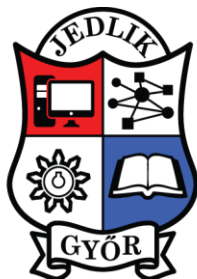




győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

Záródolgozat feladatkiírás

Tanuló(k) neve²: Csepi Szilveszter, Hegyi Áron Ferenc, Menyhárt Attila
Képzés: nappali munkarend
Szak: 5 0613 12 03 Szoftverfejlesztő és tesztelő technikus

A záródolgozat címe:

Gump

Konzulens: Nits László
Beadási határidő: 2023. 04. 28.

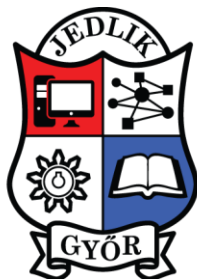
Győr, 2022. 10. 01

² Szakmajegyzékes záródolgozat esetében több szerzője is lehet a dokumentumnak, OKJ-s záródolgozatnál egyetlen személy ad le záródolgozatot.



győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

Konzultációs lap³

	A konzultáció		Konzulens aláírása
	ideje	témája	
1.	2023.02.15.	Témaválasztás és specifikáció	
2.	2023.03.14.	Záródolgozat készültségi fokának értékelése	
3.	2023.04.17.	Dokumentáció véglegesítése	

Tulajdonosi nyilatkozat

Ez a dolgozat a saját munkánk eredménye. Dolgozatunk azon részeit, melyeket más szerzők munkájából vettünk át, egyértelműen megjelöltük.

Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul vesszük, hogy a szakmai vizsgabizottság a szakmai vizsgáról kizár minket és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2023. április 28.

Csepi Szilveszter

Hegyi Áron Ferenc

Menyhárt Attila

³ Szakmajegyzékes, csoportos konzultációs lap

Gump

Jedlik Ányos Gépipari és Informatika Technikum

Gump

Dokumentáció

Készítették:
Csepi Szilveszter
Hegyi Áron Ferenc
Menyhárt Attila

Tartalomjegyzék

1.	A vizsgaremekről	4
1.1.	Mi az a Gump?.....	4
1.2.	A Gump-ről bővebben	4
1.3.	Többnyelvűség.....	4
1.4.	A jövő.....	5
1.5.	Motiváció.....	5
1.6.	Célközönség.....	6
2.	Fejlesztési dokumentáció	7
2.1.	Csapatmunka.....	7
2.1.1.	Munkamegosztás.....	9
2.2.	Technológiák	10
2.3.	Szoftverek, szolgáltatások	10
2.4.	Publikálás.....	11
2.5.	Fejlesztői környezet.....	12
2.6.	Adatbázis	12
2.6.1.	Az adatbázis felépítése	12
2.6.2.	Táblák szerkezete	12
2.7.	UI design.....	16
2.8.	Backend	17
2.8.1.	Model	17
2.8.2.	Repository-k	17
2.8.3.	Controller-ek.....	18
2.8.4.	Autentikáció	20
2.8.5.	Autorizáció.....	20
2.9.	Frontend	21
2.9.1.	Technológiák	21
2.10.	Tesztek.....	23
2.10.1.	Backend	23
2.10.2.	Frontend	24
3.	Felhasználói dokumentáció	26
3.1.	Kompatibilitás.....	26
3.1.1.	Gump	26
3.1.2.	Localer	26

3.2.	Az alkalmazás telepítése.....	26
3.2.1.	Gump	26
3.2.2.	Localer	26
3.3.	A felület bemutatása – Gump	27
3.3.1.	Kezdőképernyő.....	27
3.3.2.	Regisztráció.....	27
3.3.3.	Bejelentkezés.....	27
3.3.4.	Navigációs sáv.....	27
3.3.5.	Főoldal	28
3.3.6.	Recept megtekintése.....	28
3.3.7.	Keresés	29
3.3.8.	Készítés	29
3.3.9.	A készítés lépései.....	29
3.3.10.	Receptek.....	30
3.3.11.	Profil	30
3.4.	A felület bemutatása – Localer.....	31
3.4.1.	Fordító felület.....	31
3.4.2.	Moderációs felület	33
4.	Demo	35
4.1.	Linkek.....	35
4.2.	Mintauser	35
5.	Összegzés.....	36
6.	Irodalomjegyzék	37
7.	Mellékletek.....	38

1. A vizsgaremekről

1.1. Mi az a Gump?

A Gump egy professzionális webes és mobilos alkalmazás kedvenc receptjeink összeállítására, megtekintésére, valamint megosztására. Egyedülálló módon az elkészítés lépéseit modulokból állítja össze az alkalmazás, melyek az egyes hozzávalók elkészítését is magába foglalják. Ezek tetszés szerint alakíthatóak, kombinálhatóak egyéb receptekkel, így lehetővé téve a moduláris felhasználást.

1.2. A Gump-ról bővebben

A Gump egyszerre tükrözi a mögötte álló hatalmas adatbázist, ahol a felhasználók és receptek adatait tároljuk, a felhasználók jelvényeit, amiket az applikáció használatával gyűjthetnek, a receptek moduláris megosztását, azaz tetszés szerinti kombinálását, részekre bontását. Előnye, hogy nagyobb rugalmasságot biztosít a felhasználók számára az ételek elkészítése során. A felhasználók kiválaszthatják a megfelelő modulokat, és azokat összeállítva saját egyedi receptet hozhatnak létre. Emellett a moduláris megközelítés lehetővé teszi a könnyebb alkalmazkodást például a diétás igényekhez.

A negyedik alappillért a partnereink és banner hirdetések képezik, amikből bevételszerzési lehetőségünk adódik.

Az alkalmazásunk segítségével egyszerűen lehet recepteket készíteni. A beépített közösségi háló segítségével a felhasználók tudják egymást követni és a recepteket kedvelni.

1.3. Többnyelvűség

A Gump összesen 6 különböző nyelvet támogat. Ezek a magyar, angol, német, francia, koreai és román nyelvek.

Ehhez készítettünk egy felületet, a Localert, amit az általunk kiválasztott fordítók tudnak igénybe venni a fordítások elkészítéséhez. Ennek működése a [Technológiák](#) fejezetben van részletezve.

1.4. A jövő

A szoftvertervezés során rengeteg olyan ötlet merült fel, amit jelenleg nem sikerül megvalósítani. Ilyen például a Tensorflow segítségével történő mozdulat felismerés. Ezt a lapozásnál gondoltuk használni, hogy főzés/sütés közben elegendő legyen a telefon képernyője felett elhúzni a kezünket, anélkül, hogy a képernyőt megérintenénk.

Továbbá a szakvégzett szakácsokat szerettünk volna megkülönböztetni az alkalmazásban, hogy a felhasználók tudják, az ő receptjeik biztosan kiválóan vannak elkészítve. A szakvégzett szakácsok bizonyítványaikat beküldhették volna az oldalunkra, amik hitelességét a moderátorok ellenőrizték volna. Így egy kis pipával gondoltuk jelezni a többi felhasználó számára, hogy az a szakács hiteles.

Ezen felül a Gump a felhasználói tevékenységek alapján személyre szabott receptajánlásokat jelenített volna meg a kezdőlapon.

A Dall-E nevezetű mesterséges intelligenciát is kívántuk ötvözni alkalmazásunkhoz, ami a receptek tartalmát kielemezve egy viszonylag életszerű képet generált volna le az ételekhez.

Végül egy szintén AI-al kapcsolatos elképzelésünkben, a Github Copilot segítségével szerettünk volna tökéletes és formabontó recepteket létrehozni.

Ám idő hiányában ezekről le kellett mondanunk és az alapvető működést biztosító elemek elkészítésére helyeznünk a hangsúlyt.

1.5. Motiváció

Tisztában vagyunk vele, hogy léteznek receptmegosztó alkalmazások, azonban egyik sem támogatja a többnyelvűséget és a moduláris felhasználást. Mivel mindannyian szeretünk a konyhában tevékenykedni és receptes könyveket olvasgatni szükségesnek láttuk, hogy létrehozzunk egy olyan alkalmazást, ami elsősorban a fentebb említett elképzeléseinkhez igazodik, hogy minél kifinomultabb receptek elkészítését tehessük lehetővé.

1.6. Célközönség

A Gump minden olyan felhasználót szívesen fogad, aki szeret a konyhában sütni/főzni, szeretne új, mások által elkészített receptek alapján ételeket készíteni, vagy érez magában elég tettvágyat ahhoz, hogy saját receptjeit is megoszthassa a nagyvilággal.

