**Jedlik Ányos Gépipari és Informatika Technikum**

A képen embléma látható

Automatikusan generált leírás

Dokumentáció

Készítették:

Csepi Szilveszter

Hegyi Áron Ferenc

Menyhárt Attila

Tartalomjegyzék

[1. A vizsgaremekről 4](#_Toc133540348)

[1.1. Mi az a Gump? 4](#_Toc133540349)

[1.2. Honnan jött a név? 4](#_Toc133540350)

[1.3. A Gump-ről bővebben 4](#_Toc133540351)

[1.4. Többnyelvűség 4](#_Toc133540352)

[1.5. A jövő 4](#_Toc133540353)

[1.6. Motiváció 5](#_Toc133540354)

[1.7. Célközönség 5](#_Toc133540355)

[2. Fejlesztési dokumentáció 6](#_Toc133540356)

[2.1. Csapatmunka 6](#_Toc133540357)

[2.1.1. Munkamegosztás 8](#_Toc133540358)

[2.2. Technológiák 9](#_Toc133540359)

[2.3. Szoftverek, szolgáltatások 9](#_Toc133540360)

[2.4. Publikálás 10](#_Toc133540361)

[2.5. Fejlesztői környezet 11](#_Toc133540362)

[2.6. Adatbázis 11](#_Toc133540363)

[2.6.1. Az adatbázis felépítése 11](#_Toc133540364)

[2.6.2. Táblák szerkezete 11](#_Toc133540365)

[2.7. UI design 14](#_Toc133540366)

[2.8. Backend 15](#_Toc133540367)

[2.8.1. Model 15](#_Toc133540368)

[2.8.2. Repository-k 15](#_Toc133540369)

[2.8.3. Controller-ek 16](#_Toc133540370)

[2.8.4. Autentikáció 18](#_Toc133540371)

[2.8.5. Autorizáció 18](#_Toc133540372)

[2.9. Frontend 19](#_Toc133540373)

[2.9.1. Technológiák: 19](#_Toc133540374)

[2.10. Tesztek: 21](#_Toc133540375)

[2.10.1. Backend: 21](#_Toc133540376)

[2.10.2. Frontend: 22](#_Toc133540377)

[3. Felhasználói dokumentáció 24](#_Toc133540378)

[3.1. Kompatibilitás 24](#_Toc133540379)

[3.1.1. Gump 24](#_Toc133540380)

[3.1.2. Localer 24](#_Toc133540381)

[3.2. Az alkalmazás telepítése 24](#_Toc133540382)

[3.2.1. Gump 24](#_Toc133540383)

[3.2.2. Localer 24](#_Toc133540384)

[3.3. A felület bemutatása – Gump 25](#_Toc133540385)

[3.3.1. Kezdőképernyő 25](#_Toc133540386)

[3.3.2. Regisztráció 25](#_Toc133540387)

[3.3.3. Bejelentkezés 25](#_Toc133540388)

[3.3.4. Navigációs sáv: 25](#_Toc133540389)

[3.3.5. Főoldal 26](#_Toc133540390)

[3.3.6. Recept megtekintése 26](#_Toc133540391)

[3.3.7. Keresés 27](#_Toc133540392)

[3.3.8. Készítés 27](#_Toc133540393)

[3.3.9. A készítés lépései 27](#_Toc133540394)

[3.3.10. Receptek 28](#_Toc133540395)

[3.3.11. Profil 28](#_Toc133540396)

[3.4. A felület bemutatása – Localer 29](#_Toc133540397)

[3.4.1. Fordító felület 29](#_Toc133540398)

[3.4.2. Moderációs felület: 31](#_Toc133540399)

[4. Demo 33](#_Toc133540400)

[4.1. Linkek 33](#_Toc133540401)

[4.2. Mintauser 33](#_Toc133540402)

[5. Összegzés 34](#_Toc133540403)

[6. Irodalomjegyzék 35](#_Toc133540404)

[7. Mellékletek 36](#_Toc133540405)

# A vizsgaremekről

## 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.

## Honnan jött a név?

Mikor az alkalmazás nevén törtük a fejünket segítségül hívtunk egy online névgenerátor programot. Rengeteg fantázianév közül a Gumption-re esett a választás, ami sütnivalót jelent és úgy gondoltuk ez illene az appukhoz leginkább. Végül az egyszerűség kedvéért a   
Gumption-t Gump-ra lerövidítettük. Innen jött a név.

