PROJEKTDOKUMENTÁCIÓ

TOLDI RICHÁRD

KOVÁCS FRUZSINA RÉKA

SZEKERES DÁNIEL

SZEGED, 2022

SZEGEDI SZAKKÉPZÉSI CENTRUM
VASVÁRI PÁL GAZDASÁGI ÉS INFORMATIKAI TECHNIKUM

Az 5 0613 12 06 Szoftverfejlesztő és -tesztelő

Fogási napló

Készítette:
Toldi Richárd
Kovács Fruzsina Réka
Szekeres Dániel

Szeged
2022

Tartalomjegyzék

BEVEZETÉS (TOLDI RICHÁRD, SZEKERES DÁNIEL, KOVÁCS FRUZSINA RÉKA)

A fogási napló vezetése napjainkban továbbra is papír alapon történik, amit minden évben le kell adni az azt kiállító horgászegyesületnél, legkésőbb a tárgyévet követő február 28-ig. Ha a kifogott halat nem engedi vissza a horgász, a fogás[1] adatait rögzíteni kell a naplóba. A halőrök feladata, hogy a helyszínen ellenőrizzék, hogy a napló vezetve van-e és a benne szereplő adatok megfelelnek-e a valóságnak. A hazavinni kívánt halakkal kapcsolatban mennyiségi korlát is van, ezért is fontos a fogások naprakész vezetése és a pontos súly megadása.

Ma már mindenkinél található egy okostelefon mobilinternet kapcsolattal, amivel akár a fogás után gyorsan és könnyedén le is lehet jelenteni a fogott és hazavinni kívánt halat. Így az adatbázis percre pontos lehet és a halőrök távolról is meg tudják figyelni, hogy a kifogott hal lejelentésre került-e.

A PROBLÉMA FELVETÉSE

A papíralapú vezetés és annak leadása nem eredményez percre pontos adatbázist, illetve könnyen vissza is lehet élni vele, hiszen azt bármikor kitölthetik utólag is visszamenő időponttal. A halőröknek, hogy ellenőrizni tudják az adott horgász naplóját a helyszínre kell menniük, mert távolról nem tudják megfigyelni a lejelentés tényét.

A papír alapú naplókat évente kell leadni az egyesületnek, így konkrét adatokkal csak az év elteltével fog rendelkezni. Ebből kifolyólag, egy év elteltével tudja kiszabni a bírságot, ha átlépik a súlykorlátot. Statisztikával is egy év után fog rendelkezni, szükséges intézkedéseket is késleltetve tudja meglépni. A horgászok gyakran „elvesztik” a fogási naplójukat és ezáltal a benne lévő adatok is elvesznek. Így nem lehet nyomon követni és betartatni az éves mennyiségi korlátot. A hatóság ezt a problémát még nem tudja megoldani az elvesztett adatokat pótolni az éves lejelentés következtében.

[1] Fogás - A kapás eredménye a fogás és ha nem engedik vissza a halat az lesz a zsákmány.

VÁLASZTOTT TÉMA INDOKLÁSA

Jelenleg papír alapú fogási naplóba történik a megtartani kívánt fogások regisztrálása, amit a halőrök csak a helyszínen tudnak ellenőrizni. Visszaélésekre ad lehetőséget, hogy a horgász bármikor bejegyezheti a szükséges adatokat és az éves lejelentés szintén erre ad lehetőséget.

A digitális verzió megnehezítené a visszaéléseket, mert az ellenőr látná a rendszerben, hogy a megtartani kívánt hal mikor lett regisztrálva a fogási naplóba. Nehezebb lenne megmagyarázni, hogy pont az ellenőrzés előtt fogta ki, rakta haltartóba és még dokumentálta is a fogást. Elhagyni sem lehetne, mert az adatokat másodpercre pontosan egy központi adatbázisban tárolnák.