2. Fejlesztési dokumentáció

2.1. Csapatmunka

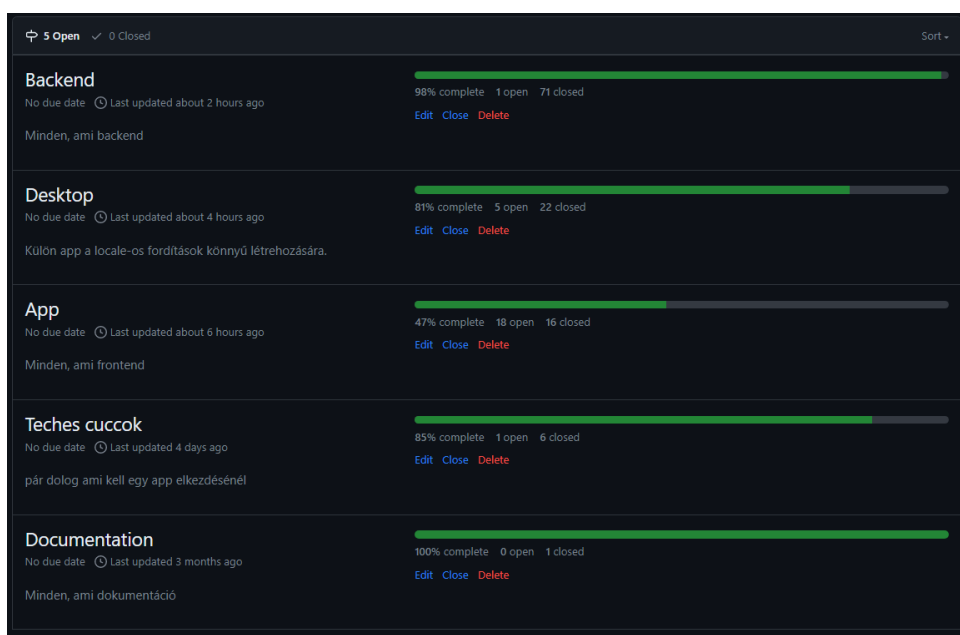
A munka elkezdése előtt alaposan átbeszéltünk, hogy mit várunk el a kész alkalmazástól, összegyűjtöttük az ötleteinket, rangsoroltuk őket fontosság szerint és megalkottuk a specifikációt. A feladatok nem voltak szigorúan elosztva területenként, bárki dolgozhatott mind a backenden, mind a frontenden.

A csapatmunka két legfontosabb eszköze a Github és a Discord volt. Githubra tettük ki a feladatokhoz kapcsolódó issue-kat. Minden issue-t a képen látható tulajdonságokkal ruháztunk fel az átláthatóság kedvéért.

3 féle prioritást különböztettünk meg. A P2 a legfontosabb, a P1 az átlagos fontosságú, a P0 pedig a kevésbé fontos terveket jelentette.



Minden issue tartozott egy milestonehoz, ami a mi esetünkben 5 különböző mérföldkövet takart. Ez érdekes statisztikákkal egészítette ki a fejlesztési folyamatot, mivel láttuk, hogy az egyes milestone-ok százalékosan mennyire vannak készen.



Gump

Az iteration pedig segített abban, hogy az előre meghatározott határidőhöz tartsuk magunkat.

Kihasználtuk a Github Projects adta lehetőségeket, így módunk volt az issue-kat állapot (status) alapján is megkülönböztetni. Az állapotot a Github határozta meg automatikusan az általunk elkészített workflow alapján.

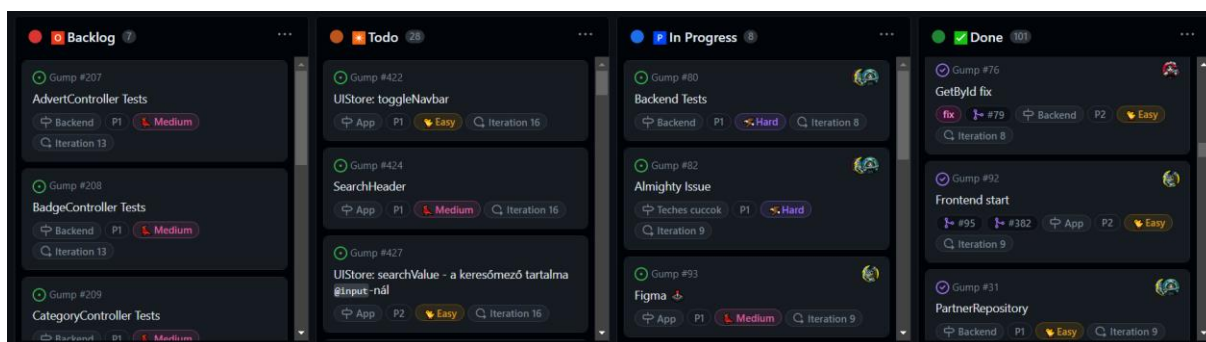
„Backlog-ba” kerültek azok a feladatok, amiket még nem alakítottunk át issue-ra. Ilyenek lehettek azok a tervek, amik még nem voltak alaposan kidolgozva és csak később szerettünk volna rá energiát fektetni.

A „todo” állapotot az issue-vá alakítást követően vették fel a feladatok. Itt az issue-k már rendelkeztek rendes leírással, tasklisttel, feliratokkal, kategóriákkal.

Az „in progress” állapotban az issue-hoz egyikőnk már hozzáadta magát, azt jelentett, hogy foglalkozik vele. Ebben az állapotban szoktunk branch-et is létrehozni, mivel a feature branch workflow elveit követtük.

A „done” state akkor került kiváltásra, amint az issue-hoz tartozó pull request el lett fogadva, be lett mergeelve a main-be és az issue-hoz tartozó branch törölve lett.

Ezen a képen látható a Github Status Board, ahol az issue-kat vezettük.



Összesen 145 issue-t hoztunk létre és több, mint 980 commit volt Githubon.

A Githubon a pull requestek létrehozásánál a módosított fájlakon automatikusan végment egy pár ellenőrzés a kód minőségét illetően. Az egyik ilyen szolgáltatás a Codacy volt, ami a kódot analizálva keresett hibákat, biztonsági veszélyeket, teljesítményt lassító tényezőket

vagy kódolási konvenciók megszegését. Emellett kötelezővé tettük, hogy legalább egy csapattagnak jóvá kell hagynia az írt kódot a mainbe való egyesítés előtt.

Ezen felül igénybe vettük a Copilot for PR's segítségét is, ami minden új pull requesthez automatikusan generált egy leírást a forráskódot elemezve, hogy abban milyen módosítások kerültek végrehajtásra. Ez a funkció, csak a fejlesztési szakasz vége felé került elérhetővé számunkra, de így is hatalmas segítséget nyújtott a review elkészítésében és egymás kódjának elemzésében.

A kitűnő csapatmunka eredményét az bizonyíthatja, hogy merge conflictok igencsak ritkán fordultak elő.

Iskolán kívül a Discord nevezetű alkalmazás segítségével valósítottuk meg a szöveges és hanghívással történő kommunikációt a gördülékeny együttműködés céljából. Kérdések esetén mindig egymás rendelkezésére álltuk, gyakran összeültünk és megvitattuk, hogy hol tartunk az alkalmazás fejlesztésében.

2.1.1. Munkamegosztás

	Csepi Szilveszter	Hegyi Áron Ferenc	Menyhárt Attila
Adatbázis			ER-Modell, AB-Modell
Backend	Modellek, Controllerek, Repository Tesztek, Hosting	Repository-k, Github Controller	Autentikáció, Autorizáció, Modellek, Repository-k, Controller tesztek
Frontend	Navigáció, Recipe View	UI design, Localer, Gump, Hosting	Moderátor felület
Egyéb	Prezentáció, Dokumentációk	Github techs	Apk buildelése

Mint látható – a controllerek és repository-k esetében – szándékosan egymás kódját teszteltük, tehát nem az tesztelte a kódot, aki készítette, hanem a csapat másik tagja. Így tettük hatékonyabbá a hibák kiszűrését.

2.2. Technológiák

Az alkalmazást 3 fő részegységre lehet bontani. Az adatbázisra, a backendre és a frontendre.

Adatbázisként, mi a MongoDB 3-as verzióját választottuk, ami a NoSQL adatbázisszerverek közé tartozik. Gyors, objektum-alapú adatbázis, a dokumentumokat JSON-szerű formátumban tárolja, amit BSON-nek neveznek. Szerintünk könnyű kiigazodni rajta és a MongoDB.Driver .Net NuGet-ről telepíthető könyvtár segítségével erőfeszítés nélkül használható a C# alapú backendünkön.

A backenden az ASP.NET Core 7.0-t használtuk, mivel a C#-ot már régóta ismerjük és szeretjük, könnyen meg tudtuk valósítani az elképzeléseinket benne.

A frontend megalkotására a Nuxt.js 3-ra esett a választásunk, ami a Vue.js 3, a Vite és Nitro alapján készült, első osztályú TypeScript támogatással. A Vue.js-el csak ebben a tanévben kerültünk közelebbi kapcsolatba és rögtön elnyerte tetszésünket átláthatóságával. Azonban a Nuxt.js használata még a Vue.js-nél is könnyebb, így ez a választás is magától értetődő.

