelen keresést van végrehajtva a modul segítségével. A sikertelen keresések főként arra alapulnak, hogy jelenleg darabszámhoz nem tudunk kalóriát számolni, mivel nem lehet előre hogy tudni hány gramm lehet például egy darab krumpli vagy tojás.

7.2. Teljes recept importálás

Ez a teszt a teljes importer működését teszteli le egy példa recepten keresztül. Itt jelenleg csak egy receptet tesztelünk, mivel ezeknek a steszeknek az előállítása időigényes. A fejlesztést segítette, hogy a lokális teszteléshez készült egy `localTesting.ts` fájl, ami több már előre kiválasztott recept importálását futtatja végig.

A jövőben hasznos lehet több automatikus tesztet elkészíteni, esetleg a teszt előállítását automatizálni.

8. fejezet

Továbbfejlesztési lehetőségek

8.1. Személyre szabott ajánló

Mivel a regisztrációnál sok egyéb adat van még eltárolva a felhasználóról, ezért ezeket fel lehet használni, hogy személyre szabott receptajánlója legyen mindenkinek a kezdőoldalon. Ez azt jelenti, hogy felmérést lehet indítani arról, hogy az adatbázisban lévő receptek mely felhasználó csoporthoz vannak leggyakrabban elmentve. Ezeket az adatokat a jövőben fel lehet használni a receptek listázásához a ajánlóban.

Amíg a receptet nem mentette ki eléggé sok ember, addig nehéz lehet megállapítani, hogy mely felhasználói csoportoknak tetszhet. Így ilyenkor a feltöltő csoportjához sorolhatjuk, ezzel is segítve a recept korai javaslását.

8.2. Recept ellenőrző admin felület

Jelenleg már létezik minden recept alatt egy bejelentő mező, ahol hibákat tudnak a felhasználók küldeni a receptek állapotáról. Jelenleg ez még csak el van tárolva az adatbázisban, viszont a jövőben, a könnyű kezelés érdekében ezekhez lehetne csinálni egy adminisztrációs felületet. Ezen az oldalon ki lennének listázva a hibás receptek és az admin egy kattintással rá tud nézni a receptre úgy, ahogy a felhasználók is látják.

Itt két lehetősége lenne a hiba megoldására. Ha olyan a recept, hogy nem egyértelmű hogyan lehetne javítani, akkor az törlésre kerül. Ehhez írni kell egy olyan funkciót, ami nemcsak törli az adatbázis receptek kollekciójából, hanem törli, a felhasználók recept gyűjteményeiből is. Amennyiben a javítás lehetséges, akkor az admin számára az összes adat módosítható, így könnyen meg tudja tenni a javítást.

8.3. Desktop-os és Telefonos kinézet továbbfejlesztése

Jelenleg a mobilos és desktopos nézetek nem használják ki az adtott platform által nyújtott lehetőségeket tökéletesen. Így például az asztali nézetben sok üres kihasználatlan hely marad, míg bizonyos komponensek a telefonon túl nagyok. Ezeket az alap beviteli komponensek lecserélésével lehetne javítani, illetve az asztali nézetet esetleg újra gondolni az üres helyeknek új felhasználát kitalálni.

8.4. Mértékegység átváltás

Mivel három féle mértékegységet szokás használni főzésnél, mégpedig az Egyesült Államok által használt rendszert (cup, ounce, inch, stb) vagy az általános SI mértékegységeket (ml, g, dkg, stb.) és konyhai mértékegységeket (tsb, tbsp, stb.), így a recept nézetben hasznos volna, ha az összetevők mértékegységeit automatikusan át tudná váltani a program. Ez azért jutott eszembe, mivel általában az emberek csak az egyiket vagy a másikat ismerik jobban. Tehát be lehetne állítani a profilban, hogy a felhasználó melyiket preferálja, valamint, hogy akarja-e, hogy mindig át legyenek váltva. Ezen kívül lehet még akár minden recept tetején egy gombbal megkérni a weblapot, hogy váltsa át a mértékegységeket.

8.5. Főzési folyamat lejátszó

Ha valaki telefonon néz meg egy receptet, akkor elég zavaró tud lenni mikor egy recept lépés közepén azt írják, hogy “használd fel a fentebb említett alapanyagokat”, amihez föl-le kell mozgatni a képernyőt. Ezzel elvesztve, hogy hol tartottunk a rept lépései között. Ennek a problémának megoldása érdekében lehetne egy “lejátszás” gomb, ami továbbvisz egy olyan lapra, amin a lépések egyesével vannak megjelenítve a hozzájuk tartozó alapanyagokkal. Amikor egy lépés kész, egy gombbal a következő lépésre lehetne menni.