A leadott naplókat minden évben összesítik a horgászegyesületek, ami jelentős többletmunkát ró rájuk. Ez a folyamat könnyedén automatizálható, ha a fogások jelentése digitálisan történik. Nem kellene külön embereket alkalmazni az adatok rögzítésére, feldolgozására. Nagy adatmennyiség és a rövid idő következtében a hibázás lehetősége is nő, így az adatok pontossága is megkérdőjelezhető.

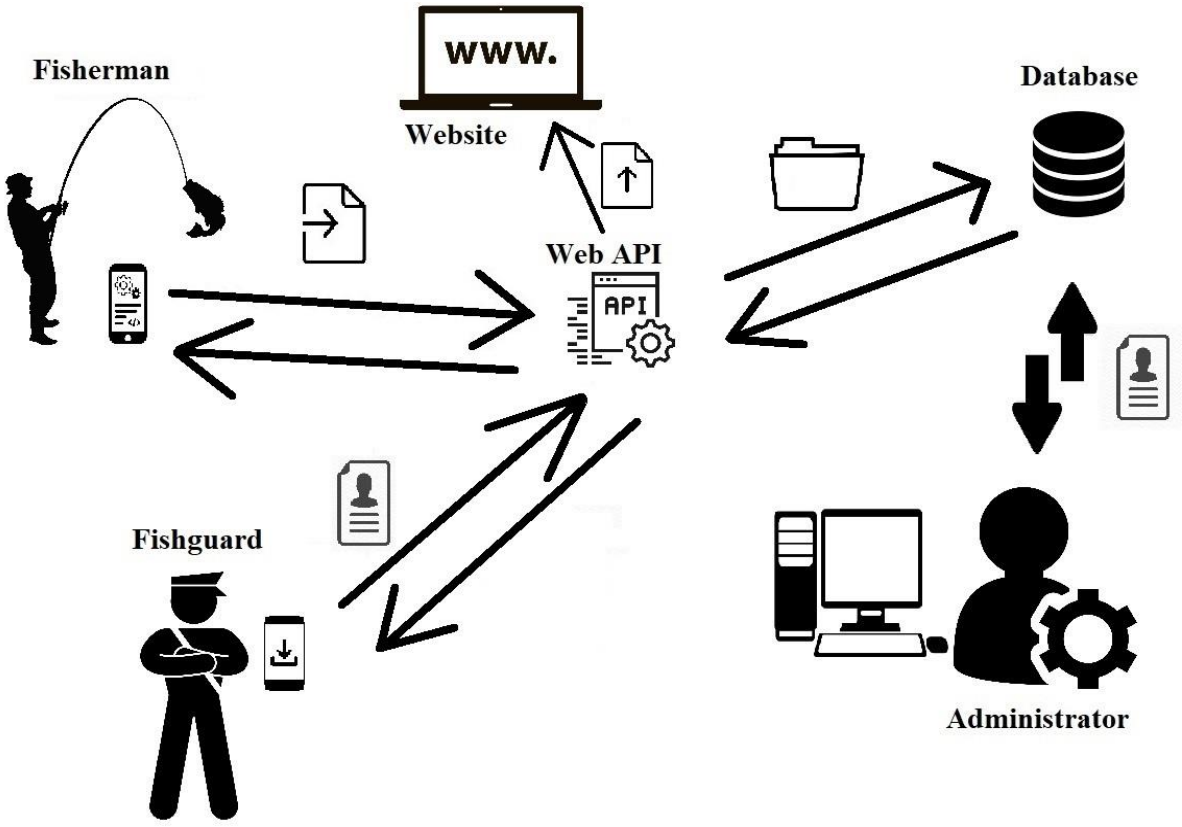
Az adatok feldolgozása, statisztikák elkészítése is automatizálhatóvá válnának és azonnal naprakészek lennének.

1. FEJLESZTŐI DOKUMENTÁCIÓ

1.1 TERVEZÉS (TOLDI RICHÁRD, SZEKERES DÁNIEL, KOVÁCS FRUZSINA RÉKA)

Első lépésben igyekeztünk az elvárásoknak megfelelő témát találni. Feltétel volt, hogy az alkalmazás valódi problémára nyújtson megoldást és életszerű legyen. Az adatbázissal történő adattárolási és kezelési funkciók mellett RESTful[2] architektúrának megfelelő szerver és kliens oldali komponenseket is tartalmaznia kellett.