2.3. Szoftverek, szolgáltatások

Az alkalmazás fejlesztése során az alábbi szoftvereket, szolgáltatásokat vettük igénybe:

- **Visual Studio Code:** Nyílt-forráskódú, cross-platform kódszerkesztő program, rengeteg kiegészítővel, így bármilyen területen helyt áll, tökéletes volt Nuxt frontend és Asp.Net backend fejlesztésre egyaránt.
- **Github:** Git-kezelő szolgáltatás. Itt valósítottuk meg a project tervezését, a feladatok kiosztását, a csapatmunkát, a kód tárolását, ellenőrzését és menedzselését.
- **Figma:** Az alkalmazás felhasználói felületének (UI) terveit itt készítettük el. Nagy segítség volt, hogy az elkészített komponenseket svg vagy css kód formájában két kattintással sikerült átvinnünk a frontendre.
- **Discord:** Itt valósítottuk meg a projekt részleteit érintő írásos és szóbeli kommunikációt.

- **Capacitor:** Cross-platform natív runtime webapplikációkhoz. Ennek a segítségével készítettük el a weboldalunkból a natív telefonos applikációt.
- **Draw.io:** Ingyenes, nyílt forráskódú diagramkészítő alkalmazás. A programunk ER-modelljét ezen szoftver segítségével alkottuk meg.
- **Github Copilot:** Ez egy AI páros programozó, amit a Visual Studio Code-hoz kiegészítőként telepítettünk. Segítségével a már meglévő kódunkat analizálva és a kommenteket figyelembevéve egészítette ki azt. Roppant nagy segítség volt akkor, amikor elakadtunk és nem tudtuk mivel lehetne az adott funkcionalitást megvalósítani.
- **Dependabot:** Ez egy Github kiegészítő szolgáltatás, ami automatikusan utánanézett a függőségek (dependency-k) újabb verzióinak, ügyelve azok kompatibilitására, illetve figyelmeztetve a régi verziók veszélyeire.
- **Codacy:** Tiszta kód elvek, biztonsági intézkedések, teljesítmény és sok más figyelésére alkalmas szolgáltatás. A pull requestek során a kódot analizálva keresett hibákat, biztonsági veszélyeket, teljesítményt lassító tényezőket vagy kódolási konvenciók hibákat.
- **Docker:** A backendet a szerven Docker konténerben futtatjuk, hogy izolálva legyen a szükségtelen szoftverektől. Hasznos eszközöket nyújt, például automatikus újraindítás biztosít egy szerver leállás folytán.
- **Nginx:** Web és fordított proxy szerver. Az API kéréseket irányítja a backend felé.
- **Let's encrypt:** SSL tanúsítványt állítottunk ki a segítségével, hogy HTTPS kapcsolaton keresztül folyhasson a kommunikáció az alkalmazás és a szerver között.
- **Name.com:** Domain név regisztrálására vettük igénybe.

2.4. Publikálás

- **DigitalOcean:** A backendet a DigitalOcean szerverein hostoljuk, mivel kiváló sebességű Ubuntu Linux rendszert biztosít. A Github Student Pack segítségével 100\$ ingyen kreditet kaptunk, amivel 1 évig biztosan tudjuk futtatni a szervert.

- **Netlify:** Az adminisztrációs és fordító oldalt pedig a Netlify-ra tettük fel, ami szintén ingyenes és könnyen kezelhető. Hasznos tulajdonsága, hogy automatikusan frissíti a weboldalt a Github repository alapján.

2.5. Fejlesztői környezet

Elsősorban mindhárman laptopon dolgoztuk Windows 10 és Windows 11 operációs rendszeren a fentebb említett szoftverek, szolgáltatások igénybevételével. Mindannyiunk laptopja teljesen megbízható volt, erős hardverrel rendelkezett, így a program buildelése nem vett sok időt igénybe. Azonban ritkán, de előfordult, hogy az iskolai gépeken végeztünk el 1-2 munkát. Ott viszont előjöttek problémák hiszen nem minden szoftver volt telepítve az iskolai gépekre, valamint az ASP.NET 7.0-ás verziója sem állt rendelkezésre.

Az alkalmazás mobilos felületét természetesen Android-on is teszteltük.

2.6. Adatbázis

2.6.1. Az adatbázis felépítése

Ahogy azt a [2.2-es fejezetben](#) olvashattuk az adatbázist MongoDB-ben valósítottuk meg.

Az adatbázis tervezését az ER-modell elkészítésével kezdtük meg, amely az „1. ábra” mellékletben tekinthető meg.

Az adatbázis AB diagramja a „2. ábra” melléklet alatt található meg.

2.6.2. Táblák szerkezete

a) users:

- `_id`: Elsődleges kulcs; long
- `token`: A jelszó titkosításához használt Bcrypt salt; string
- `username`: A felhasználónév, amivel a user be tud lépni az alkalmazásba. Egyedinek kell lennie az adatbázisban; string
- `password`: A felhasználó jelszava titkosítva Bcrypt Salt és Peper segítségével 10-es round-al; string

- email: A felhasználó email címe. Egyedinek kell lennie az adatbázisban; string
- profilePicture: A profilkép azonosítója. Ha nem választ ki profilképet a felhasználó, alapértelmezett értéke 1-es; long
- language: Annak a nyelvnek a kódja, amin a felhasználói felület megjelenik; string
- recipes: A felhasználó által létrehozott és lementett receptek azonosítójának listája; array
- likes: A felhasználó által kedvelt receptek azonosítója; array
- following: A felhasználó által követett felhasználók; array
- followers: Azon felhasználók azonosítója, akik követik az adott felhasználót; array
- badges: A felhasználó által birtokolt jelvények azonosítói; array
- isModerator: Annak ellenőrzésére szolgáló mező, hogy a felhasználónak joga van-e moderátori tevékenységet végeznie; boolean

b) recipes:

- _id: Elsődleges kulcs; long
- title: A recept címe; string
- author: A recept készítőjének azonosítója; long
- image: A recepthez tartozó kép azonosítója; long
- language: A recept nyelvének kódja; string
- serves: A recept adagszáma; short
- categories: A recept kategóriáinak azonosítója; array
- tags: A recept tagjeinek azonosítója; array
- ingredients: A recepthez tartozó hozzávalók listája; object
- steps: A recepthez tartozó lépések azonosítója; array
- viewCount: A recept megtekintésének száma; int
- saveCount: A recept mentéseinek száma; int
- likes: Azon felhasználók azonosítója, akik kedvelték a receptet; array
- referenceCount: Annak a száma, hogy hány recept hivatkozik az adott receptre; int

- **isArchived:** A recept archivált állapotát jelző mező. Azon receptek kerülnek ilyen státuszba, amiken van hivatkozás, így ezen receptek nem törölhetőek, csak archiválhatóak; boolean
- **isOriginal:** Megmutatja, hogy a recept eredeti-e, tehát nem egy módosításból származó receptről van szó; boolean
- **originalRecipe:** Az eredeti recept azonosítója; long
- **isPrivate:** A recept láthatóságát jelzi. A privát recepteket csak a készítőjük és a **visibleTo** mezőben felsorolt felhasználók tekinthetnek meg; boolean
- **visibleTo:** Abban az esetben, ha a recept **isPrivate** értéke true ezen mező alatt jelennek meg azon felhasználók, akiknek joga van a receptet megtekinteni; array
- **forks:** Azon receptek azonosítója, amik hivatkoznak az adott receptre; array

c) **partners:**

- **_id:** Elsődleges kulcs; long
- **name:** A partner neve
- **contractUrl:** A partner elérhetőségének url címe; string
- **apiUrl:** A partner termékeinek lekéréséhez használható url cím; string
- **ads:** A partnerhez tartozó hirdetések azonosítója; array

d) **badges:**

- **_id:** Elsődleges kulcs; long
- **name:** A jelvény megnevezése; string
- **description:** A jelvényhez tartozó leírás; string
- **image:** A jelvényhez tartozó kép azonosítója; long

e) **categories:**

- **_id:** Elsődleges kulcs; long
- **name:** A kategória megnevezése; string

f) **ingredient:**

- **_id:** Elsődleges kulcs; long

Gump

- name: A hozzávaló neve; string
- value: A hozzávaló mennyisége; short
- volume: A hozzávaló mennyiségéhez tartozó mértékegység; string
- linkedRecipe: A hozzávalót tartalmazó recept azonosítója; long

Gump

g) adverts:

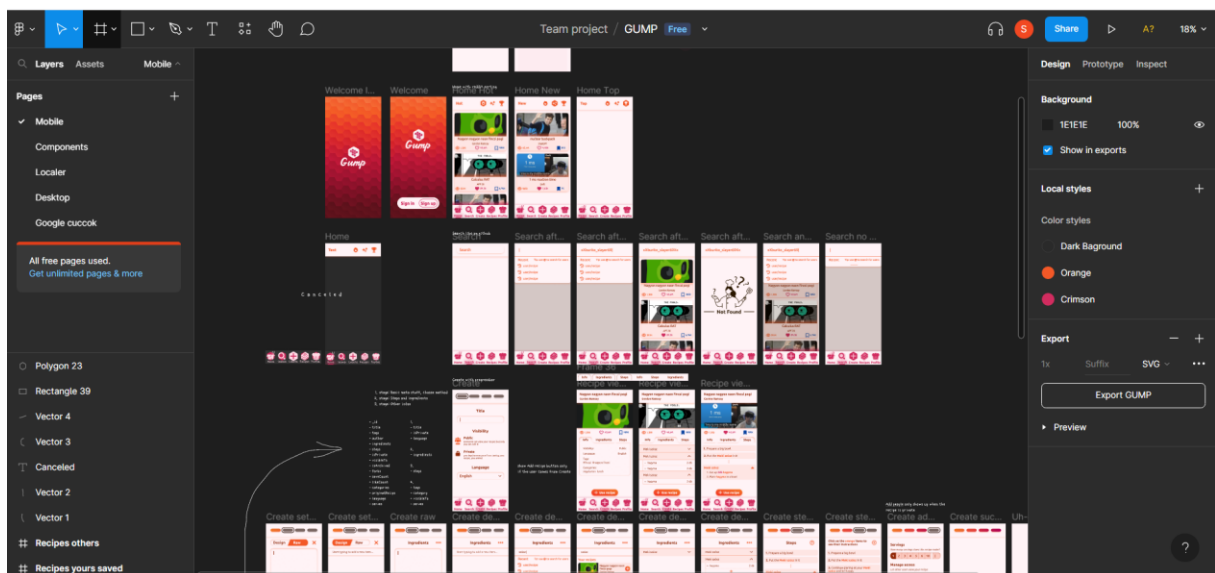
- `_id`: Elsődleges kulcs; long
- `partner`: A hirdetéshez tartozó partner azonosítója; long
- `title`: A hirdetés címe; string
- `image`: A hirdetéshez tartozó kép azonosítója; long

h) images:

- `_id`: Elsődleges kulcs; long
- `image`: A kép tartalma base64-es formátumban; string
- `owner`: A kép tulajdonosa. Nem kötelező mező; long

2.7. UI design

Az adatbázis modell megalkotása után a felhasználói felület, a UI tervezése következett. Ezt a Figma nevezetű alkalmazás segítségével valósítottuk meg. Nagy segítség volt, hogy az elkészített komponenseket svg vagy css kód formájában két kattintással sikerült átvinnünk a frontendre.



2.8. Backend

2.8.1. Modell

A backend megvalósítását a modellek elkészítésével folytattuk a korábban bemutatott adatbázis modell alapján. Minden collection-hoz létrehoztunk egy osztályt a megfelelő property-kkel.

A képen a *BadgeModel* felépítése látható. A *CollectionName* attribútum határozza meg, hogy az osztály melyik *MongoDB* kollekcióhoz tartozik. Láthatjuk, hogy a *BadgeModel* az *IEntity* interfészt örökli, tehát *Id* tulajdonságnak mindenképpen kell szerepelnie az osztály definiálásában.

A *BsonId* adja meg az elsődleges kulcsot, a *BsonRepresentation* pedig a property adatbázisban megjelenő típusát adja meg. A *BsonElement*-tel a mezőneveket definiáltuk.

```
[CollectionName("badges")]
37 references
public class BadgeModel : IEntity
{
    [BsonId]
    [BsonRepresentation(BsonType.Int64)]
    183 references
    public ulong Id { get; set; }

    [BsonElement("name")]
    28 references
    public string Name { get; set; }

    [BsonElement("description")]
    11 references
    public string Description { get; set; }

    [BsonElement("image")]
    [JsonProperty("image")]
    13 references
    public ulong ImageId { get; set; }
}
```

Előfordult, – például az *ImageId* esetében – hogy külön *JsonProperty* attribútumot is megadtunk, hogy a használt elnevezési konvencióknak mindenhol eleget tegyünk.

2.8.2. Repository-k

Minden kollekciónak készítettünk egy-egy repository-t is, amik alapvetően a CRUD műveleteket látják el. Itt történik az adatok ellenőrzése is, hogy ne kerüljön semmilyen mezőhöz olyan adat, ami nem várt működéshez vezetne. Ilyen esetekben hibaüzeneteket dobunk, többek között egyéniket, hogy egyértelmű legyen az ok. Csináltunk egy generikus szülő osztályt is, hogy azok a metódusok, amik mindenhol jól jönnek, ne kelljen mindenhol újra megírni.

Ilyen például a GetById metódus, amit mindegyik modell használ.

```
public T GetById(ulong id)
{
    T entity = Collection.AsQueryable().FirstOrDefault(x => x.Id == id);

    if (entity == null)
    {
        throw new NotFoundException($"{typeof(T).Name.Replace("Model", string.Empty)} with id {id} does not exist");
    }

    ValidateFields(entity, "Id");

    return entity;
}
```

Láthatjuk, hogy a visszatérési érték (T) egy tetszőleges osztály lehet, tehát a metódus generikus.

Az adatbázisból Id alapján lekérjük a dokumentumot. A ValidateFields ellenőrzi az Id helyességét, majd a végén visszaadjuk a lekérdezett objektumot. Abban az esetben, ha az adott id-vel nem létezik elem, akkor *NotFoundException* üzenettel befejezzük a metódus futását.

2.8.3. Controller-ek

A controllerek biztosítják a kommunikációt a backend és a frontend között. A backenden többek között az alábbi endpointok érhetőek el az <https://api.gump.live> címen:

a) Advert:

Get All Adverts – Összes hirdetés lekérése

Elérési út: /advert

Típus: GET

Jogosultság: csak moderátor

Hibakódok: 401 Unauthorized

Get Advert By Id – Hirdetés lekérése id alapján

Elérési út: /advert/:id

Típus: GET

Route Paraméter: id

Jogosultság: bárki

Hibakódok: 400 Bad Request

Get Random Advert – Véletlenszerűen hirdetés kiválasztása

Elérési út: /advert/random

Típus: GET

Jogosultság: bárki

Hibakódok: 404 Not Found

Create Advert – Hirdetés létrehozása

Elérési út: /advert/create

Típus: POST

Request body: CreateAdvertDto

Jogosultság: csak moderátor

Hibakódok: 400 Bad Request

Update Advert – Meglévő hirdetés módosítása

Elérési út: /advert/update

Típus: PATH

Request body: AdvertModel

Jogosultság: csak moderátor

Hibakódok: 400 Bad Request, 401 Unauthorized, 404 Not Found

Delete Advert – Hirdetés törlése id alapján

Elérési út: /advert/delete/:id

Típus: DELETE

Route Paraméter: id

Jogosultság: csak moderátor

Hibakódok: 401 Unauthorized, 404 Not Found

b) Auth:

Login - Bejelentkezés

Elérési út: /auth/login

Típus: POST

Request body: LoginDto

Jogosultság: bárki

Hibakódok: 401 Unauthorized

c) Github:

Get Access Token – Github Access Token lekérése

Elérési út: /github/access_token?code=test

Típus: GET

Paraméter: code

Jogosultság: bárki

A többi HTTP kérés az alábbi linken érhető el. Itt teszteltük a route-ok helyes működését:

<https://www.postman.com/security-saganist-93283134/workspace/gump/documentation/26477376-904f2ea0-9352-4122-a3c6-e39b867bd003>

2.8.4. Autentikáció

A backend az autentikációhoz JWT szabványt használ a biztonság fokozása érdekében. A megszerzett session token kihasználása ellen egy biztonságos secret token-nel és egy 60 perces lejáratú idővel védekezünk.

2.8.5. Autorizáció

A Gump alapvetően 3 féle felhasználót különböztet meg. A regisztrálatlan, a regisztrált és a moderátor felhasználót.

Felhasználói jogok:

- Regisztrálatlan felhasználók: Lehetőségük van valamennyi nyilvánosan elérhető receptet megtekinteni és lementeni.
- Regisztrált felhasználók: Igénybe vehetik a közösségi felületet, tudnak felhasználókat követni és recepteket kedvelni. Természetesen létrehozásra is lehetőségük van.
- Moderátorok: Ezen felhasználók feladata, hogy folyamatosan ellenőrizzék az oldalra felkerül tartalmakat és oda nem illő tevékenység esetén megtegyék a megfelelő intézkedéseket. Ez lehet a recept szerkesztése, levétele vagy a receptet feltöltő felhasználó végleges kitiltása. Valamint kezelik a partereket és hirdetéseiket. Minderre a Localer webes felületét használják.

2.9. Frontend

A frontenden két különálló felületet valósítottunk meg. Az egyik maga a Gump, ami egy mobilos applikáció formájában érhető el, a másik pedig a Localer, amit a fordítóink és a moderátorok tudnak használni.

