



Készítette:

Gyöngyösi Róbert

Jakab Eduárd

Jakab Evelin

Pap Roland Levente

Szak: Infokommunikációs hálózatok és rendszerek (Távközlés) III.év

Tantárgy: Szoftverrendszerek tervezése

Vezető tanár: Ferencz Katalin, Dr. Szántó Zoltán

Sapientia Erdélyi Magyar Tudományegyetem

Marosvásárhelyi kar, 2023-2024

Tartalomjegyzék

1. BEVEZETŐ.....	3
2. A PROJEKT CÉLJA	3
3. KÖVETELMÉNY SPECIFIKÁCIÓ	4
3.1. FELHASZNÁLÓI KÖVETELMÉNYEK -> USE CASE DIAGRAM	4
3.2. RENDSZERKÖVETELMÉNYEK	5
3.2.1. <i>Funkcionális követelmények</i>	5
3.2.2. <i>Nem funkcionális követelmények</i>	5
3.2.3. <i>Termék követelmény:</i>	5
3.3. FONTOSABB MŰVELETEK MAGYARÁZATA.....	5
3.3.1. <i>Felhasználó regisztrációja</i>	6
3.3.2. <i>Felhasználó bejelentkezése</i>	7
4. TERVEZÉS.....	7
4.1. WIREFRAME	7
4.2. ARCHITEKTÚRA.....	8
4.3. MODULOK LEÍRÁSA.....	9
4.3.1. <i>Adatbázis</i>	9
4.3.2. <i>Aktivitás diagramok</i>	11
a. <i>Aktivitás diagram a bejelentkezéshez</i>	11
b. <i>Aktivitás diagram a vásárlás – és eladáshoz</i>	12
4.4. PROJEKT MENEDZSMENT - KANBAN	13
4.5. VERZIÓKÖVETÉS - GITHUB	14
5. ALKALMAZÁS MŰKÖDÉSE.....	17
5.1. UI – KONKRÉT MEGVALÓSÍTÁS.....	17
6. ÖSSZEGZÉS	21
6.1. TOVÁBBI FEJLESZTÉSI LEHETŐSÉGEK.....	21
7. BIBLIOGRÁFIA/HASZNÁLATI TOOL-OK	21

1.Bevezető

Minden ember tapasztalhatta már élete során, hogy nincs ideje elmenni az üzletbe, piacra, stb... Megtapasztalhatta már azt is, hogy az üzletekben a zöldség, illetve a gyümölcsök íze és minősége nem olyan volt mint a hazai termelőké. Ezek mellett, a kis termelők gondjai, hogy nem tudják, hogy hogyan adhatnák el könnyebben illetve gyorsabban a terményeiket.

Igy mi ki talátuk, hogy hogyan könnyíthetnénk meg a vásárlók vásárlásait és a kistermelők eladásait. Létre hoztunk egy felületet, a FrissKert-et, amely működik weboldalon. A termelők könnyedén feltölthetik a FrissKert oldalára a terményeiket, amelyeket más felhasználók is láthatnak az oldalon a regisztrálás után. Így könnyedén eladhatnak és vásárolhatnak a felület felhasználói.

Fontos kiemelni, hogy egy weboldalon mindig lehet valamit fejleszteni, javítani. Sok kiegészíteni való dolog van még a weboldalon, de az alábbi fázisban a következő funkciók kerültek megvalósításra:

- UI felület
- Bejelentkezés/ regisztráció
- Vásárolható termékek megtekintése
- Új termék feltöltése
- Kapcsolatfelvétel az eladóval e-mailen

2.A projekt célja

Projektünk célja, hogy hozzá segítsük az embereket a friss zöldség és gyümölcs vásárlásához direkt a kis termelők kertjéből, ugyanakkor ezzel segítsük a termelők terményeinek eladásait is a modern eszközök használatával. A termelő megoszthatja a terményeit az applikációban/ weboldalon, amelyeket a többi felhasználó megnézheti és láthatja, hogy hol helyezkedik el az a termék, és felveheti a kapcsolatot a termék termelőjével email illetve a megadott telefonszámon keresztül.

A projekt tervezésekor a következő célok kerültek megfogalmazásra:

- megkönnyíti a vásárlók mindennapi vásárlásait
- segít a friss és házi termékek (zöldségek illetve gyümölcsök) egyszerűbb elérését
- gyorsabb és hatékonyabb vásárlás
- egyszerű interakció a felhasználó és az adott használati felület között

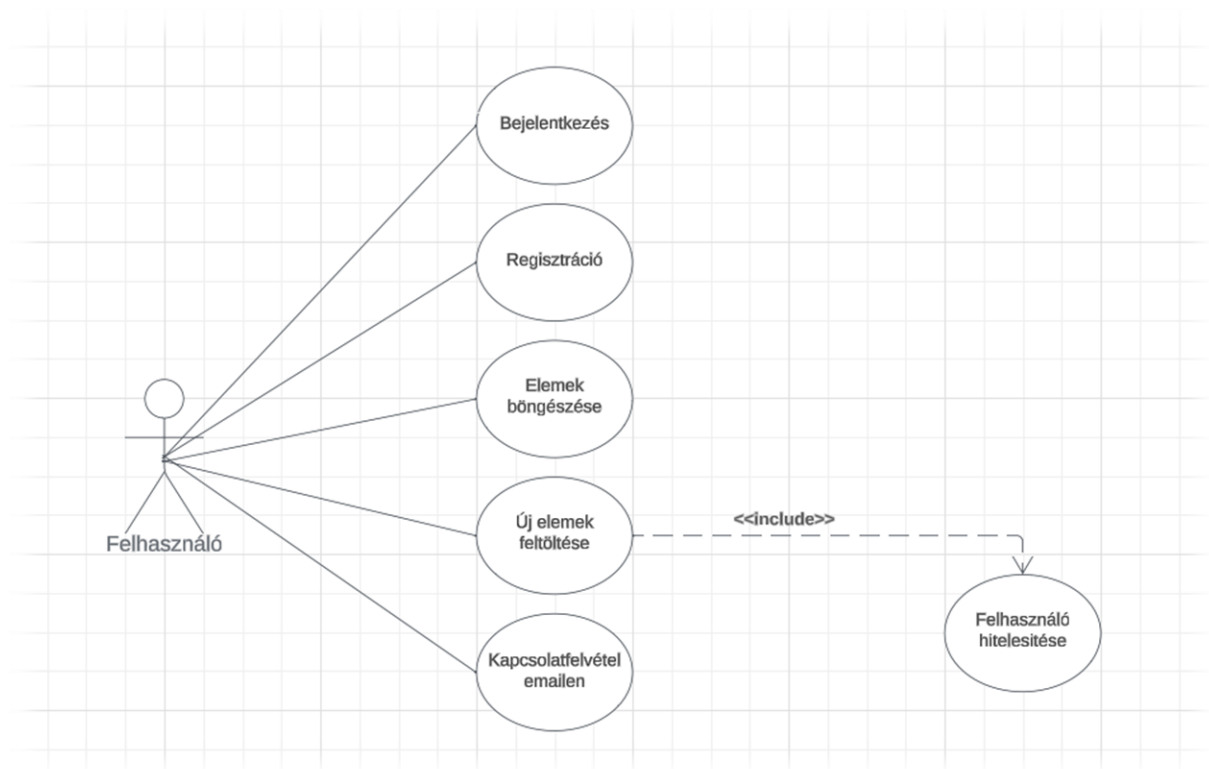
3.Követelmény specifikáció

3.1. Felhasználói követelmények -> Use Case Diagram

Az alábbi ábrán (1.ábra) a Használati Eset diagram (Use Case diagram) a felhasználó lehetséges interakcióinak ábrázolása a rendszerrel. A Use Case diagram különböző használati eseteket és különböző típusú felhasználókat mutat be, amelyekkel a rendszer rendelkezik.