## A Gump-ről 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 a diétás igényekhez.

A negyedik alappillért a partnereink és banner hirdetéseik képzik, amikből bevételszerzési lehetőségünk adódik.

Az alkalmazásunk segítségével a kliensek egyszerűen tudják elkészíteni receptjeiket. A beépített közösségi háló segítségével a felhasználók tudják egymást követni és a recepteket kedvelni.

## 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 bárkit használhat a fordítások elkészítéséhez. Ennek működése a !!!!!6.4-fejezetben van részeleten kifejtve.!!!!!!!!

## 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.

## 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.

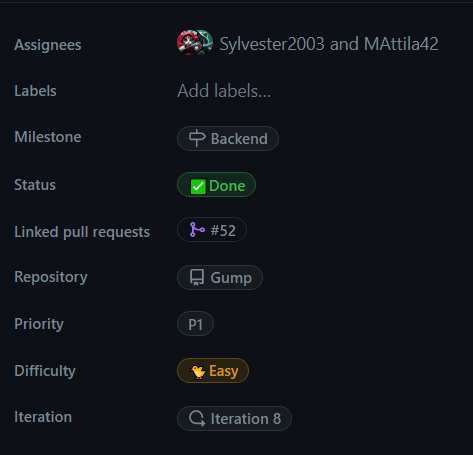
## Célközönség

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.

# Fejlesztési dokumentáció

## 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.

Miden 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.

A képen szöveg, monitor, képernyőkép, televízió látható

Automatikusan generált leírás

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.

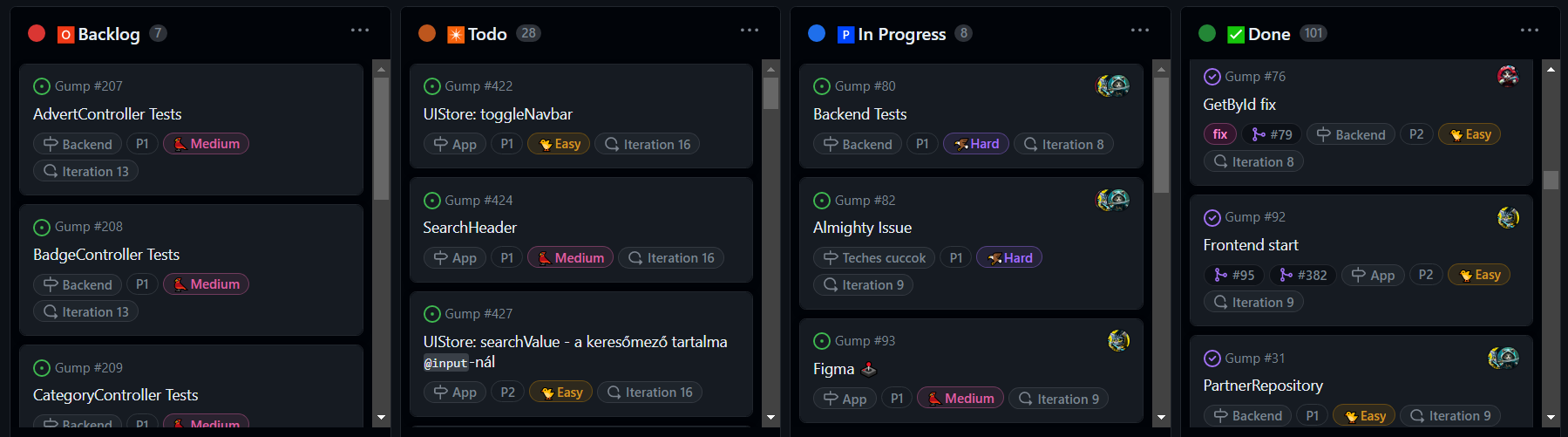
„Backlog-ba” kerültek azok a feladatok, amiket még nem alakítottunk át issue-ra. Ilyenek lehettek azok a tervek, amik még nem voltak alaposan kidolgozva és csak később szerettünk volna rá energiát fektetni.