A témaválasztása után meghatároztuk a környezet szereplőit, hogy tudjuk milyen programot fogunk biztosítani a számukra. A projekt tervezése során minden szereplőnek szerettünk volna egy szoftvert adni a kezébe.



1. ábra A projekt bemutatása

[2] REST - **Re**presentational **S**tate **T**ransfer – reprezentáción alapuló állapotátvitel REST a típusú architektúrák kliensekből és szerverekből állnak. A kliensek kéréseket indítanak a szerverek felé a szerverek ezeket a kéréseket feldolgozzák és a megfelelő választ küldik vissza.

Úgy határoztuk meg, hogy a mobil alkalmazás tökéletes lesz a halőrnek, az asztali alkalmazás, ami közvetlenül hozzáfér az adatbázishoz az adminisztrátornak. A weboldal pedig a horgászok alkalmazása lesz mobiltelefonon vagy akár nagyobb eszközökön is.

Az adatbázis tervezése során fontos volt, az adatbázis nagysága és használhatósága. El akartuk kerülni a felesleges adatok tárolását és a kevésbé fontos táblák meglétét. Egyszerű, de használható adatbázis kialakítása volt a cél, ami képes a szoftverek minden igényét kiszolgálni.

Az adatbázis után a dizájn terveket készítettük el a FIGMA segítségével, felhasználóbarát kezelőfelület létrehozása volt a cél. A FIGMA egy online és ingyenesen használható dizájn tervező alkalmazás, aminek a segítségével könnyedén meg lehet tervezni az alkalmazások kinézetét. Különböző ötlettel álltunk elő és azokból választottuk ki a célnak megfelelőt. Szempont volt a felület jól láthatósága, olvashatósága és egyszerű kezelhetősége.

A dizájn megtalálása után eldöntöttük, hogy mit is kell tudnia az adott szoftvereknek, meghatároztuk a szoftverek főbb funkcióit és hozzá a teszteket.

1.2 ADATBÁZIS (TOLDI RICHÁRD)

1.2.1 Fejlesztői környezet bemutatása

A fejlesztés során az adatbázist hoztuk létre először a XAMPP alkalmazás segítségével. A XAMPP egy szabad és egyben nyílt forrású platformfüggetlen grafikus webalkalmazás, ami magába foglalja az Apache webszerver-t. A program segítségével SQL[3] utasításokkal vagy grafikus módon, menü vezérelt funkciókkal létrehozhatók MySQL[4] relációs adatbázisok.

A szoftver használata megkönnyíti a fejlesztést mivel nem kell szervert használni, hanem a localhoston keresztül történik az adatbázis használata.