- Regisztráció: Az alkalmazás használata kizárólag regisztrált felhasználók számára engedélyezett. Az alkalmazás megnyitása után a Regisztráció gombon keresztül érjük el, azt hogy tudjunk regisztrálni. Regisztrációhoz a felhasználónak névre, telefonszámra, lakcímre, email címre és jelszóra van szüksége.
- Bejelentkezés: Ha már regisztrált a felhasználó, utána a bejelentkezésnél már csak email címre és a jelszóra lesz szüksége a bejelentkezéshez.
- Elemek böngészése: Bejelentkezés után a FrissKert felületén már lehet az eladók és a terményeik között keresgélni.
- Új elemek feltöltése: Új zöldségeket illetve gyümölcsöket lehet feltölteni a weboldalra.
- Kapcsolatfelvétel emailen: Email cím által kommunikálhatnak az eladó és a vásárló egymás között.

A projekt esetében a használati eset diagram a következő:



1.ábra: Use Case Diagram

3.2. Rendszerkövetelmények

A hatékony használathoz minden számítógépes szoftvernek bizonyos hardverkomponenseknek vagy szoftvererőforrásoknak van szüksége. Ezeket az előfeltételeket rendszerkövetelményeknek nevezzük.

A rendszerkövetelményeken belül megkülönböztetünk funkcionális és nem funkcionális követelményeket.

3.2.1. Funkcionális követelmények

- Az alkalmazás használata regisztrációhoz kötött, amely egy érvényes névvel, telefonszámmal, lakcímmel, email címmel és jelszóval valósítható meg.
- Bejelentkezni a már regisztrált felhasználók tudnak a regisztrációkor megadott email címmel és jelszóval.
- A felhasználók Email címmel vannak azonosítva.
- Adatok tárolása és megjelenítése a weboldal leírása szerint.
- Adatok bevitele és ezek mentése.
- Az alkalmazás lehetőséget ad saját termények eladására/ vásárlására.
- Az alkalmazáson belül az eladók oldalán megtekinthetjük az eladandó termékek listáját.

3.2.2. Nem funkcionális követelmények

- Asztali gép vagy laptop
- internet kapcsolat
- Böngésző: Chrome 120.0.6099.131+, Safari 17.0+ (a popup ablakok háttérének blur-je nem jelenik meg)
- Windows 10 pro+ vagy macOS 14.0+ operációs rendszer
- Windows: XAMPP 8.2.12, macOS: XAMPP 8.2.4
- Tárhely: kb. 500 MB
- Fejlesztői környezet: Visual Studio Code 1.85.1
- Fejlesztési nyelvek: HTML, CSS, JavaScript

3.2.3. Termék követelmény:

- megközelíthetőség, azaz könnyű kezelés a felhasználó számára
- reszponzív felhasználói interfész
- kijelző méret szerint adaptív felhasználói interfész

3.3. Fontosabb műveletek magyarázata

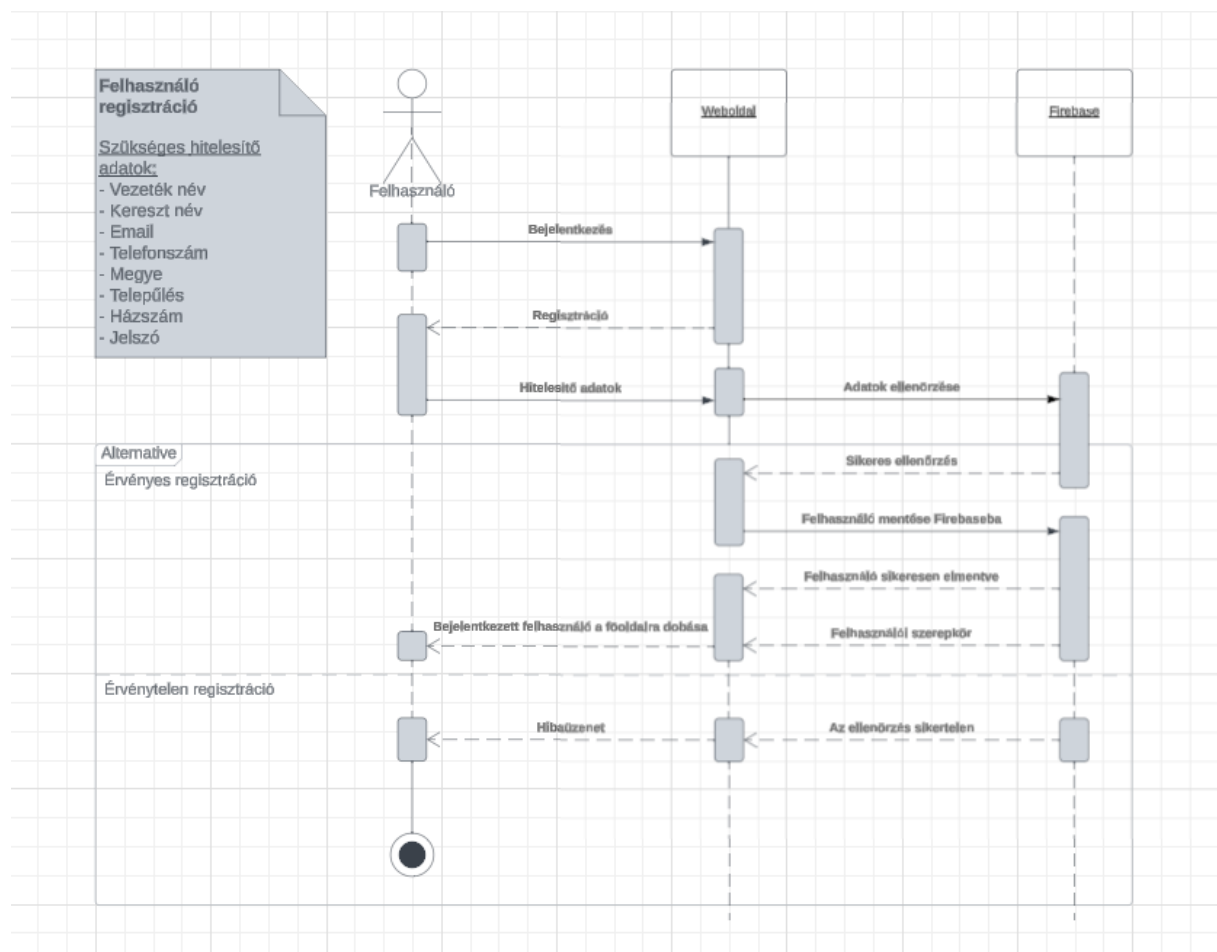
A fontosabb műveletek magyarázatát a szekvencia diagramok segítségével mutatjuk be. A szekvencia diagram feladata objektumok egymás közti üzenetváltásainak ábrázolása egy időtengely mentén elhelyezve.

3.3.1. Felhasználó regisztrációja

A alábbi ábrán (2.ábra) található a felhasználó regisztrációjának folyamata egy egyszerűsített szekvencia diagram formájában. A FrissKert oldalára be nem jelentkezett felhasználó tud magának készíteni egy profilt a regisztrációhoz szükséges adatok megadásával, amelyek a következők:

- Vezeték név
- Kereszt név
- Email
- Telefonszám
- Lakcím: megye, teletülés, házszám
- Jelszó

Az adatok ellenőrzése után a felhasználó elmentődik az adatbázisunkba, amelyet Firebase-be valósítottunk meg és ez után a felhasználó korlátlanul használhatja a FrissKert weboldalát. Ha nem megfelelő valamelyik mező, akkor a rendszer hibaüzenetet ad vissza és újra kell próbálkoznia.