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 mergelve 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 900 commit volt Githubon.

A Githubon a pull requestek létrehozásánál a módosított fájlokon 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.

### 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 |  |

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.

## 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.

## Szoftverek, szolgáltatások

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

* ***Visual Studo 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 tattintással sikerült átvinnünk a frontendre.
* ***Discord:*** Itt valósítottuk meg a projekt részleteit érintő írásos és szóbeli kommunikációt.
* ***Capacitor:*** Cross-platform native 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 (dependecy-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 konvenciós hibákat.
* ***Docker:*** A backendet a szerven Docker konténerben futtatjuk a teljesítmény javítása érdekében.
* ***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.

## 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:*** A frontendet pedig a Netlify-ra tettük fel, ami szintén ingyenes és könnyen kezelhető. Hasznos tulajdonsága, hogy automatikusan frissíti a weboldalt a Github repository alapján.

## 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.

## Adatbázis

### Az adatbázis felépítése

Ahogy azt a [2.2-es fejezetben](#_Technológiák) 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.

### Táblák szerkezete

1. 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

1. 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

1. 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

1. badges:

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

1. categories:

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

1. ingredient:

* \_id: Elsődleges kulcs; long
* name: A hozzávaló neve; string
* value: A hozzávaló mennyisége; short
* volume: A hozzávaló mennyiségéhez tartozó mértékegység; string
* linkedRecipe: A hozzávalót tartalmazó recept azonosítója; long

1. adverts:

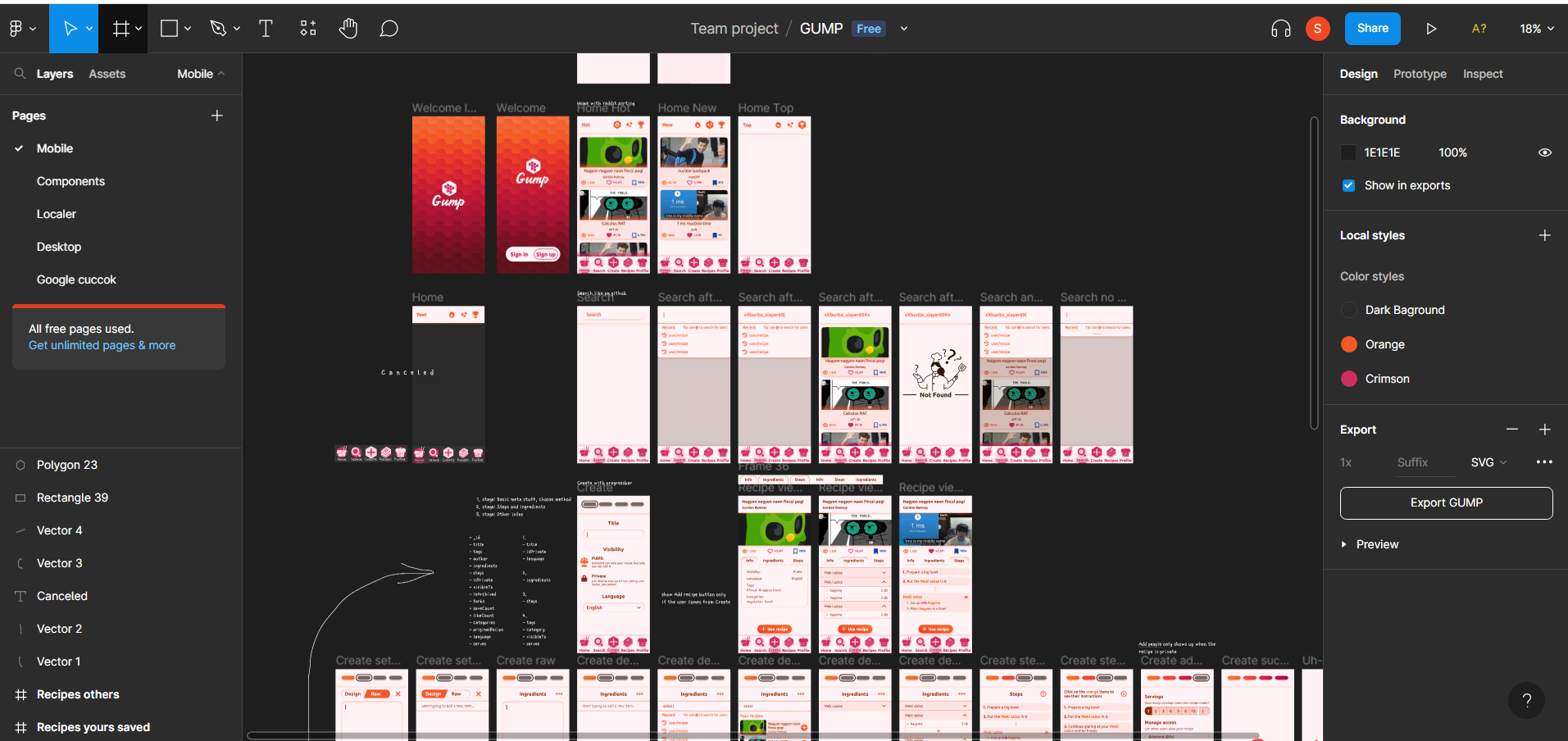
* \_id: Elsődleges kulcs; long
* partner: A hirdetéshez tartozó partner azonosítója; long
* title: A hirdetés címe; string
* image: A hirdetéshez tartozó kép azonosítója; long

1. images:

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

## UI design

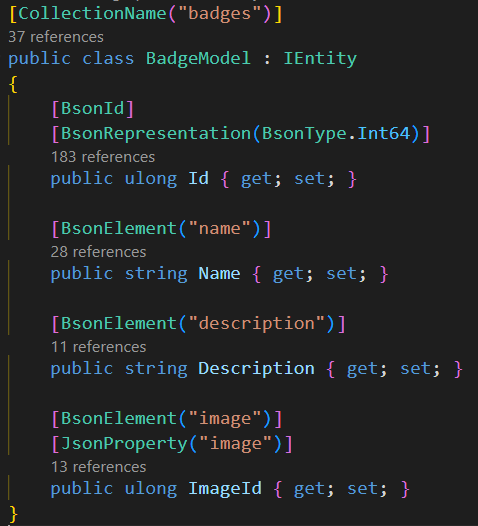
Az adatbázis modell megalkotása után a felhasználói felület, a UI tervezése következett. Ezt a Figma nevezetű alkalmazás segítségével valósítottuk meg. Nagy segítség volt, hogy az elkészített komponenseket svg vagy css kód formájában két tattintással sikerült átvinnünk a frontendre.



## Backend

### Model

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* gyűjteményhez tartozik. Láthatjuk, hogy *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 *BsonElementel* a mezőneveket definiáltuk.

Előfordult, – például az *ImageId* esetében – hogy külön *JsonProperty* attribútumot is megadtunk, hogy az API kérések során eltérő legyen a kulcs neve a property nevétől.

### 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 azokat a metódusokat, amik mindenhol jól jönnek ne kelljen mindenhol újra megírni.