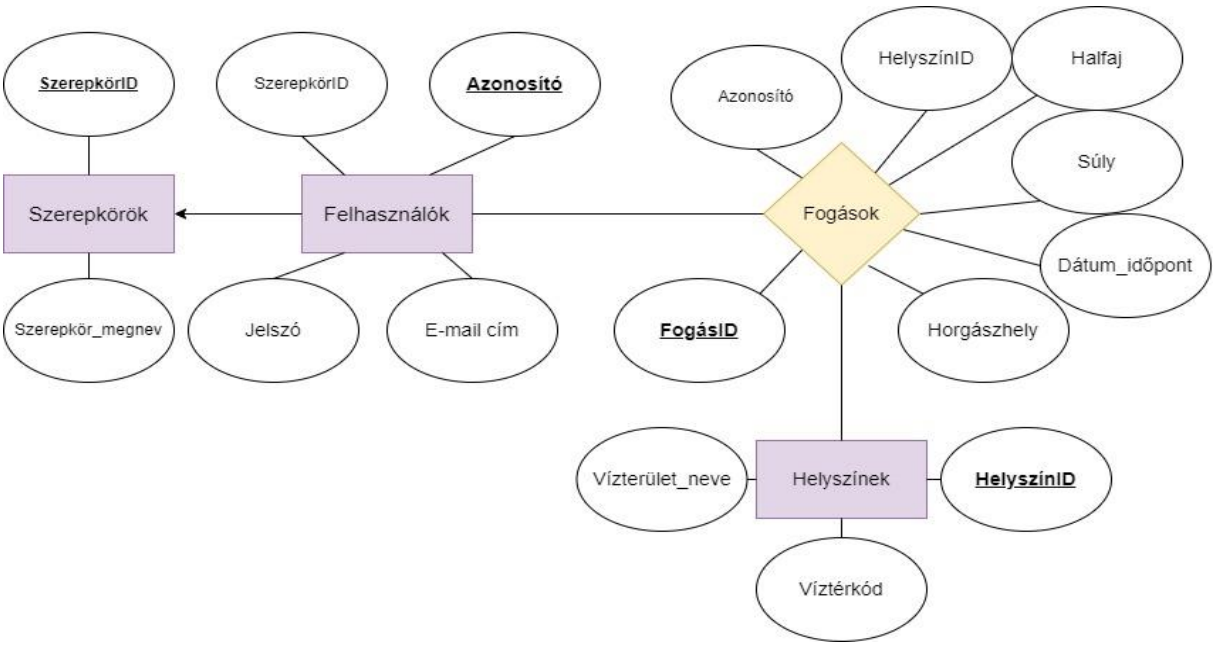
[3] Structured Query Language: strukturált lekérdezőnyelv, a relációs adatbázisok által használt nyelv

[4] MySQL – Az egyik legelterjedtebb adatbázis kezelő, ami különböző platformokon futtatható nyílt forráskódú és egyszerűen használható.

1.2.2 Adatbázis bemutatása

Az adatbázis a horgászok és a fogási adataiknak tárolását teszi lehetővé. Jelenleg tizenhárom fiktív horgász, egy adminisztrátor és egy halőr, valamint tizenhárom Csongrád - Csanád megyei víztér és hatvanhét fiktív fogás található az adatbázisban. A kliens programok a Web API[5]-n keresztül férnek hozzá adatokhoz és így képesek a lekért információk megjelenítésére vagy az újak tárolására. Az asztali alkalmazás közvetlenül hozzáfér az adatbázishoz. A biztonság érdekében a jelszavakat hash[6]-elt formában tároltuk el.

2. ábra Adabázis Bachmann-ábrája



3. ábra Adatbázis E-K diagramja

[5] Application programming interface – alkalmazásprogramozási felület, ami hozzáférést biztosít egy adott szoftver vagy eszköz utasításkészletéhez

[6] Hash – Titkosítási módszer amely egy egyirányú mechanizmus, a hashelt adatokat nem lehet vissza-hashelni.

Az első számú mellékletben láthatóak az adatbázisban szereplő táblák és mezők részletes információi.

1.3 WEB API (TOLDI RICHÁRD)

1.3.1 Fejlesztői környezet bemutatása

A Visual Studio 2019 16.11.3-as verzió integrált fejlesztői környezet „IDE[7]" segítségével készült a szoftver. A Visual Studio ASP.NET Core Web API projekt stílusában dolgoztunk Entity keretrendszerrel, amihez használnunk kellett a .NET csomagkezelőjét a NuGet-eket. A NuGet egy központi csomagtároló, ahonnan könnyen és gyorsan tudunk mások által elkészített osztálykönyvtárakat használni a fejlesztés során és ezzel is meggyorsítani a fejlesztést.

Telepített NuGet-tek:

- Microsoft.EntityFrameworkCore (verzió: 5.0.12)
- Microsoft.EntityFrameworkCore.SqlServer (5.0.12)
- Microsoft.EntityFrameworkCore.Tools (5.0.12)
- Microsoft.VisualStudio.Web.CodeGeneration.Design (5.0.2)
- Pomelo.EntityFrameworkCore.MySql (5.0.3)
- BCrypt.Net-Next (4.0.2)
- Microsoft.AspNetCore.Authentication (2.2.0)
- Microsoft.AspNetCore.Authorization (6.0.0)
- Swashbuckle.AspNetCore (6.2.3)

1.3.2 Program sajátosságai

A mai modern webalkalmazások API-kat tesznek elérhetővé, amelyek segítségével az ügyfeleknek szánt kliens szoftverek az API-n keresztül érhetik el a szükséges információkat, adatokat.