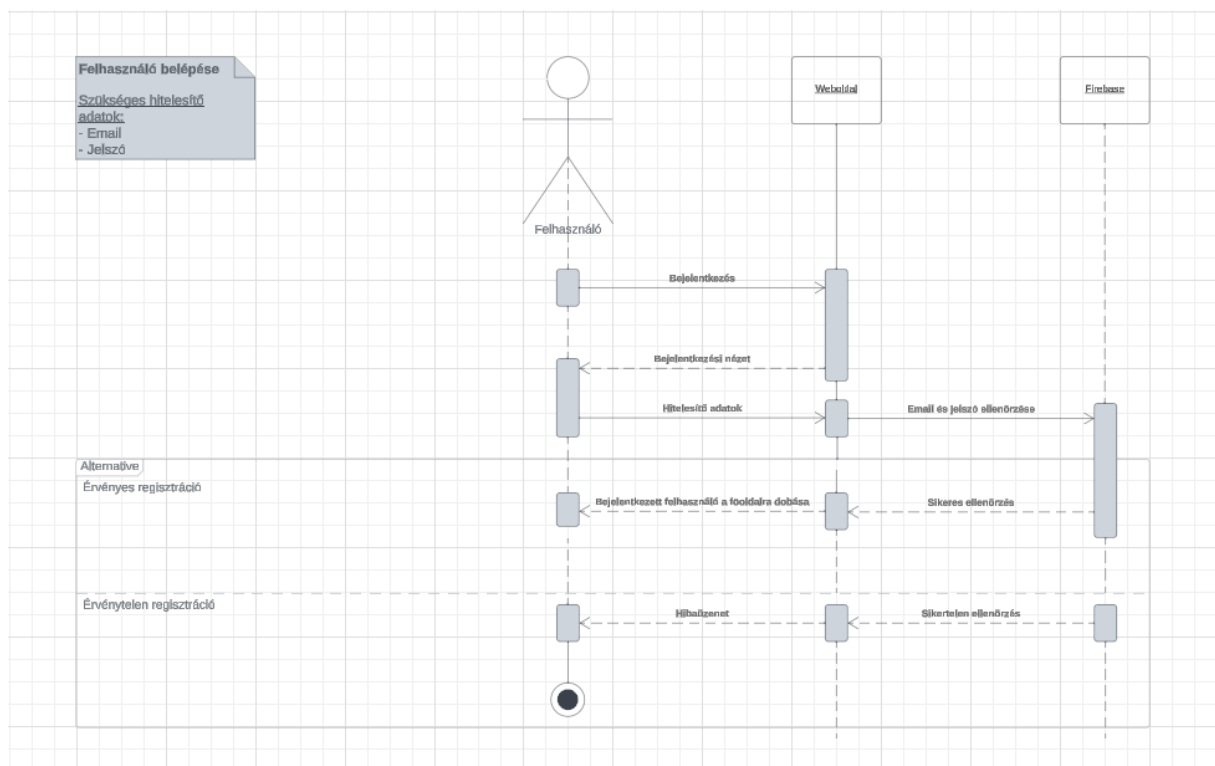


2.ábra: Felhasználó regisztrációja

3.3.2. Felhasználó bejelentkezése

Az alábbi ábrán (3.ábra) található a felhasználó bejelentkezésének folyamata szintén egy egyszerűsített szekvencia diagram segítségével bemutatva. Ez hasonlóan működik, mint a fentebb említett felhasználó regisztrációja, annyi hogy a felhasználó már létezik az adatbázisunkban így nem kell újra regisztrálnia, csak bejelentkeznie.

A bejelentkezés folyamata, hogy a felhasználó megadja a saját email címét és jelszavát. A rendszer a megadott adatokat hitelesíti s majd ellenőrzi azokat. Ez után a rendszer belép a FrissKert főoldalára, ha sikeres az ellenőrzés. Viszont, ha a bejelentkezés sikertelen, akkor a rendszer vissza fog küldeni egy hibaüzenetet.

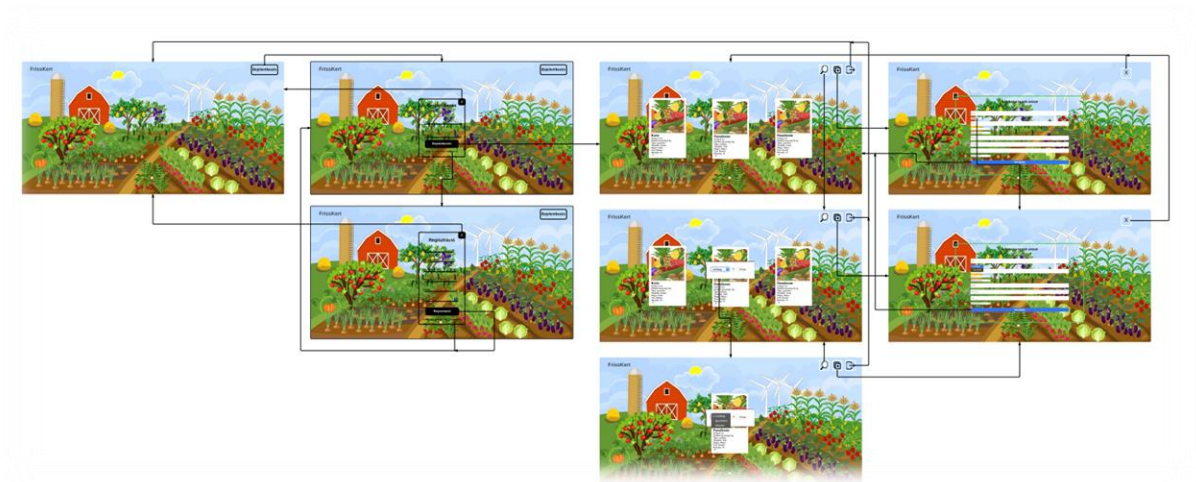


3.ábra: Felhasználó bejelentkezése

4. Tervezés

4.1. Wireframe

A lent látható ábrán (4.ábra) Wireframe fontos szerepet játszott a weboldal tervezésében és a kivitelezés során sok változáson ment keresztül, amíg eljutottunk a végleges formátumig. Ez a Wireframe Lucidchart modellező programban volt elkészítve, ami kezdetleges nehézségek ellenére, viszonylag könnyű megvalósítást eredményezett.



4.ábra: Wireframe

4.2. Architektúra

A alábbi ábra (5.ábra) a weboldal architektúrája figyelhető meg. A következő komponensek és ezen komponensek kapcsolataiból tevődik össze:

- Felhasználók (Users)
- Böngésző : Web felhasználói felület (Browser user interface)
- Adatbázis
- WebServer (XAMPP-Apache)

1. Felhasználók

A felhasználók egy laptopon vagy asztali gépen található böngésző segítségével kerül interakcióba a webalkalmazással.

A weboldalra két típusú szerepkörrel rendelkező felhasználó regisztrálhat, valamint jelentkezhet be: vásárlók és eladók. A szerepkörtől, valamint a szándéktól függetlenül a felhasználóknak regisztrálniuk kell és be kell jelentkezniük.

2. Felhasználói felületek

Ez a része a weboldal megjelenítésért felel, más szóval lehetővé teszi a felhasználóknak a weboldal használatát.

Ez felelős az adatok megjelenítéséért: megjeleníti az termékeket, a termékek feltöltését.

A felhasználói felület a felhasználó által végzett interakciókat küldi az egyes függvényeknek, melyek végrehajtnak egy adott operációt és visszaadnak egy eredményt a felhasználói felületnek, ahol a felhasználó láthatja az interakciójának az eredményét. Ilyen például a termékek szűrése típus szerint: gyümölcs vagy zöldség. Ugyanakkor a felhasználói felület magába foglalja az alkalmazás design-ját.

3. Adatbázis

Az adatok tárolásáért felel, amelyekre a többi komponenseknek szükségük van, például termékek és felhasználók.