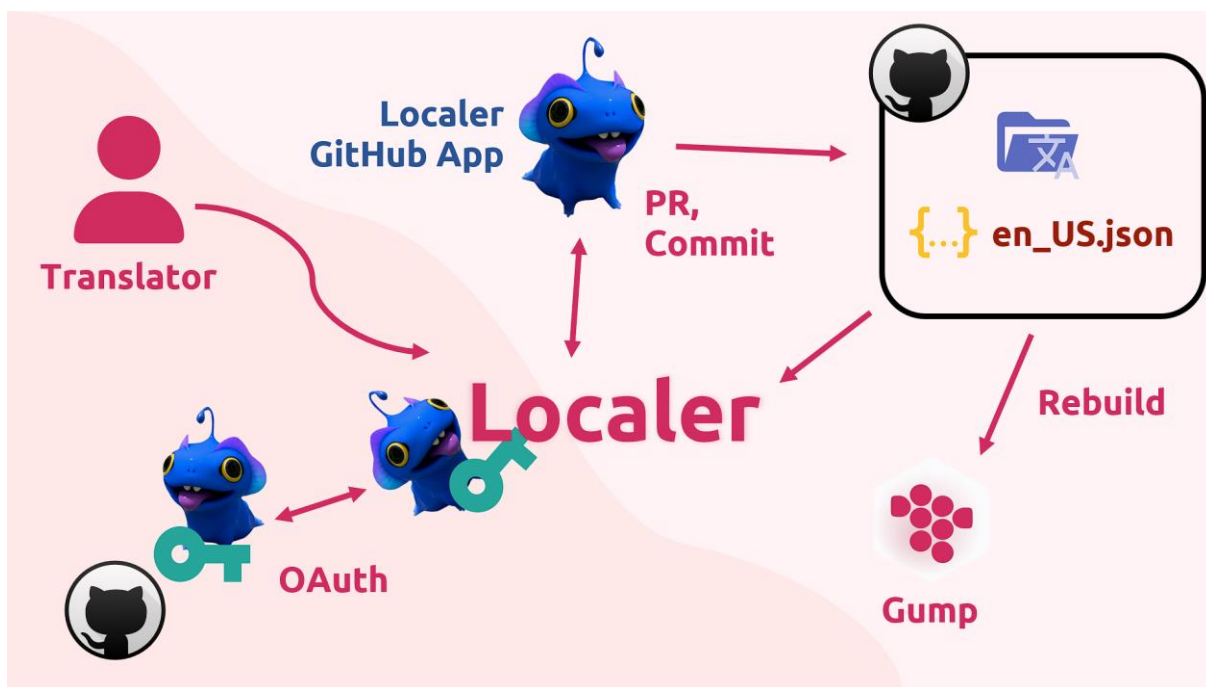
2.9.1. Technológiák

Localer:

Vue.js és WindiCSS frameworkben valósítottuk meg.

A Localer elsősorban a Gump többnyelvűségének elkészítésére használható. A Gump-on található feliratok kulcsként jelennek meg, majd minden egyes kulcshoz lehet definiálni egy leírást és az egyes nyelveken való megjelenését.

Működésének sematikus ábrája itt látható:



A használatához OAuth-szal össze kell kapcsolni a felületet a fordító GitHub fiókjával. A legfrissebb fordításokat azonnal lekérdezi a Repository-ból. Amikor a Mentés gombra kattintunk, akkor az app készít egy branchet, ahova egy commitban pusholja a módosításokat, valamint nyit egy új pull requestet. A mi jóváhagyásunkkal kerül csak be a forráskódba, aminek a buildelésével kerül be a végleges app-ba.

A Localer-hez csak olyan Github fiókkal lehet hozzáférni, akik rendelkeznek írásjoggal a Gump repository-hoz.

Gump:

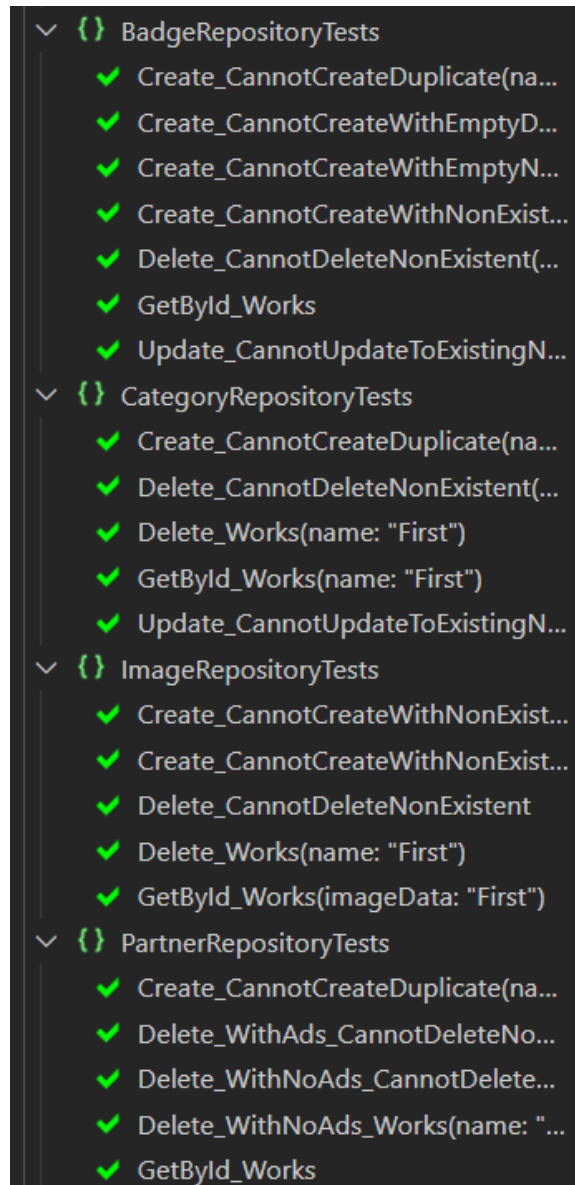
Nuxt.js és UnoCSS keretrendszerben készült, amit a Capacitor nyílt forráskódú futtatókörnyezet segítségével alakítottuk át natív mobil alkalmazássá. Az Ionic framework pedig a mobilos UI komponensek kialakításában nyújtott segítséget.

2.10. Tesztek

2.10.1. Backend

A hibák kiszűrése érdekében a backenden végeztünk repository és controller teszteket is. Több, mint 60 unit tesztet sikerült elkészítenünk. Teszteltünk hibás bemeneti paraméterekre, autorizáció nélküli cselekedetre. Leellenőriztük az adatbázisba való mentést, frissítést, törlést és lekérdezést, illetve, hogy a kontroller megfelelő hibakóddal reagál hibás adat esetén.

Teszteknél létrehoztunk minta adatokat mindegyik Model estén a RepositoryTestsBase osztályban.



A BadgeModel-hez elkészített minta osztálypéldány:

```
public BadgeModel Badge { get; set; } = new()
{
    Description = "Test",
    ImageId = 1,
    Name = "Test",
};
```

Ezeket a minta adatokat az egyes tesztek a Get generikus metódus segítségével kéri le a következőképpen:

```
public static T Get<T>() where T : IEntity
{
    var repositoryTestsBase = new RepositoryTestsBase();
    var property = repositoryTestsBase.GetType()
        .GetProperties(BindingFlags.Public | BindingFlags.Instance)
        .FirstOrDefault(x => x.PropertyType == typeof(T));

    var result = (T)Activator.CreateInstance(typeof(T));

    var value = property.GetValue(repositoryTestsBase);
    if (value is not null)
    {
        result = (T)value;
    }

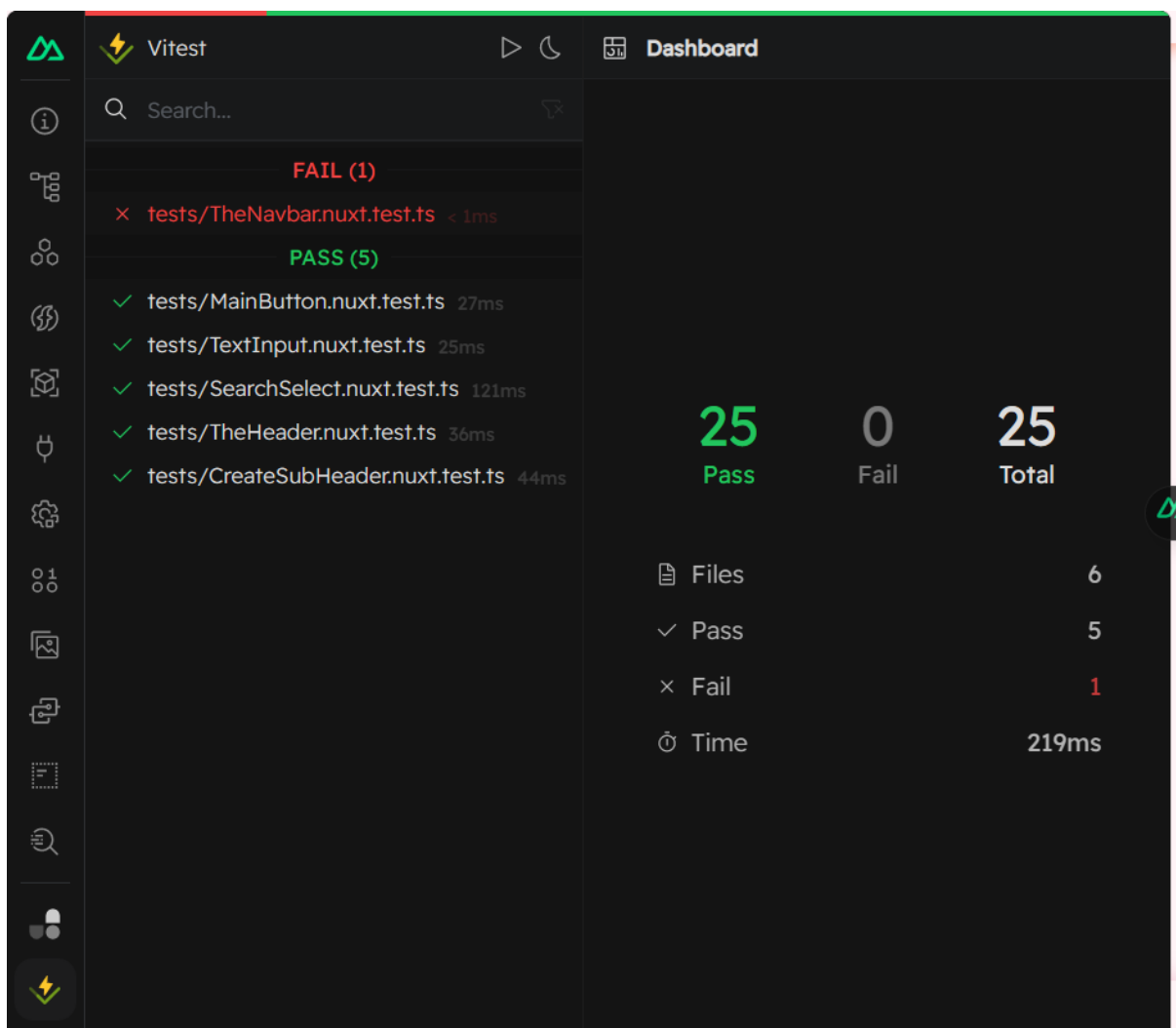
    return result;
}
```