A jó API platformfüggetlen azaz bármelyik kliens képes a meghívására. Ezt protokollok segítségével tudjuk létrehozni, valamint olyan mechanizmusok használatával, amelyek

[7] IDE – Integrated Development Environment – Olyan szoftveralkalmazás, ami átfogó környezetet biztosít a szoftverfejlesztéshez.

lehetővé teszik, hogy a kliens és a webszolgáltatás meg tud egyezni a kicserélődő adatok formátumáról. Az adatok elérhetőségéről a CORS[8] szabályzata gondoskodik, amit külön be kell állítani, hogy kitudja szolgálni az azonos gépen futó alkalmazásokat, a lépés kihagyása esetén támadásként kezeli a kéréseket a szoftver és a jó kéréseket is el fogja utasítani.

A REST alapú webszolgáltatás egyik nagy előnye, hogy a szerver megírható ASP.NET-ben vagy PHP[9]-ban és az ügyfélalkalmazások pedig más nyelven vagy akár más eszközkészlettel is készülhetnek. Az alkalmazásnak nem kell állandó kapcsolatban lennie az adatkiszolgálóval, így a lekérdezések között egy úgy nevezett „nyugalmi" állapot van és ezáltal nem terheli a szervereket vagy a hálózatot.

A kliensek kérésekkel „request"[10]-tel tudnak kommunikálni a szerverrel, ha a kliens a megfelelő kérést elküldi a szervernek az a kért adatot átadja. Az adatok védelmének érdekében bizonyos kérésekhez azonosításra is van szükség, ilyen például a törlés vagy a módosítás kérése.

Az alkalmazás modell rétegében történik az adatbázis tábláinak leképezése, hogy maga a program objektumként tudja leképezni az egyedeket és azokat kezelhetővé teszi a hozzájuk tartozó tulajdonságokkal. Az osztályok mintájára beöltött adatokat már könnyedén tudjuk kezelni és manipulálni. Erre a manipulációra szolgál a DTO osztály, ahol az egyes egyedek tulajdonságait lehet változtatni anélkül, hogy az eredeti osztálystruktúrát átalakítanánk.

A Controller réteg maga a kliens programok által használt végpontokat tartalmazzák. Minden adatbázisban kezelendő táblához kontroller szükséges. Az alap CRUD (create, read, update, delete) készítés, olvasás, felfrissítés és törlés végpontok mellett saját végpontokat is készíteni kellett, hogy a mobil és a weboldal minél több funkcióval bírjon.

Az alapvető végpontokat is a szoftvereknek megfelelően kellett kialakítani, hogy az információkat a számukra legjobb formában kapják meg és keveset kelljen az adatokkal

---

[8] CORS - Cross-Origin Resource Sharing – Mechnaizmus amely lehetővé teszi, hogy más származású, szervertől eltérő eredettel bíró szoftverek számára is engedélyezve legyen az erőforrások betöltése.

[9] PHP - Hypertext Preprocessor – Általános szerveroldali szkriptnyelv dinamikus weboldalak készítésére szolgál.

[10] Kérés – Request - HTTP request a hyper text transfer protocol egy információátviteli protokoll, amely kérés-válasz alapú protokollt biztosít a kliens és a szerver között.

dolgozni. Összesen 29 végpontot hoztunk létre, amelyeket kezelhetnek a kliens programok, valamint 4 darab végpont a memória alapú teszteléshez. A végpontoknál be lett állítva az elérhetőségük, útvonaluk és a visszaadott válaszuk.