Ilyen például a GetById metódus, amit mindegyik model használ.

A képen szöveg látható

Automatikusan generált leírás

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 adott id-vel nem létezik elem, akkor *NotFoundException* üzenettel befejezzük a metódus futását.

### Controller-ek

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

1. 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

1. Auth:

**Login** - Bejelentkezés

Elérési út: /auth/login

Típus: POST

Request body: LoginDto

Jogosultság: bárki

Hibakódok: 401 Unauthorized

1. 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>

### 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 30 perces lejárati idővel védekezünk.

### 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.

## 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.

### 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 képen diagram látható

Automatikusan generált leírás

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.

***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.

## A képen szöveg látható Automatikusan generált leírásTesztek:

### 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 hogy a kontroller megfelelő hibakóddal reagál hibás adat esetén.

Teszteknél létrehoztunk minta adatokat mindegyik Model estén a RepositoryTestsBase osztályban.

A BageModel-hez elkészített minta osztálypéldány:

A képen szöveg látható

Automatikusan generált leírás

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

A képen szöveg látható

Automatikusan generált leírás

!!!!!!!!!LEÍRNI!!!!!!!!!!!!!

### Frontend:

A frontend 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.

Graphical user interface, text

Description automatically generated

# Felhasználói dokumentáció

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

## Kompatibilitás

### Gump

* Operációs rendszer: Android 5.0+
* Memória: 1GB+
* CPU: 1GHz+

### 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 letesztelve!

## Az alkalmazás telepítése

### Gump

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

### Localer

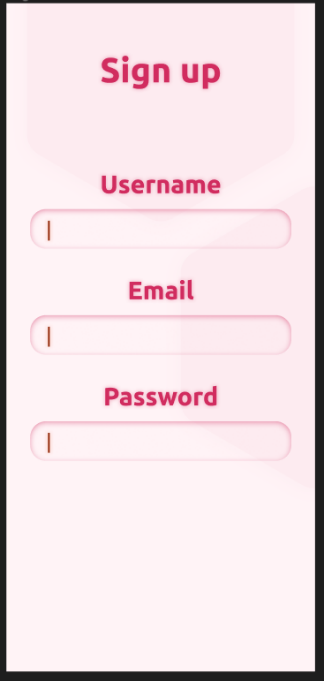
* A Localer az alábbi webcímen érhető el: <https://gump.live/>

## A felület bemutatása – Gump

### Kezdőképernyő

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

### 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. A jelszóra nincs megkötés. Hiba esetén a felhasználó értesítve van.

### Bejelentkezés

Bejelentkezni kizárólag érvényes felhasználónév és jelszó párossal lehetséges.

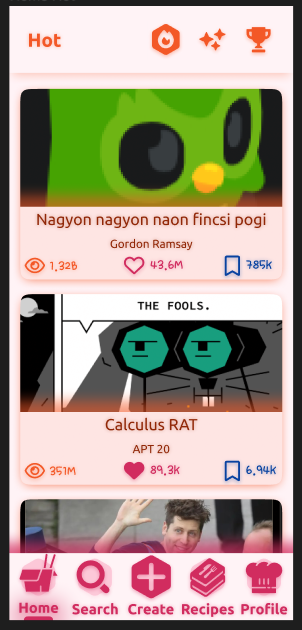
### Navigációs sáv:

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



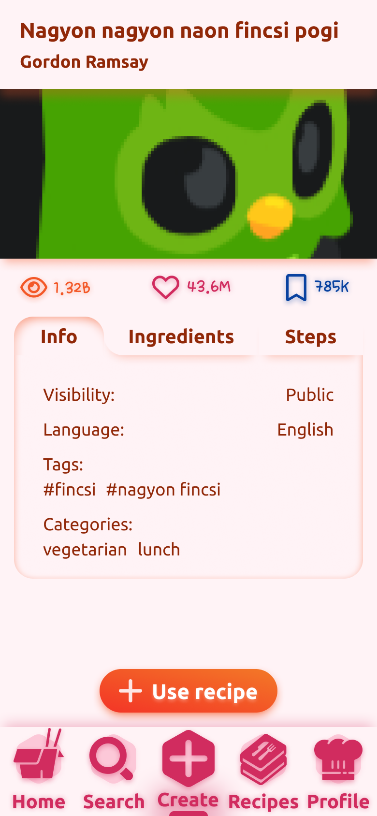
### Főoldal

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 mentettek le.