A metódus először létrehoz egy RepositoryTestBase objektumot, amiben a minta osztálypéldány van definiálva. Ezután meghatározza a T típusú objektum property-jeit, majd létrehoz egy új T példányt az alapértelmezett értékekkel és visszaadja azt.

2.10.2. Frontend

A frontenden a UI és a unit teszteket a Vitest segítségével valósítottuk meg. Teszteltük a navigációs menük helyes működését, az alkalmazás által megjelenített adatok helyességét,

formátumát és elrendezését. Megbizonyosodtunk arról, hogy gombnyomás hatására megjelennek-e a kívánt elemek a DOM-ban és tartalmazzák a megfelelő szöveget.



Vitest Dashboard

Search...

FAIL (1)

- × tests/TheNavbar.nuxt.test.ts < 1ms

PASS (5)

- ✓ tests/MainButton.nuxt.test.ts 27ms
- ✓ tests/TextInput.nuxt.test.ts 25ms
- ✓ tests/SearchSelect.nuxt.test.ts 121ms
- ✓ tests/TheHeader.nuxt.test.ts 36ms
- ✓ tests/CreateSubHeader.nuxt.test.ts 44ms

25 **0** **25**
Pass **Fail** **Total**

Files	6
✓ Pass	5
× Fail	1
⌚ Time	219ms

3. Felhasználói dokumentáció

A Gump-ot felhasználói receptek létrehozására, megosztására, böngészésére, kedvelésére, mentésére, átdolgozására tudják használni. Ezekon túl jelvényeket gyűjthetnek az applikáció használatával különböző feltételek teljesítése esetén.

3.1. Kompatibilitás

3.1.1. Gump

- Operációs rendszer: Android 5.1+ (API 22+)
- Memória: 1GB+
- CPU: 1GHz+

3.1.2. Localer

- Webes alkalmazás, asztali számítógépre szánt használatra.
- Támogatott böngészők: Google Chrome 112.0, Mozilla Firefox 112.0, Microsoft Edge 112.0
 - Más böngészőkben való helyes működést nem tudunk garantálni, mivel a fentebbi programokon lett csak letesztelve!

3.2. Az alkalmazás telepítése

3.2.1. Gump

- A Gump apk az alábbi linken érhető el: <https://github.com/14A-A-Lyedlik-Devs/Gump/releases>
- Érdemes a legfrissebb verziót letölteni.

3.2.2. Localer

- A Localer az alábbi webcímen érhető el: <https://gump.live/>

3.3. A felület bemutatása – Gump

3.3.1. Kezdőképernyő

A telepítés után a felhasználót az alábbi ablak fogadja, ahol lehetősége van regisztrálni, bejelentkezni, vagy bejelentkezés nélkül továbbmenni.



3.3.2. Regisztráció

A regisztráció során 3 adat megadása szükséges, felhasználónév, email cím és jelszó. A felhasználónévnek és email címnek egyedinek kell lennie, sőt utóbbinak érvényes formátumúnak. Hiba esetén a felhasználó értesítve van.

3.3.3. Bejelentkezés

Bejelentkezni kizárólag érvényes felhasználónév és jelszó párossal lehetséges.

3.3.4. Navigációs sáv

Az applikációban 5 különböző oldal megtekintésére van lehetőség. Ezek az alábbiak:

- **Főoldal:** A receptek megtekintése különböző rendezési szempontok szerint
- **Keresés:** Receptek, felhasználók keresése
- **Készítés:** Recept elkészítése
- **Receptek:** Saját, kedvelt és elmentett receptjeink megtekintése
- **Profil:** Saját felhasználói profil megtekintése, szerkesztése



3.3.5. Főoldal

A főoldalon láthatóak a receptek az alábbi rendezési szempontok szerint:

- Felkapott: Olyan receptek kerülnek ide, melyeket rövid idő alatt sok felhasználó mentett le.
- Új: Időrendi sorrendben megjeleníti a recepteket.
- Legjobb: Azon receptek, melyeket összességében a legtöbben kedveltek.

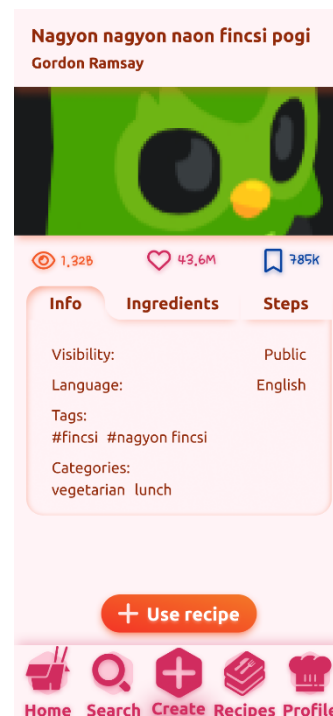
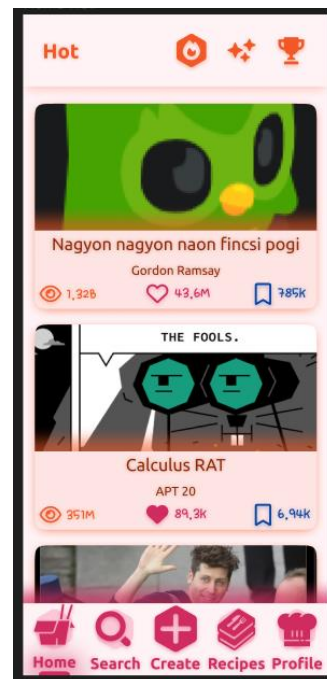
3.3.6. Recept megtekintése

Egy recept megtekintésekor az alábbi adatokat láthatjuk:

- Megtekintések száma
- Kedvelések száma
- Mentések száma
- Alap információk: Láthatóság, nyelv, tag-ek, kategóriák
- Hozzávalók
- Lépések

Egy recept megtekintésekor az alábbi műveletek végezhetőek:

- Recept kedvelése
- Recept mentése
- Recept felhasználása: Ez csak akkor érhető el, ha a receptet a készítés során tekintjük meg.



3.3.7. Keresés

Lehetőség van receptekre keresni tag-ek alapján, felhasználókra név alapján. Felhasználóra való keresés esetén @ jellel kell kezdeni a keresést.

Amennyiben a keresés nem vezet eredményre a felhasználó értesítésre kerül.

3.3.8. Készítés

A receptek elkészítésére 2 megadási mód létezik, a design és a raw. Előbbi esetében az applikáció innovatív felületén kell megírni a receptet, utóbbinál pedig hagyományos, szöveges módon.

Természetesen a leghatékonyabb elkészítés érdekében a két módszert együttesen is lehet alkalmazni.