Repository réteg biztosítja, hogy az információk csak a megfelelő azonosítás után legyenek elérhetők, ezzel is megakadályozva az illetéktelen hozzáférést vagy az adatok szándékos, véletlen rongálását. Az azonosítás megfelelően működik, de a mobil és a weboldal még nem képes JWT[11]-nel kommunikálni ezért átmenetileg a funkció fel van oldva a végpontoknál.

1.3.3 Tesztelés

A Postman 9.14.1-es verziójával valósult meg a szerver manuális tesztelése. A Postman API-k építésében és tesztelésében nyújt nagy segítséget. A programmal könnyedén kezdeményezhetünk kéréseket a szerver felé, ezt megtehetjük azonosítás nélkül vagy akár azonosítással is. Azonosításból is számtalan lehetőséget kínál a felület a fejlesztők számára. Helyes kérések küldése esetén a program feldolgozza a szervertől kapott választ és azt olvasható formában közli velünk.

A második számú mellékletben látni lehet a tesztelt kérések listáját és a Postman programot. Összesen 29 végpont lett a Postmannel manuálisan letesztelve. Minden teszt során az elvárt eredményt kaptuk. A fejlesztést manuális tesztek folyamatosan végig kísérték, mellettük egység az az „unit” teszteket is alkalmaztunk az egyes metódusok, funkciók vizsgálatához.

A Web API-ban található a HalfajController ami a memória alapú egységtesztekhez készült, más funkcióval nem bír a kontroller. A többi kontrollerhez már adatbázis alapú egység teszteket írtunk, amelyek a végpontokat vizsgálják az adatbázisban szereplő adatok segítségével, a sikeres tesztek érdekében a XAMP kontrol panelt futtatni kell az általunk készített adatbázissal. A memória alapú tesztekhez az adatbázis nem szükséges.

A végpontok vizsgálatához 24 darab egységtesztet hoztunk létre és további 5 darabot a memória alapú teszteléshez is. Minden teszt sikeresen lefutott.

[11] JSON WEB TOKEN – JWT révén tud a szerver és a kliens oldali szoftver biztonságosan kommunikálni, hogy az információk ne kerüljenek illetéktelenekhez.

1.4 WEBOLDAL (TOLDI RICHÁRD, KOVÁCS FRUZSINA RÉKA)

1.4.1 Fejlesztői környezet bemutatása (TOLDI RICHÁRD)

A weboldal a Visual Studio Code 1.64.2-es verziójával készült. Az elkészítéséhez a Vuejs keretrendszert használtuk a 14.16.0-ás verziójú Nodejs-sel. A vizuális ellenőrzésekhez és a manuális tesztekhez Google Chrome böngészőt használtunk.

1.4.2 Weboldal sajátosságai (TOLDI RICHÁRD)

A weboldal fejlesztése a Vuejs keretrendszer harmadik verziójával és hozzáadott könyvtárak segítségével történt. Axios könyvtár teszi lehetővé az adatok elküldését és kérését, Vuex könyvtár teszi lehetővé az „állapot" kezelését. Az azonosításért a Google Firebase[12] felhő alapú adatbázisa és online azonosítása felel és ennek használatát a Firebase könyvtára teszi lehetővé. Router könyvtár az oldalak közti mozgást teszi lehetővé.

Telepítésre került a SweetAlert2 a figyelmeztető felugró ablakokhoz, Bootstrap a reszponzív megjelenéshez és a „modal" valamint a „table" komponensek használatához. A reszponzívitáshoz a Flexbox is hozzájárult, valamint számtalan CSS[13] elem az oldal esztétikus megjelenéséhez. A Date-fns a dátumok formázásához. Valamint online módon importálásra került a Bootstrap és a Google Font Awesome is az ikonok használhatósága érdekében. JQuery is telepítésre került a részletesebb táblázatok használatához.

A weboldal két szerverrel és adatbázissal kommunikál. A regisztrációkor a weboldal lekéri a regisztrálni kívánt email címet az általunk fejlesztett Web API-tól, ha az email cím megtalálható az adatbázisunkban akkor lehet csak regisztrálni. A belépési adatok mentése és kezelése már a Firebase adatbázisával történik. A jelszavak hash-elt formában kerülnek eltárolásra a belépési címekkel.