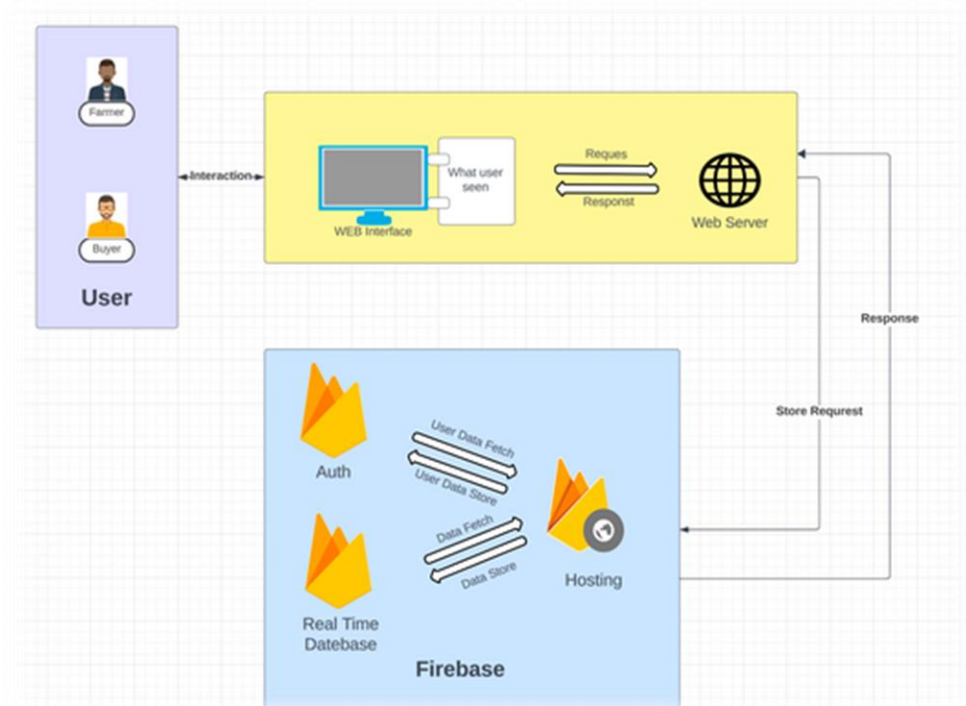
Egyiránt lehet adatokat feltölteni, tárolni, valamint tárolt adatokat lekérni.

A Firebase Realtime Database az interneten keresztül működik. Amikor a weboldal

FrissKert - Szoftverrendszerek Tervezése Projekt

használja a Firebase SDK-t (Software Development Kit) a kliensoldali kód kommunikál a Firebase szerverrel az interneten keresztül. Ezáltal az a weboldal valós időben jeleníti meg az adatbázisban eltárolt adatokat. Ez azt is jelenti hogy folyamatosan frissíti a weboldal állapotát ahogy valami változást észlel az adatbázisban. Ilyen változásokat okozhat például új termék feltöltése. Ez különösen fontos esetünkben mivel percenként tölthetnek fel új terméket az eladók.

Hasonlóképpen a Firebase egy szolgáltatása az Authentication, amely lehetővé tette az ügyfél azonosítás és hitelesítés funkciók használatát.



5.ábra: Architektúra diagram

4.3. Modulok leírása

A weboldal több elem komplex összetétele, melyek külön-külön fontos szerepet játszanak a működésben.

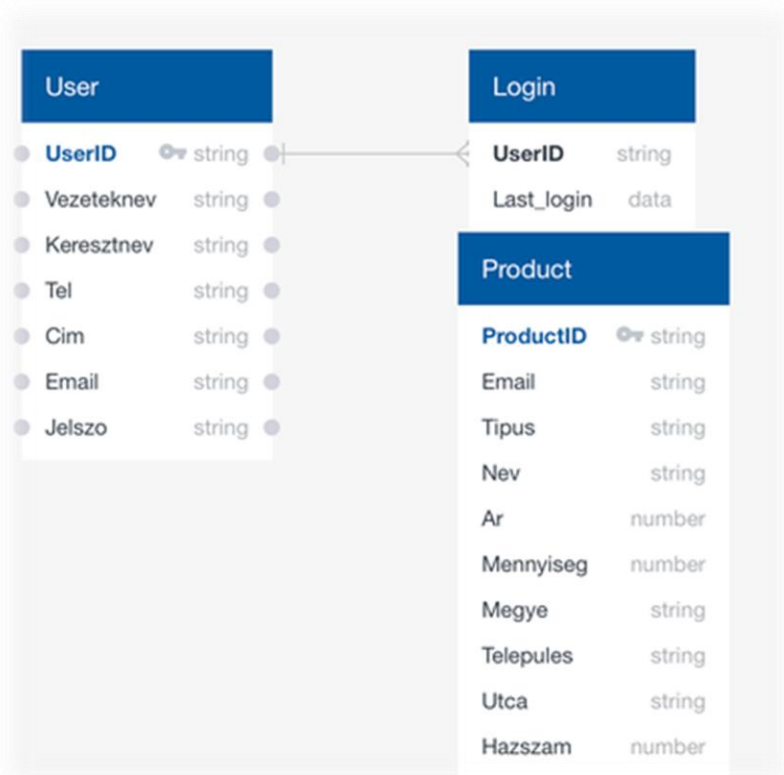
4.3.1. Adatbázis

Az adatok Firebase-ban vannak eltárolva, a bejelentkezést a Firebase Authentication oldja meg, valamint a többi adat eltárolására Realtime Database-t használtuk.

A Firebase Authentication átveszi a regisztrációs formból az email címet és a jelszót és létre hozza a felhasználó belépési adatait, illetve minden egyes felhasználónak generál egy egyedi UserID-t.

FrissKert - Szoftverrendszerek Tervezése Projekt

A Realtime Database-be a rendszer eltárolja a regisztrációs formából az adatokat: vezetéknév, keresztnév, email, telefonszám, lakcím, viszont kivételt képez a jelszó, mivel azt nem tárolja el a biztonság érdekében. Ugyanakkor a termékek adatainak mentését is ez szolgáltatja.



6.ábra: Adatbázis Quick Database Diagram-ban

A fenti ábrán (6.ábra) látható User táblázatban tároljuk el az új felhasználóról az adatokat:

- UserID
- Vezeték név,
- Kereszt név,
- Telefonszám,
- Lakcím,
- Email,
- Jelszó,

amelyeket a regisztráció során maga a felhasználó ad meg és amelyek a továbbiakban fontos szerepet játszanak a weboldal működésében.

A Login táblázatban a Last_login menti el, hogy a felhasználó mikor jelentkezett be utoljára a weboldalra.

A Product táblázatban látható adatmezők a következők:

- egyedi ID: ProductID
- Email
- Tipus (zöldség vagy gyümölcs)
- Név (a termék neve)

FrissKert - Szoftverrendszerek Tervezése Projekt

- Ár (Lej-ben)
- Mennyiség
- Lakcím: Megye, Település, Utca, Házszám: az a hely, ahol a megtalálható az eladandó termék

A termékek automatikusan generált egyedi ID-t kapnak, melyet a Firebase szolgáltatja.



7.ábra: Adatbázi a Realtime DataBase-be

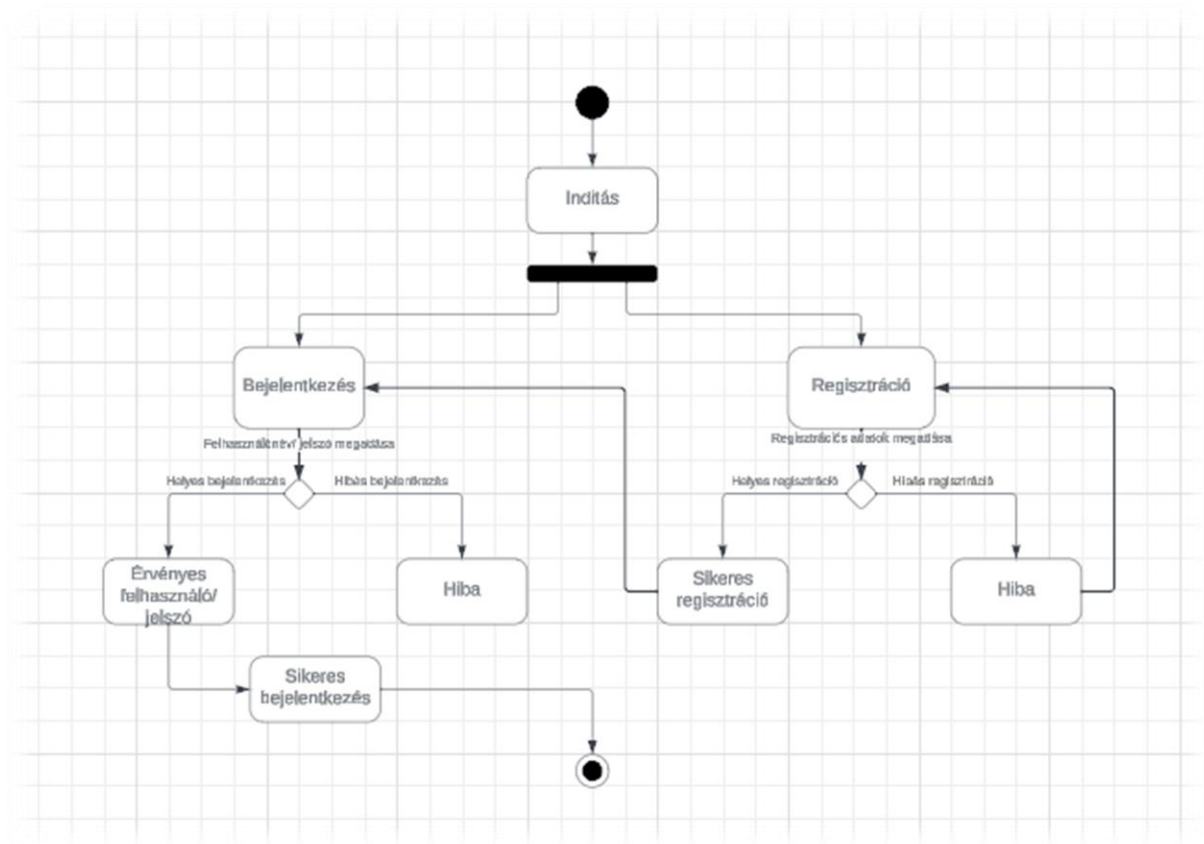
A fenti ábrán (7.ábra) a Firebase Realtime Database funkciója látható, a felhasználók, a termékek és a bejelentkezések adataival.