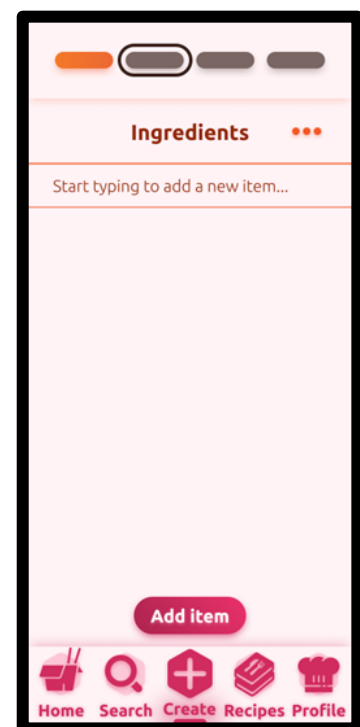
3.3.9. A készítés lépései

1. Alap adatok:

A recept címének megadása, láthatóság beállítása, recept nyelvének meghatározása.

2. Hozzávalók:

A beviteli mezőben megadott szöveg alapján a Gump felajánl olyan recepteket, amik a megadott hozzávalóhoz kapcsolódnak. Itt választhatunk a saját és más publikus receptek között. A receptre kattintáskor megjelenik annak bővebb leírása. A + gombra kattintáskor a recept hozzáadásra kerül a hozzávalók listához.



„Elem hozzáadása” estén további hozzávaló kerülhet megadásra.

3. Lépések:

Itt kerülnek felsorolásra az étel elkészítéséhez szükséges lépések.

Az „Elem hozzáadása” gombra kattintva további lépések adhatóak hozzá a recepthez.

4. Végső simítások:

- Adag: Hány főre készült a fogás.
- Hozzáférés kezelése: Privát recept esetén jelenik meg. Olyan felhasználók sorolhatóak itt fel, akik számára a privát láthatóság ellenére megtekinthető a recept.
- Kategória: A recept kategóriája.
- Tag-ek: A recepthez tartozó tag-ek. A könnyebb megtalálhatóság érdekében itt minden olyan kifejezést fel kell sorolni, ami a receptünket jellemzi, mivel a kereső ez alapján fogja a recepteket megtalálni.

Bármilyen hiba esetén a „váratlan hiba történt” felirat jelenik meg. Ilyenkor újra kell kezdeni a recept létrehozását.

3.3.10. Receptek

Az alábbi receptek jelennek meg ebben a menüpontban.

- Saját receptek
- Kedvelt receptek
- Elmentett receptek

3.3.11. Profil

A felhasználó az alábbi adatokat tekintheti meg:


- Követők száma
- Követettek száma
- Jelvények

Az alábbi adatok módosítására van lehetőség:

- Profilkép
- Felhasználónév
- Email cím
- Jelszó
- Applikáció nyelve

3.4. A felület bemutatása – Localer

Az alábbi navigációs menüpontok érhetőek el:

Home Translate Moderation Gump repo 

- Kezdőlap: Általános információk
- Fordító felület: Github-al történő bejelentkezés után érhető el. Itt lehet elkészíteni az applikációhoz a fordításokat.
- Moderátor felület: Receptek, felhasználók, partnerek és hirdetések moderálása. Csak moderátori bejelentkezéssel érhető el.
- Github repo: Link a Github repositoryhoz.

3.4.1. Fordító felület


A Gump-ban megjelenő szövegek különböző nyelvekre való fordítása végezhető el ezen felületen.

1. Kezdőlap:

Mielőtt bárki belevág a fordítások elkészítésébe javasoljuk a fordító felület kezdőlapját alaposan elolvasni, értelmezni. Itt van lehetőségünk kiválasztani azon nyelveket, amikhez fordításokat szeretnénk elkészíteni.

Choose your language

Select your languages below. You can also add a new language.

de_DE x en_US x hu_HU x 

Jelenleg 6 nyelv érhető el az appban: Angol, Magyar, Német, Francia, Koreai és Román.

Amennyiben a listában nem szerepel a nyelv könnyedén hozzáadható a nyelv kódjának használatával az „Új nyelv hozzáadása” gomb megnyomásakor. Az országgódok listája ezen oldalon érhető el (Two Letter): <https://www.finner.org/Utilities/CountryLanguageList.shtml>

Bal oldalon láthatók a kulcsok, ezek a frontenden egy-egy változóra mutatnak, amik értékei az applikációban a megjelenítési nyelvnek megfelelően jelennek meg.

2. Fordítás készítése:

Egy kulcs kiválasztásakor az alábbi felület jelenik meg:

Home Title Top

Notes

The home page's title changes between "Hot", "New" and "Top" according to the sorting method the user selected. "Top" is the place of the absolute most popular recipes.

de_DE

en_US

Top

fr_FR

Meilleur,e

hu_HU

Legjobb

ko_KR

ro_RO

Itt megadható a kulcshoz egy lírás, illetve az egyes nyelvekhez elkészített fordítás. Csak azon nyelveket lehet módosítani, amiket a kezdőlapon kiválasztottunk.

Egy másik kulcsra való hivatkozás is megadható egy-egy fordítás elkészítésében. A kulcs megadása a „:@” jel után adható meg.

A „{” jelek között a frontenden elérhető egyéb változók értékére lehet hivatkozni.

Új kulcs hozzáadására is van mód, az „Add new key” gombra kattintva.

A kulcsok melletti számok a fordítás elkészültségét mutatják.

3.4.2. Moderációs felület

1. Receptek moderálása:

- Az összes recept kilistázásra kerül.
- Műveletek:
 - Recept adatainak módosítása
 - Recept törlése

2. Felhasználók moderálása:

- Az összes felhasználó kilistázásra kerül.
- Műveletek:
 - Felhasználók törlése
 - Felhasználó profilképének törlése

3. Partnerek kezelése:

- Az összes partner kilistázásra kerül.
- Műveletek:
 - Új partner létrehozása
 - Partner adatainak módosítása
 - Partner törlése

4. Hirdetések kezelése:

- Az összes hirdetés kilistázásra kerül.
- Lehetőség van partnerre szűrni a megjelenő listát
- Műveletek:
 - Új Hirdetés létrehozása
 - Hirdetés adatainak módosítása
 - Hirdetés törlése

Hibás adatok esetén visszajelzést ad a frontend a felhasználó számára és nem küldi el a hibás bevittet a backend felé.

Gump

Amennyiben a Gump-al szerződést köt egy partner cég, hogy hirdetéseket jeleníthessen meg az applikációban az adminisztrációs díj befizetése ellenében egy moderátor kirendelésre kerül a cég részére. Bármilyen hirdetést a céghez tartozó moderátor tud feladni, módosítani és törölni. A cégnek az igényeit a moderátor felé kell jeleznie.

4. Demo

4.1. Linkek

- Backend – API: <https://www.api.gump.live>
- Backend – Postman: <https://www.postman.com/security-saganist-93283134/workspace/gump/documentation/26477376-904f2ea0-9352-4122-a3c6-e39b867bd003>
- Frontend – Localer: <https://gump.live>
- Frontend – Gump: <https://github.com/14A-A-Lyedlik-Devs/Gump/releases>

4.2. Mintauser

Tesztelési célokra az alábbi felhasználói bejelentkezések érhetőek el:

- TestUser:
 - Felhasználónév: TestUser
 - Jelszó: secret
- Moderator:
 - Felhasználónév: Moderator
 - Jelszó: moderator001

5. Összegzés

A fejlesztés során rengeteg kihívással kerültünk szembe. Voltak olyanok, amiket pár perc, néhányat csak hetek múltán sikerült orvosolni. Büszkék vagyunk arra, hogy csapatunk bármilyen felmerülő problémára megoldást tudott találni, rengetegszer segítettünk egymásnak a helyes megoldás megtalálásában. Egyszóval nagyszerű csapat voltunk, szerettünk együtt dolgozni, sokat nevettünk és sírtunk. Örültünk annak, hogy mindenki belekóstolhatott mind a backend, mind a frontend fejlesztésébe, mert nem ezen területek alapján osztottuk el a munkát. Reméljük az elkészült applikáció beváltja a hozzá fűzött reményeket és azzal a lelkesedéssel tudjuk bemutatni az alkalmazást, mint ahogyan a fejlesztésbe belekezdünk. Az biztos, amit idén tanultunk az életben nagy hasznukra lesz.

6. Irodalomjegyzék

Anonymus, MongoDB (2023.04.28.) <https://hu.wikipedia.org/wiki/MongoDB>

Anonymus, TypeScript (2023.04.28.)