---

[12] Felhő – Cloud – Online adatbázis, szolgáltatás, ami elérhető bárki számára az interneten keresztül. Az adatok tárolása vagy maga a szolgáltatás folyamata nem a felhasználó gépén történik, hanem egy távoli tőle független eszközön, szerveren zajlik.

[13] Cascading Style Sheets – Webfejlesztéshez használt stílust jelölő nyelv. Segítségével lehet a webes alapú programok megjelenését formázni.

Belépés után a program tárolja a felhasználó állapotát így amíg ki nem jelentkezik beléptetve marad. A sikeres belépést követően az oldal lekéri az email címhez tartozó adatokat és az első kezdő oldalon a felhasználó láthatja a saját adatait. Az oldalon az email címét módosítani is tudja, ha nem adminisztrátor vagy halőr.

Sikeres bejelentkezés és azonosítás után a felhasználó lejelentheti a fogását egy űrlap, azaz egy „form[14]" segítségével. Az oldalon minden kitöltendő mező átesik egy ellenőrzésen a leadás előtt. Amíg a jelenteni kívánt adatok nem felelnek meg a valóságnak a leadást megtiltja az oldal. Az oldalon a kötelező információkat select/option html elemek segítségével kérjük be a felhasználótól, hogy ezzel is csökkentsük a hibázási lehetőséget. A megadott információk ellenőrzésére reguláris[15] kifejezéseket használtunk. Amennyiben sikerül a kötelező információk helyes megadása, a „form" le adhatóvá válik. A leadás előtt az oldal újra kiírja a megadott információkat a felhasználónak és azok elfogadásával kerül elküldésre a Web API számára egy POST kéréssel.

A horgász jogosultsággal bíró felhasználó, képes a saját fogásait és azok adatait megtekinteni táblázatos formában és köztük keresni. Egyéni statisztikáit is megtekintheti a fogásokkal kapcsolatban. Továbbá képes országosan más horgászok által jelentett fogások között keresni, de azonosítási adatokat nem látja. Adatbázisban megtalálható 13 horgászatra kijelölt víztér információit is megtudja nézni és név szerint keresni köztük. A táblázatokért a Bootstrap „table", „card" és a „modal" komponensei voltak segítségünkre.

A halőr belépés után más funkciókkal találkozik, a profil oldal nála is látható, de az email címe nem módosítható. A fogások oldalon már a JQuery javascript könyvtárral készült táblázat látható, ami a horgászok engedély számát is tartalmazza. A táblázatban lapozási, rendezési funkció abc, szám vagy dátum szerint megtalálható. Az adatok szűrésére használható a keresés, ha csak egy személy kapásait szeretné látni a halőr és azokat rendezni.

A JQuery táblázat asztali számítógépen, tableten vagy laptopon kényelmesen és jól használható, de kisebb eszközökön mobiltelefonon már kevésbé kedvező a használata.

---

[14] Form – Weboldalaknál használt űrlap, amivel adatokat lehet bekérni a felhasználóktól.

[15] Reguláris kifejezések (Regex) – Egy karakterlánc minta, amivel szöveg keresését, cseréjét és ellenőrzését tudjuk könnyedén elvégezni.

Ezért a halőr kapott egy felhasználók nevű oldalt is, ahol a felhasználók kerülnek kilistázásra engedélyük számával és email címükkel. A megjelenítésükért Bootstrap kártya komponensek felelnek. A kártyák között lehet szűrni, akár egy engedélyre is rá lehet keresni. A kártyákra kattintva a Bootstrap „modal" segítségével jelennek meg időrendi sorrendben az adott horgász fogásai és azok adatai.