4.3.2. Aktivitás diagramok

a. Aktivitás diagram a bejelentkezéshez

Az alábbi ábra (8.ábra) az Aktivitás diagram a bejelentkezését mutatja be egy aktivitás diagram segítségével. A weboldal elindítása után a felhasználó megpróbál bejelentkezni a FrissKert oldalára. A rendszer le ellenőrzi, hogy hitelesek-e a felhasználó adatai. Ha helyesek az adatok, akkor a felhasználó sikeresen be tud jelentkezni és ezután már a FrissKert főoldalára viszi át. Viszont, ha nem tud sikeresen bejelentkezni, akkor a rendszer Hibát ad ki a felhasználónak, miután ő vagy újra próbálja a bejelentkezést vagy regisztrálja magát. Regisztrálás során is a rendszer ellenőrzi a beírt adatokat és eldönti róluk, hogy helyesek vagy sem. Ha nem sikeres a regisztráció, akkor a felhasználó újra kell próbálja a regisztrálást. Ha a

regisztrálás sikeres, akkor a továbbiakban már könnyedén bejelentkezhetsz a felhasználó és használhatja a FrissKert oldalát.



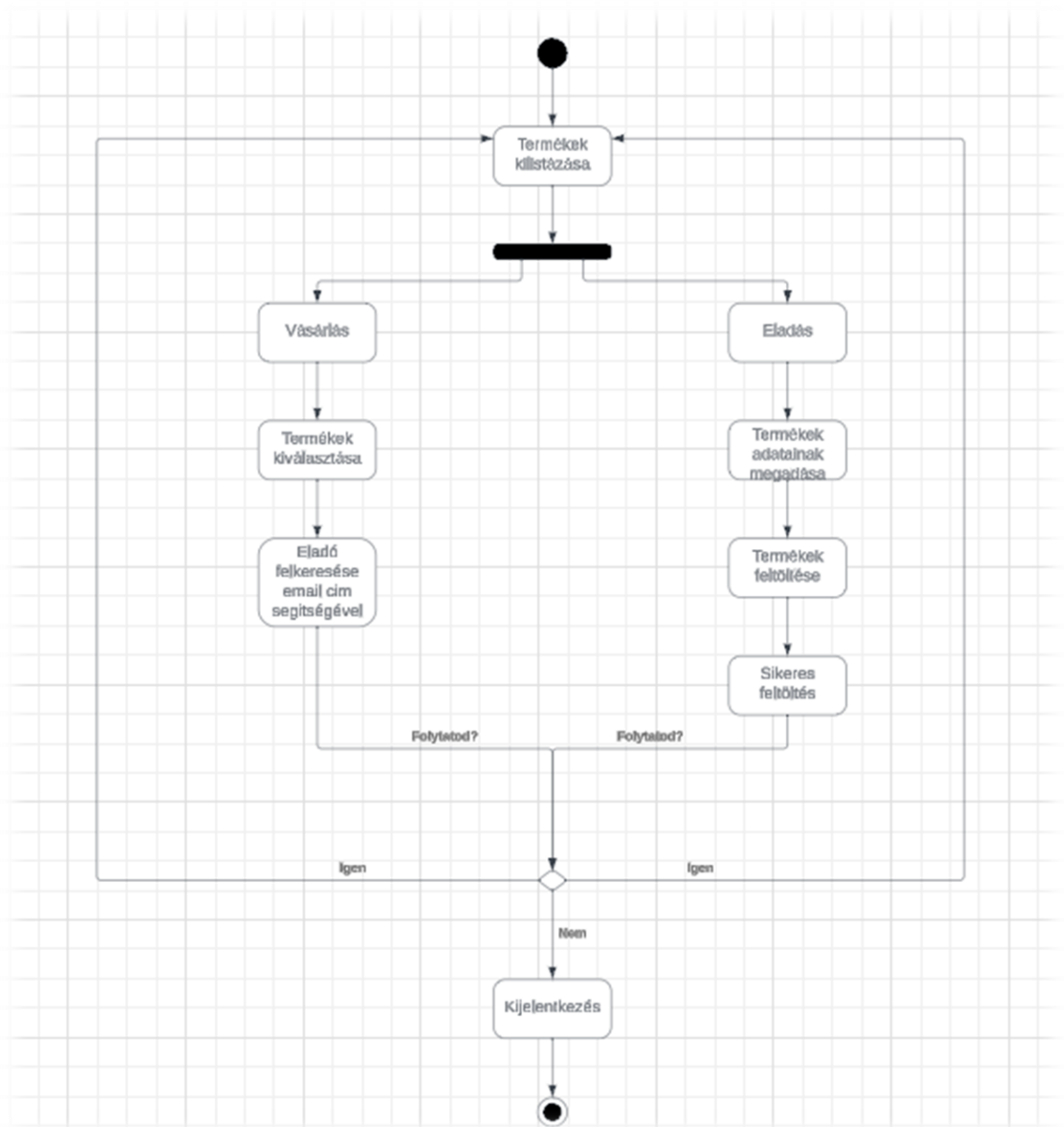
8.ábra: Aktivitás diagram a bejelentkezéshez

b. Aktivitás diagram a vásárlás – és eladáshoz

Az alábbi (9.ábra) ábra az Aktivitás diagram a termék vásárlás – és eladási lehetőségeit mutatja be. Miután a felhasználó sikeresen bejelentkezett, utána eldöntheti, hogy eladó vagy vásárló szeretne lenni.

Ha vásárló szeretne lenni, akkor az egyik lehetősége az, hogy szabadon böngészhet illetve keresgélhet az eladandó termékek között. Egy másik lehetőség a vevő számára, hogy kiválaszthatja azt, hogy gyümölcs vagy zöldséget szeretne nézni illetve vásárolni. Ezek után a vevő eldöntheti, hogy az eladandó áruk oldalán folytatja a termékek böngészését vagy kijelentkezik a FrissKert oldaláról.

Ha a felhasználó eladó szeretne lenni és ezáltal a termékeit árusítani, nincs más dolga, minthogy megadja a termékeinek az adatait. Miután az eladandó termék adatai beírásra kerülnek a termék feltöltése lépik érvénybe, és ha az eladó mindent jól csinált, akkor a rendszer vissza küld egy „Sikeres feltöltés” üzenetet. A továbbiakban az eladó is eldöntheti, hogy ott marad a FrissKert főoldalán vagy kijelentkezik.