<https://www.typescriptlang.org/docs/handbook/2/typeof-types.html>

Anonymus, VueJs (2023.04.28.) <https://vuejs.org/guide/essentials/reactivity-fundamentals.html#declaring-reactive-state>

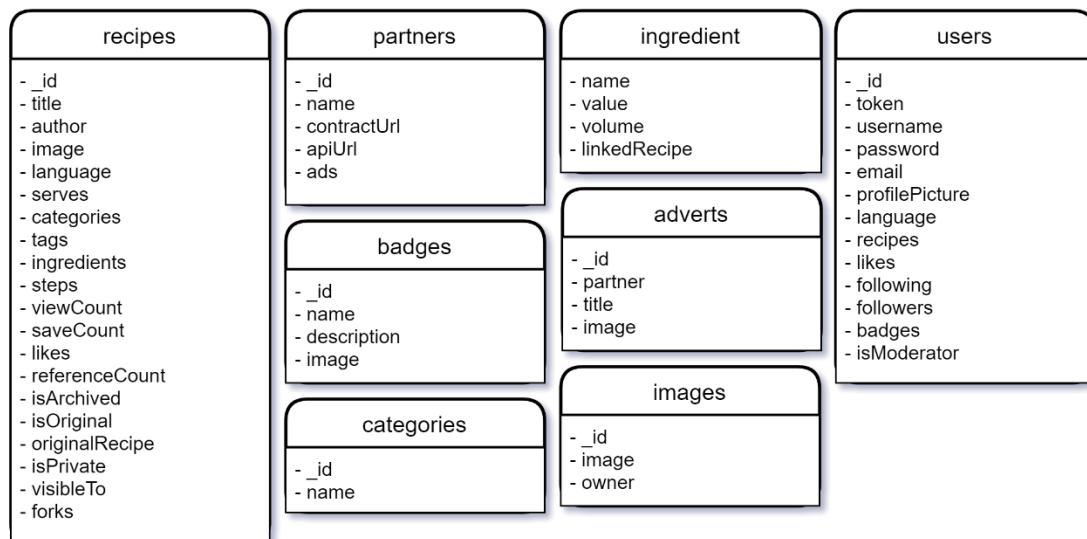
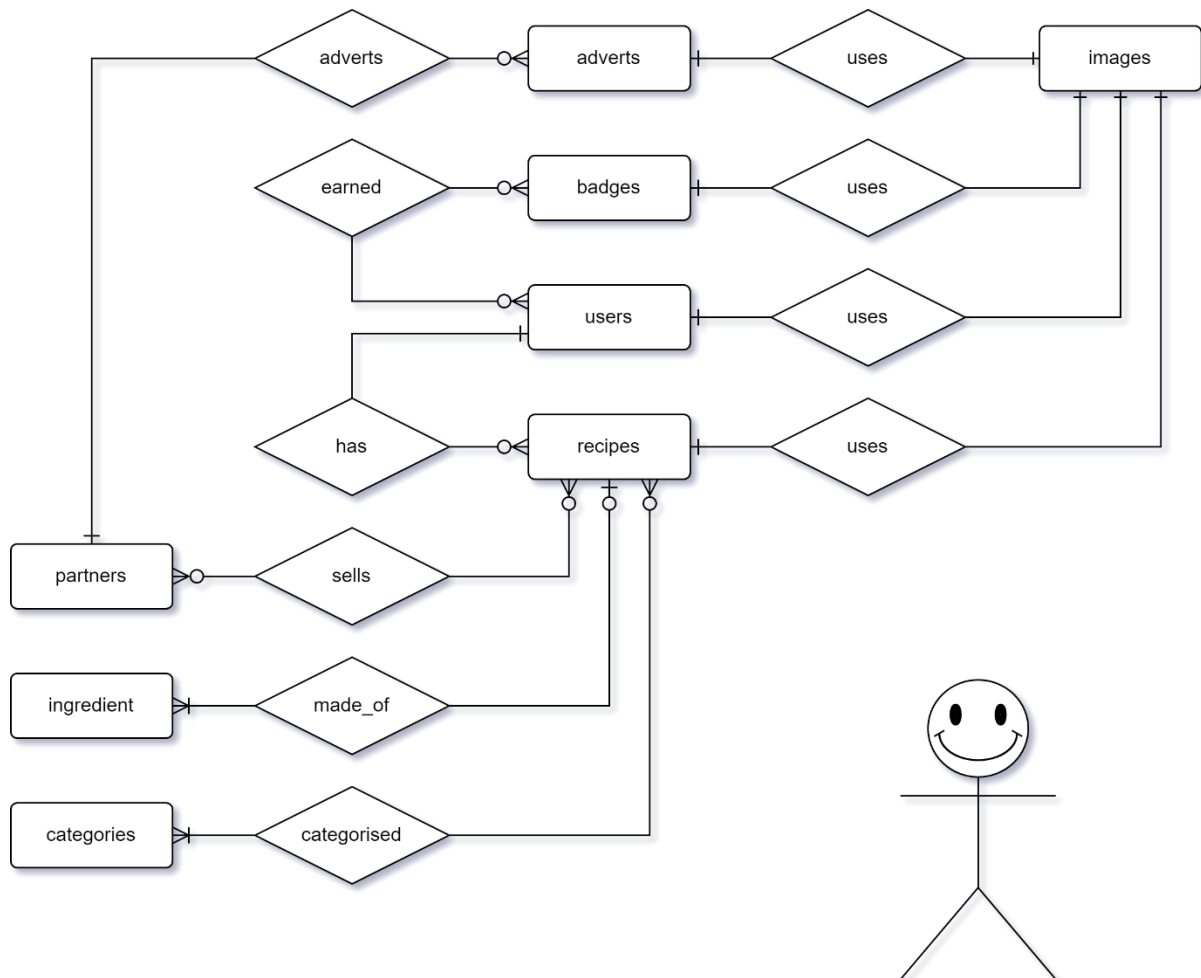
Anonymus, Pinia (2023.04.28.) <https://pinia.vuejs.org/core-concepts/state.html>

Microsoft, Asp.Net (2023.04.28.) <https://learn.microsoft.com/en-us/aspnet/core/fundamentals/apis?view=aspnetcore-7.0>

Infojegyzet (2023.04.28.) (<https://www.infojegyzet.hu/webszerkesztes/dokumentacio/>)

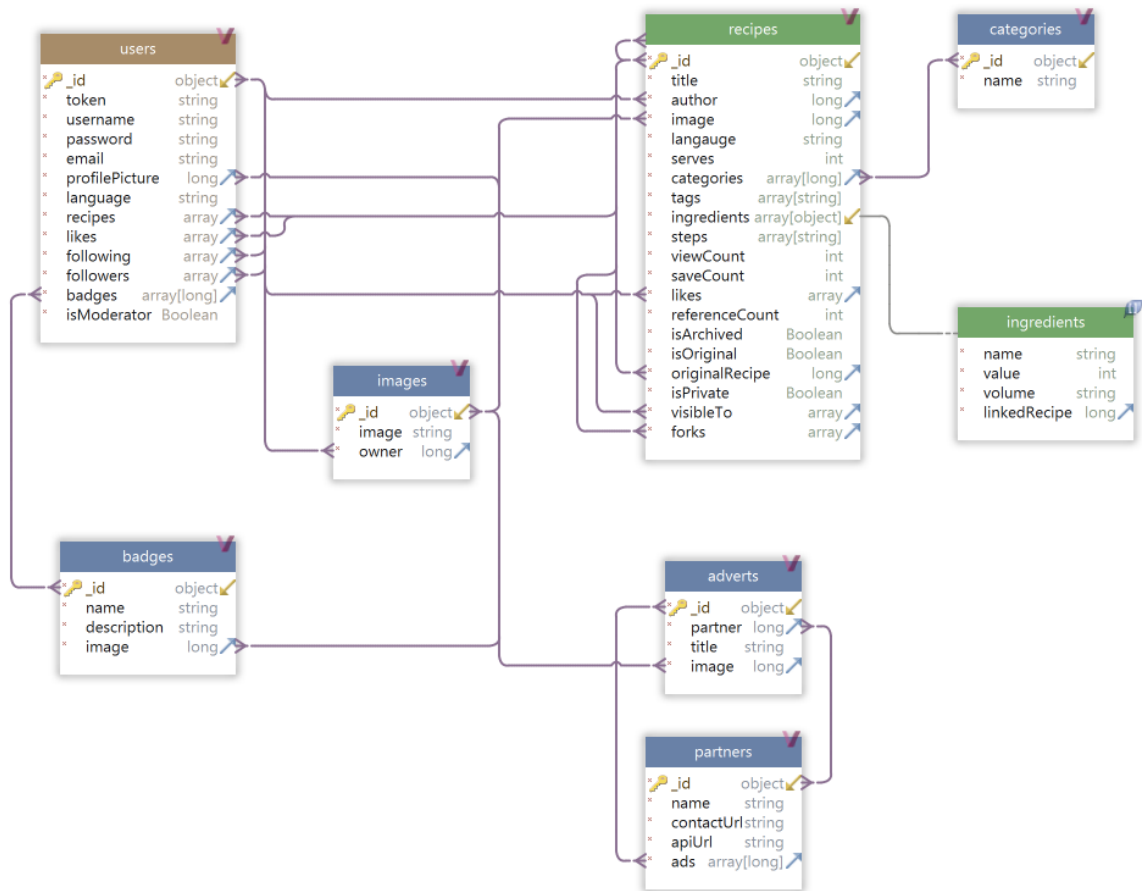
DigitalOcean, Backend (2023.04.28.) <https://www.digitalocean.com/>

7. Mellékletek



1. ábra

Gump



2. ábra