Így könnyen követhető lenne a főzés folyamata. Valamint, mivel az alapanyagok külön mértékegységgel és mennyiséggel vannak eltárolva, ezért pontos értékeket lehet megadni a különböző lépésekben ha ugyanazt az alapanyagot több részletben és több lépésben kell felhasználni.

8.6. Videók és képek jobb felhasználása

Vannak olyan receptek, ahol videó is szerepelt a recept adatai között. Viszont ezt a videó lenne jelenleg nincs eltárolva és a felhasználónál nem jelenik meg. Ehhez a szükség lenne a programban egy videó lejátszóra. A videó eltárolására két lehetőség van. Az egyik esetben a teljes videó a weblap saját felhő alapú tárhelyére lenne kimentve, ez a legmegbízhatóbb de a várhatóan a legdrágább megoldás is. A Második opció az pedig, hogy csak a videóra mutató URL címet tároljuk el hasonlóan a jelenlegi képekhez.

Jelenleg a képeknek is csak az url címe van elmentve de a képeket letöltő Cloud Function már kész van. Ez azért nem fut jelenleg, mert a kliens oldalról a manuálisan megadott receptek képeit bonyolultabb feltölteni.

Nyilatkozat

Alulírott Vas Laura, gazdaságinformatikus szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet Szoftverfejlesztés 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.

Szeged, 2021. november 30.

.....

aláírás

Irodalomjegyzék

[1] Angular dokumentáció,
https://angular.io/docs

[2] Firebase dokumentáció,
https://firebase.google.com/docs/build

[3] Angular Material dokumentáció,
https://material.angular.io/guide/getting-started

[4] Angular Fire dokumentáció (Firebase API),
https://github.com/angular/angularfire

[5] Angular Service Worker dokumentáció,
https://angular.io/guide/service-worker-intro

[6] Shema.org,
https://schema.org/

[7] Mértékegység átváltó 1,
https://www.npmjs.com/package/js-quantities

[8] Mértékegység átváltó 2,
https://www.npmjs.com/package/convert-units

[9] Angular PWA dokumentáció,
https://angular.io/guide/service-worker-getting-started

[10] Jasmine dokumentáció,
https://jasmine.github.io/

9. fejezet

Melléklet

9.1. Projekt elérése

9.1.1. Forrás fájok

A projekt forrás fájljai eléhetőek a GitHub-on a következő linken:
`https://github.com/vluv99/RecipeHoarder`

9.1.2. Élő weboldal

A kész weblap a következő címen érhető el:
`https://recipe-hoarder.web.app/`

Szótár

AngularJS Az Angular keretrendszer előző verziója. A jelenlegi Angulár ennek teljes átírása.. 19

Apache Egy elterjedt webszerver szoftver, ami azNGINX mellet a másik legnagyobb webszerver platform.. 23, 40

API Application Progamming Interface, különböző programok közötti kapcsolatot ír le. Például: milyen URL címeket lehet meghívni egy szerveren adat lekérdezés céljából.. 28

cache A böngésző által ideiglenesen eltárolt adatok, amiket egy beállítható ideig nem fog újból lekérdezni a szervertől.. 28

Framework Keretrendszer. 19

HTTP Hypertext Transfer Protocol, egy applikáció szinten működő rendszer, ami webböngészők és webes szerverek közötti komminikációt tesz lehetővé.. 28

JSON / JavaScript Object Notation A Javasicript álltal használt objektum mentési formátum.. 28

Metaadatok Egy adatot leíró adat, például annak utolsó módosítási ideje.. 20

NGINX Az Apache-val együtt az egyik legelterjedtebb webszerver. Sok esetben fordított proxy-nak is szokás használni.. 23, 40

RegEx Regular Expresion (Reguláris Kifejezés) egy szöveg értelmezésre használható minta felismerő nyelv.. 28, 33

Single Page Alkalmazás Egy olyan webes alkalmazás, ahol egy index.html lap van újra és újra feltöltve dinamikusan adatokkal.. 19

TypeScript Egy JavaScript-re forduló program nyelv. A JavaScript Microsoft álltal tovább fejlesztett verziója.. 19, 23

Unicode Vulgar Fraction Unikódban meghatáozott azon karakterek, amelyek egy törtet írnak le egyetlen karaterrel. pl.: ¼. 28

USDA Unister States Department of Agricuture, Amerikai Egyesült Államok Mezőgazdasági Minisztériuma . 28