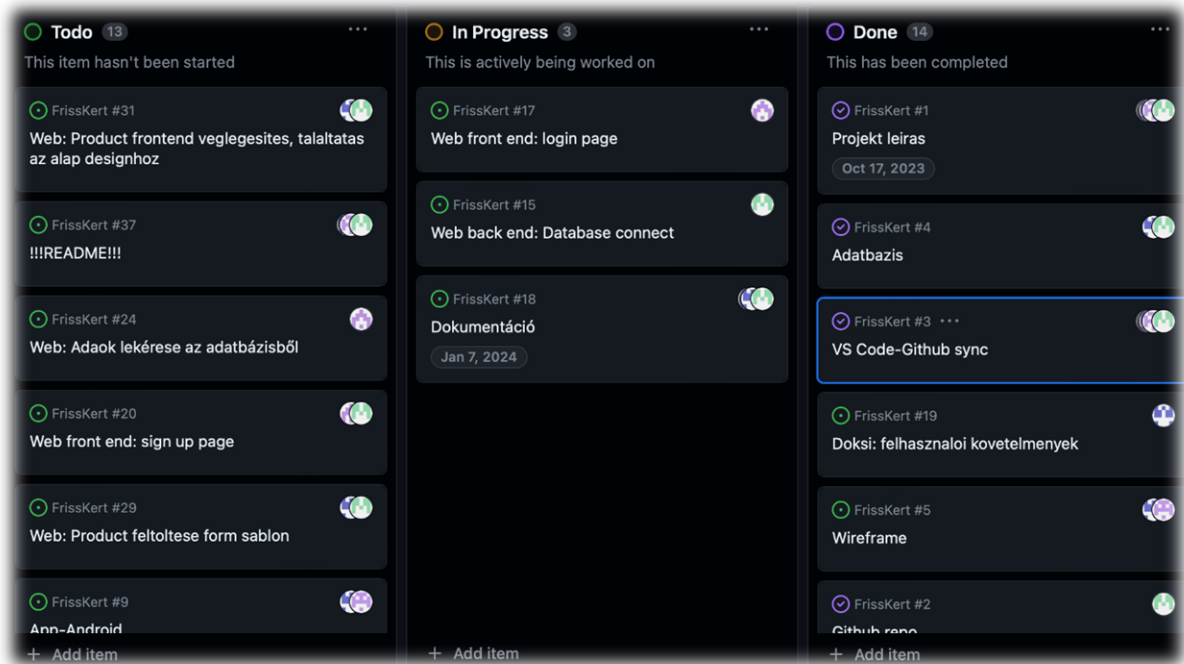


9.ábra: Aktivitás diagram a vásárlás – és eladáshoz

4.4. Projekt menedzsment - Kanban

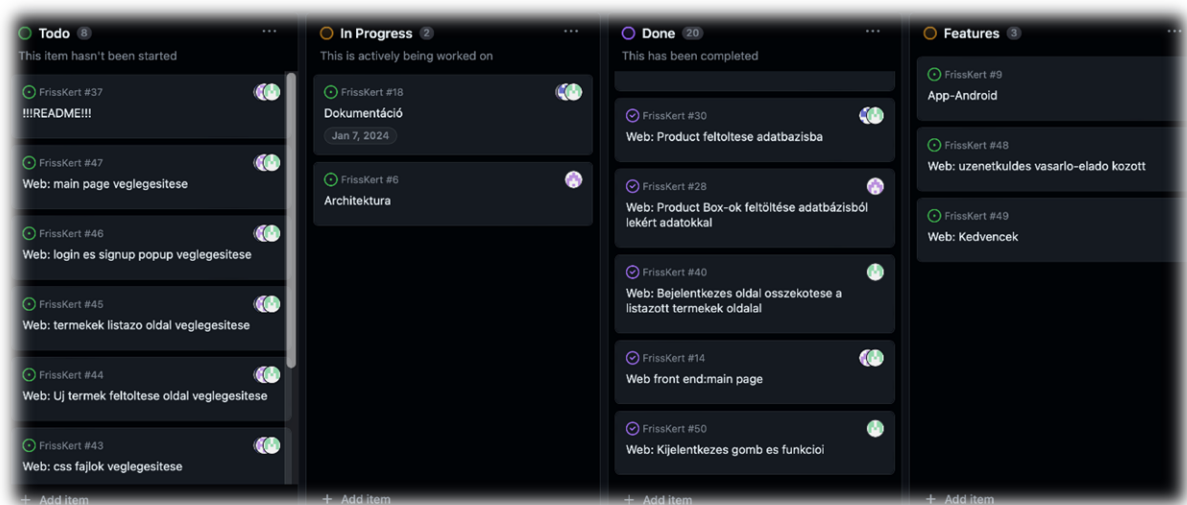
Csapat munkánkban a fejlesztés során fontos volt az együttműködés illetve ugyan olyan fontos volt a csapat irányítása valamint a kommunikáció is. Ahhoz, hogy mindezt meg tudjuk valósítani a Github volt a segítségünkre, amely egy kiváló eszköz a feladatok haladásának követésére és a munka szervezésre valamint ezek felosztásra.

FrissKert - Szoftverrendszerek Tervezése Projekt



10.ábra: Github Kanban

A fenti ábrán (10.ábra) a projekt megvalósítása közben látható egy pillanat kép a Kanban felületről.



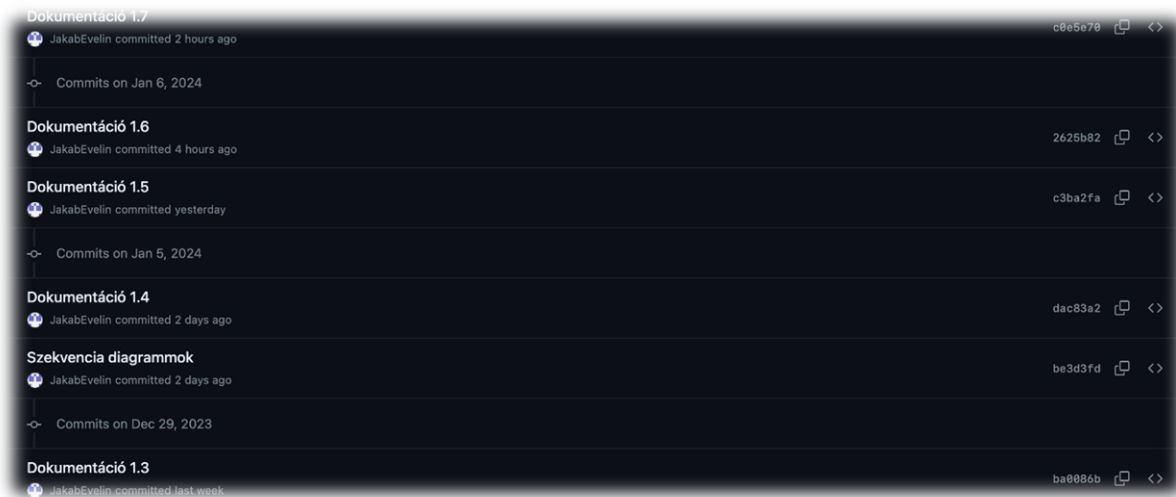
11.ábra:GitHub Kanban 2

Az előző ábrán (11.ábra) a Kanban felület az utolsó kód review előtti lépésben látható.

4.5.Verziókövetés - GitHub

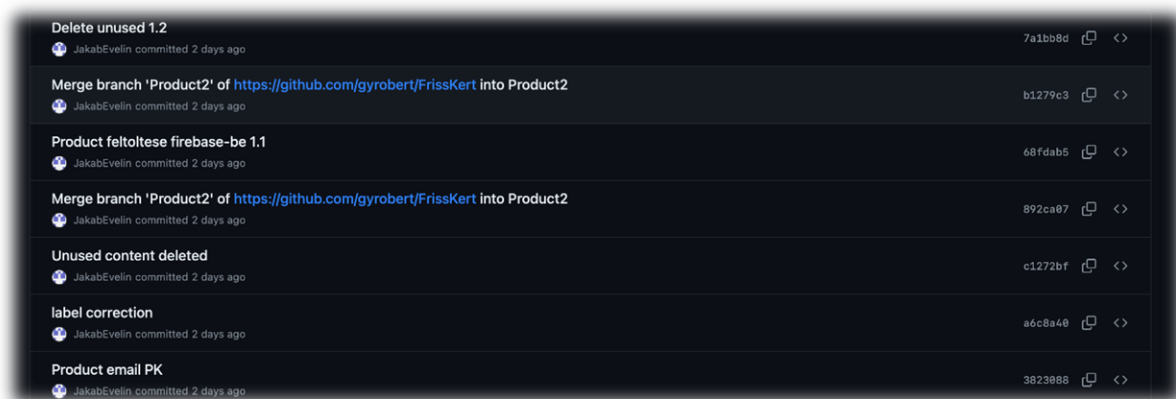
A projekt megvalósítása során nagyon fontos szerepet játszik a verziókövetés valamint kódok megosztása. Minderre a GitHub-ot használtuk segítségként. Létrehoztunk egy GitHub repository-t, és azon belül minden task-nak egy külön Branch-et annak érdekében, hogy átláthatóbb legyen külön-külön minden funkció fejlesztése.