### 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.

### 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.

### Készítés

A receptek elkészítésére 2 megadási mód létezik, a minimális és a komplex. Előbbi esetében nem hozzávalókat és lépéseket adunk meg, hanem meglévő recepteket és azokból hozzuk létre a kész receptet.

Utóbbit választva a szerkesztő megadja az összes hozzávalót és kifejti a lépéseket tömören, egyértelműen.

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

### A készítés lépései

1. Alap adatok:

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

1. 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.

1. Lépések:

Itt kerülnek felsorolásra az étel elkészítéséhez szükséges lépések.

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

1. 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. 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.

### Receptek

Az alábbi receptek jelennek meg ebben a menüpontban.

* Saját receptek
* Kedvelt receptek
* Elmentett receptek

### 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

## A felület bemutatása – Localer

Az alábbi navigációs menüpontok érhetőek el:

A képen embléma látható

Automatikusan generált leírás

* 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.

### Fordító felület

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

1. Kezdőlap:

A képen szöveg látható

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

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 országkódok listája ezen oldalon érhető el (Two Letter): <https://www.fincher.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.

1. Fordítás készítése:

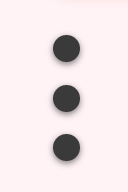
Egy kulcs kiválasztásakor az alábbi felület jelenik meg:



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. Létező kulcsra kell mutatnia, különben hibaüzenetet dob vissza az alkalmazás.

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

Hibás működés esetén a jobb oldalon lévő visszajelző lámpák mutatják a hibakódokat, melyek jelentéséről a fejlesztők szolgálhatnak további információval.

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

A kulcsok melletti számok a fordítás elkészültségét mutatják.

### Moderációs felület:

1. Receptek moderálása:

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

1. Felhasználók moderálása:

* Az összes felhasználó kilistázásra kerül.
* Műveletek:
  + Felhasználók törlése

1. Partnerek moderálása:

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

1. Hirdetések moderálása:

* Az összes hirdetés kilistázásra kerül.
* Műveletek:
  + Új Hirderés létrehozása
  + Hirdetés adatainak módosítása
  + Hirdetés törlése

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

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 igényeit a moderátor felé kell jeleznie.

# Demo

## 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>

## Mintauser

Tesztelési célokra az alábbi felhasználói bejelentkezések érhetőek el:

* TestUser:
  + Felhasználónév: TestUser
  + Jelszó: secret
* Moderator:
  + Felhsználónév: Moderator
  + Jelszó: moderator001

# Összegzés

A fejlesztés során rengeteg kihívással kerültünk szembe. Voltak olyanok, amiket pár perc, egyeseket 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 belekezdtünk. Az biztos, amit idén tanultunk az életben nagy hasznukra lesz.

# Irodalomjegyzék

# Mellékletek

A képen diagram látható

Automatikusan generált leírás

1. ábra

A képen diagram, sematikus rajz látható

Automatikusan generált leírás

2. ábra

Algoritmusok:

Validate fields:

A képen szöveg látható

Automatikusan generált leírás

Frontend – Check dirty

A képen szöveg, válaszfal, képernyőkép, ezüst látható

Automatikusan generált leírás

Sikeres mentés esetén az alábbi visszajelzés olvasható: !!!!!!!!!!!!!!

A képen diagram látható

Automatikusan generált leírás