FrissKert - Szoftverrendszerek Tervezése Projekt



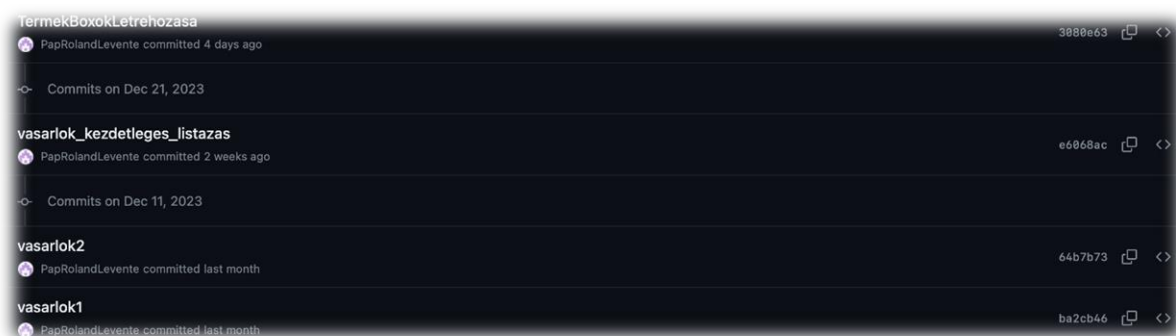
Dokumentáció 1.7	JakabEvelin committed 2 hours ago	c0e5e70		
Commits on Jan 6, 2024				
Dokumentáció 1.6	JakabEvelin committed 4 hours ago	2625b82		
Dokumentáció 1.5	JakabEvelin committed yesterday	c3ba2fa		
Commits on Jan 5, 2024				
Dokumentáció 1.4	JakabEvelin committed 2 days ago	dac83a2		
Szekvencia diagrammok	JakabEvelin committed 2 days ago	be3d3fd		
Commits on Dec 29, 2023				
Dokumentáció 1.3	JakabEvelin committed last week	ba0886b		

12.ábra: Dokumentáció Branch



Delete unused 1.2	JakabEvelin committed 2 days ago	7a1bb8d		
Merge branch 'Product2' of https://github.com/gyrobert/FrissKert into Product2	JakabEvelin committed 2 days ago	b1279c3		
Product feltoltese firebase-be 1.1	JakabEvelin committed 2 days ago	68fdab5		
Merge branch 'Product2' of https://github.com/gyrobert/FrissKert into Product2	JakabEvelin committed 2 days ago	892ca07		
Unused content deleted	JakabEvelin committed 2 days ago	c1272bf		
label correction	JakabEvelin committed 2 days ago	a6c8a40		
Product email PK	JakabEvelin committed 2 days ago	3023080		

13.ábra: Product2 Branch



TermekBoxokLetrehozasa	PapRolandLevente committed 4 days ago	3080e63		
Commits on Dec 21, 2023				
vasarlok_kezdetleges_listazas	PapRolandLevente committed 2 weeks ago	e6068ac		
Commits on Dec 11, 2023				
vasarlok2	PapRolandLevente committed last month	64b7b73		
vasarlok1	PapRolandLevente committed last month	ba2cb46		

14.ábra: WebTermékekListázása Branch

FrissKert - Szoftverrendszerek Tervezése Projekt



15.ábra: Web2 Branch



16.ábra: WebAdatbázis Branch

A fenti ábrákon (12,13,14,15,16.ábra) részletek találhatók a GitHub repositorynk néhány branch-ről, a fejlesztés során feltöltött commitokról. Néhány branch nevében megtalálható a 2-es szám, ami utal egyes funkciók drasztikusabb újragondolására és fejlesztésének újramezdésére, az átláthatóság érdekében.

5. Alkalmazás működése

5.1. UI – konkrét megvalósítás

```

submittingnup.addEventListener('click', (e) => {

    var nev = document.getElementById('nev').value;
    var knev = document.getElementById('knev').value;
    var tel = document.getElementById('tel').value;
    var cim = document.getElementById('cim').value;
    var email = document.getElementById('email').value;
    var password = document.getElementById('psw').value;

    createUserWithEmailAndPassword(auth, email, password)
        .then((userCredential) => {
            // Signed up
            const user = userCredential.user;

            set(ref(database, 'User signup/' + user.uid), {
                nev: nev,
                keresztnév: knev,
                tel: tel,
                cim: cim,
                email: email
            })
            alert('Felhasználó sikeresen létrehozva!');
            window.location.href = 'login2.html';
            // ...
        })
        .catch((error) => {
            const errorCode = error.code;

```

17.ábra: Regisztráció függvény

A fenti ábrán (17.ábra) látható kódrészleten a regisztráció függvény látható, ami a regisztráció gomb megnyomását figyelő EventListener-el kezdődik. Ez után a Form-ba beírt adatok átmentődnek lokális változókbá, amelyek az adatok Firebase-be való feltöltésében játszanak szerepet. Meghívódik a CreateUserWithEmailAndPassword Firebase belső függvénye, amely feltölti a felhasználó bejelentkezéséhez szükséges adatokat, email és jelszó, a Firebase Authentication szolgáltatáshoz és ugyanakkor a többi megadott adatot is, kivétel a jelszó, amely biztonsági okokból nem kerül feltöltésre, a Realtime Database szolgáltatásba. A Firebase minden feltöltött adatnak egyedi ID-t generál, ezzel megakadályozva a duplikátumot. A függvény végén a hibakezelés látható, ha minden rendben van akkor az adatok feltöltődnek a Firebase-be és kiír a képernyőre egy üzenetet, hogy “Felhasználó sikeresen létrehozva!”, ezzel jelezve a sikeres regisztrációt, majd visszatér a fő képernyőre.

Az alábbi ábrán (18.ábra) A Firebase konfiguráció látható, amely lehetővé teszi a kapcsolatot a Weboldal és a Firebase között. Ez alatt néhány inicializáció található, mint például az applikáció, az adatbázis, és az authentication inicializálása.

```
const firebaseConfig = {
  apiKey: "AIzaSyAwzqR6MQGlhKIIeU0v54vy31-K4f64oug",
  authDomain: "frisskert-bc9e8.firebaseio.com",
  databaseURL: "https://frisskert-bc9e8-default-rtdb.firebaseio.com",
  projectId: "frisskert-bc9e8",
  storageBucket: "frisskert-bc9e8.appspot.com",
  messagingSenderId: "148948687069",
  appId: "1:148948687069:web:c60335b3ee8fe1c7afa334"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const database = getDatabase(app);
const auth = getAuth();
```

18.ábra: Firebase config

```
bejelent.addListener('click', (e) => {

  var email = document.getElementById('logemail').value;
  var password = document.getElementById('logpsw').value;

  signInWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
      // Signed in
      const user = userCredential.user;

      const dt = new Date();
      update(ref(database, 'Login/' + user.uid), {
        last_login: dt,
      })

      alert('Sikeres bejelentkezés!');
      window.location.href = 'vasarlok.html';
      // ...
    })
    .catch((error) => {
      const errorCode = error.code;
      const errorMessage = error.message;

      alert('errorMessage');
    });
});
```

19.ábra: Bejelentkezés függvény

A fenti ában (19.ábra) a bejelentkezés függvény látható, amely a bejelentkezés gomb megnyomását figyelő EventListenerrel kezdődik. Ezt követőleg átvevődnek az adatok bejelentkezés oldalról és átmentődnek lokális változókba. Ez után meghívódik a signInWithEmailAndPassword firebase függvény, amely az auth, email és password mezőkre hívódik meg és ellenőrzi, hogy létezik-e a felhasználó. Ha a kritériumok teljesítődnek, akkor lementődik az adatbázisba az utolsó bejelentkezés időpontja, az adott felhasználónak, kiíródik

FrissKert - Szoftverrendszerek Tervezése Projekt

a weblapra egy üzenet, “Sikeres bejelentkezés!”, majd átnavigál a vasarlok.html oldalra, ahol a termékek vannak listázva. A végén pedig a hibakezelés jelenik meg.

```
document.getElementById('submitproduct').addEventListener('click', function () {  
  //Beírt adatok átvétele  
  var email = document.getElementById('email').value;  
  
  if (email.trim() === '') {  
    alert("Írjon be egy email címet!");  
    return; // Ne folytassa a feltöltést, ha az email mező üres  
  }  
  
  var tipus = document.getElementById('tipus').value;  
  var nev = document.getElementById('nev').value;  
  var ar = document.getElementById('ar').value;  
  var mennyiseg = document.getElementById('mennyiseg').value;  
  var megye = document.getElementById('megye').value;  
  var telepules = document.getElementById('telepules').value;  
  var utca = document.getElementById('utca').value;  
  var hazszam = document.getElementById('hazszam').value;
```

20.ábra: Termék feltöltés függvény

Az fenti ábrán (20.ábra) a termékek feltöltését végző függvény látható, ami a regisztráció és bejelentkezéshez hasonlóan a gomb megnyomását figyelő EventListener-el kezdődik, majd a form beírt adatainak az átvételé következik, lokális változókba, viszont itt ellenőrzésre kerül, hogy az email cím ne lehessen üres, ezen esetben kilépik a függvényből és kiír a weboldalra egy hibaüzenetet, “Írjon be egy email címet!”.

```
const emailKey = email.replace(/[#$@[\]\/]/g, '_'); // A Firebase kulcsnak megfelelővé alakítás  
const db = getDatabase();  
push(ref(database, 'Product/' + emailKey), {  
  tipus: tipus,  
  nev: nev,  
  ar: ar,  
  mennyiseg: mennyiseg,  
  megye: megye,  
  telepules: telepules,  
  utca: utca,  
  hazszam: hazszam  
});  
  
.then(() => {  
  alert("Termék sikeresen feltöltve!");  
  
  window.location.href = 'vasarlok.html';  
})  
  
.catch((error) => {  
  const errorCode = error.code;  
  const errorMessage = error.message;  
  
  alert(errorMessage);  
});  
});
```

21.ábra: Termék feltöltése függvény2

Folytatásképpen a termék feltöltés függvényről (21.ábra), átalakítódik az email cím speciális karakterek kihagyásával, a replace segítségével, hogy lehessen feltölteni azonosítóként az email címet, így lehetővé téve az adatbázisban, hogy minden egyes email címhez társítva van az összes feltöltött termék. Ez után push metódus segítségével feltöltjük az adatokat az adatbázisba, úgy hogy lekérünk egy referenciát az adatbázisból, így pontos helyre mentődnek el az adatok. Ezek után a hibakezelés látható, ami sikeres feltöltés esetén kidob egy üzenetet a weboldalra, hogy “Termék sikeresen feltöltve!”, majd visszairányít a vasarlok.html oldalra, ahol a kilistázott termékek láthatóak.

```
const dbRef = ref(getDatabase());
const productsRef = child(dbRef, 'Product');
const productsArray = [];
let ar;

onValue(productsRef, (snapshot) => {

    const productsData = snapshot.val();

    if (productsData) {
        // tömböt ürítése
        productsArray.splice(0, productsArray.length);
        const productKeys = Object.keys(productsData);

        Object.values(productsData).forEach(productGroup => {
            Object.values(productGroup).forEach(product => {
                productsArray.push(product);
            });
        });
    }
});
```

22.ábra: Adatok lekérése Firebase-ból

A fenti ábrán (22.ábra) az adatok lekérését az adatbázisból valósítja meg a Firebase Realtime Database SDK (Software Development Kit) szoftvercsomag függvényeit használva. Hivatkozást inicializál a Firebase Realtime Database-re a `getDatabase()` függvény által. Ezt követően létrehoz egy másik hivatkozást, amely a Product csomópontra mutat. Az `onValue()` függvénnyel beállít egy eseménykezelőt, amely meghív egy rekurzív függvényt, amikor változást észlel az adatbázis Product csomópontján. A `snapshot.val()` függvény lekéri az adatomásolatot az adatbázisból. Ellenőrzi, hogy van e adat majd törli a `productArray` tömböt és lekéri a termékek kulcsait az `Object.keys(productsData)` függvénnyel. Végigiterál a termékeken, és azokat a `productArray`be helyezi. Tehát így kéri le az adott függvény az adatbázisból a termékeket majd tároja lokálisan egy tömbben. Fontos kiemelni, hogy az `onValue` függvény asszinkron módon működik. A kód folytatódik anélkül, hogy várna arra, hogy az eseménykezelő befejeződjön. Ezt azért fontos kiemelni, mert ennek következményeképpen az adatok feldolgozása és kezelése is aszinkron módon történik a program futása során.

6. Összegzés

Tehát a FrissKert alkalmazás egy hatékony megoldást kínál a felhasználók számára, mivel egy asztali számítógép vagy laptop segítségével otthonról elérhetőek lesznek a kistermelők által feltöltött friss zöldségek illetve gyümölcsök és ez által időt megkímélve és házi-friss-terméket biztosítva a vásárló számára.

6.1. További fejlesztési lehetőségek

- Aplikációs megvalósítás
- Kedvencek bejelölése
- Több terméket is be lehet tenni a kosárba
- Termékek értékelése
- Kommunikáció eladó és vevő között
- Szűrések több kritérium alapján

7. Bibliográfia/használati tool-ok

- https://www.youtube.com/watch?v=p4Hgzm_oNQ&list=WL&index=8&t=2459s
- <https://www.youtube.com/watch?v=qYER6hAgJik&list=WL&index=7>
- https://lucid.app/documents#/documents?folder_id=recent
- https://www.w3schools.com/js/default.asp?fbclid=IwAR1mBK0W4SZSUQ6WzW5V_Ddn1SsV4H_CpO5OL-IHjCZ6vzVbDA5hHGADHWo
- https://stackoverflow.com/?fbclid=IwAR2DHIBf7rXffvExcjo--o13H5ktNSQLDv-n6J0b-w1tYqZiFHG_0-w--gw
- <https://www.w3schools.com/html/default.asp?fbclid=IwAR0ozki8SCleRXysHoYR7EvSHaKgWAjuhxcX34vgpTmQpftEWLI-y1Q-tlM>
- https://www.w3schools.com/css/default.asp?fbclid=IwAR334XP7G--gotrPJSDVICwIQ0oAqOog4N43aRzywIE_wp0rww1kWR4TZU
- https://docs.google.com/document/d/11_RNV4kJnY_-Fnkn0k9MYLvkG5Lr7qylut5N1QJBsss/edit?fbclid=IwAR08UHAbXhyZsbyXFr3zG1lqiM-mmXAIBNz9L792u0_MqaRKVAr8f5Ce6So#heading=h.j8twx2ybs9ze
- https://firebase.google.com/docs/database/web/read-and-write?fbclid=IwAR2Kle7D09Id5P6OW-68ay1OVGehDXppeuTgG9Ixzcl5QjvDVWFZhB-QnTQ#web-modular-api_4
- <https://zszanto.github.io/teaching/se/?fbclid=IwAR2Il6fhjBYqCemdP62LJoof8IRzTMymMp8w68mBUiHXyV0aLslN2xJliLM>
- https://firebase.google.com/docs/database/web/read-and-write?fbclid=IwAR2Kle7D09Id5P6OW-68ay1OVGehDXppeuTgG9Ixzcl5QjvDVWFZhB-QnTQ#web-modular-api_4
- <https://gyires.inf.unideb.hu/KMITT/c02/index.html>
- https://firebase.google.com/docs/auth/web/start?fbclid=IwAR2gl-CnkeqXKp69k_n1u-3v3vvNP00F6gHFYEOZb1aFOli5jxf3_wuBQOg