Az adminisztrátor a harmadik szerepkör, amit az oldal meg tud különböztetni. Belépés után az adminisztrátor is először a profil oldallal találkozik, ahol csak a felhasználói szereppel bírók tudják az email címüket megváltoztatni. Hozzáfér a JQuery táblázathoz, ahol a fogások kerülnek kilistázásra a horgászok engedélyszámával, amivel a horgászokat azonosítani lehet. A felhasználók menüpontban viszont már eltér a halőrétől, itt lehet az adatbázisban tárolt horgászokat kezelni. Az oldal Bootstrap táblázattal kilistázza az összes engedéllyel rendelkező horgászt, lehet köztük keresni és szűkíteni a lista elemeit. A lista elemeire kattintva a kis ikonokkal lehet műveleteket végrehajtani. Lehet törölni a kijelölt felhasználót vagy az adatait módosítani kivéve az engedély számát. Az engedély szám nem változik általában, valamint az az elsődleges kulcs a felhasználók táblánál. Módosítás után a táblázatot frissíteni kell a jobb sarokban lévő nyilas ikonnal. Új felhasználót is tud regisztrálni az adatbázisba és a frissítés után minden módosítás vagy új felhasználó megjelenik.

Az új felhasználók mentése és a meglévők adatainak módosítása Bootstrap „modal" segítségével történik. A mezőkön egyszerűbb reguláris kifejezésekkel készült validáció van. Amíg a megadott adatok nem megfelelőek a leadási gombok le vannak tiltva, a validáció még pontosításra szorul.

Az oldalon számtalan figyelmeztetés „alert" található, ha az oldalt használó éppen tiltott dolgot vagy hibás műveletet szeretne végrehajtani. Információs felugró ablakok is megtalálhatóak az oldal különböző funkcióinál, hogy ezzel is terelje a használóját.

A menürendszer reszponzív hamburger menüvel lett ellátva, hogy kisebb kijelzőkön is jól használható legyen. Az oldal egészen 320 pixel szélességig jól használható. 320 pixel szélesség alatt a jelenleg használt eszközök 0.1%-a van a statisztikák alapján, így az eszközök 99.9% át ki tudjuk szolgálni.

1.4.3 Weboldal tesztelése (KOVÁCS FRUZSINA RÉKA)

Az alkalmazás során manuális tesztelés alá került valamennyi funkció. Többek között a belépés, hogy helytelen adatok esetén a figyelmeztetések megfelelőek-e és helyes adatok esetén a belépés, azonosítás megtörténik-e. A tesztek során minden az elvártak szerint történt. A fejlesztést manuális tesztek végigkísérték, valamint a fejlesztő aktívan használta a Google Chrome böngészője adta fejlesztői felületet és ott is a konzol, hálózat és az alkalmazás menüpontjait, hogy lássa az adatok megérkezését, helyes feldolgozását vagy a helyi tárolóban „localstorage" történő eltárolását.

A kód tesztelésére Jest keretrendszert használtunk. A Jest a Vuejs, React, Angular és egyéb keretrendszerekben teszi lehetővé a programkódok tesztelését. Segítségével könnyedén írhatunk unit teszteket a kódunkra. Összesen 9 darab egységtesztet készítettünk el. Leginkább a horgász szerepkörrel bíró felület került vizsgálat alá, hogy azon a felületen minden elem, amit előre meghatároztunk megtalálható-e. Fontos volt tesztelni a POST kérés elküldést, a helytelen adatok szűrését az alkalmazásban. Az ellenőrzések tesztelésénél derültek ki apróbb hibák, de azokat könnyen lehetett orvosolni.

1.5 ASZTALI ALKALMAZÁS (SZEKERES DÁNIEL)

1.5.1 Fejlesztői környezet bemutatása

A program a Microsoft Visual Studio Community 2022 (64-bit) 17.0.5-ös verziójú integrált fejlesztői környezet segítségével készült C# nyelven. Ezen belül WPF App (.Net Framework) projekt formájában dolgoztunk, Entity Framework-kel, amihez itt is használnunk kellett a .NET csomagkezelőjét a NuGet-eket. Az alkalmazott architektúra a MVVM[16] lett. Az adatbázissal kapcsolatos műveleteket a XAMPP Control Panel 3.2.4-es verziójával végeztük. Objektum orientált programozási alapelveket használva készült a program.

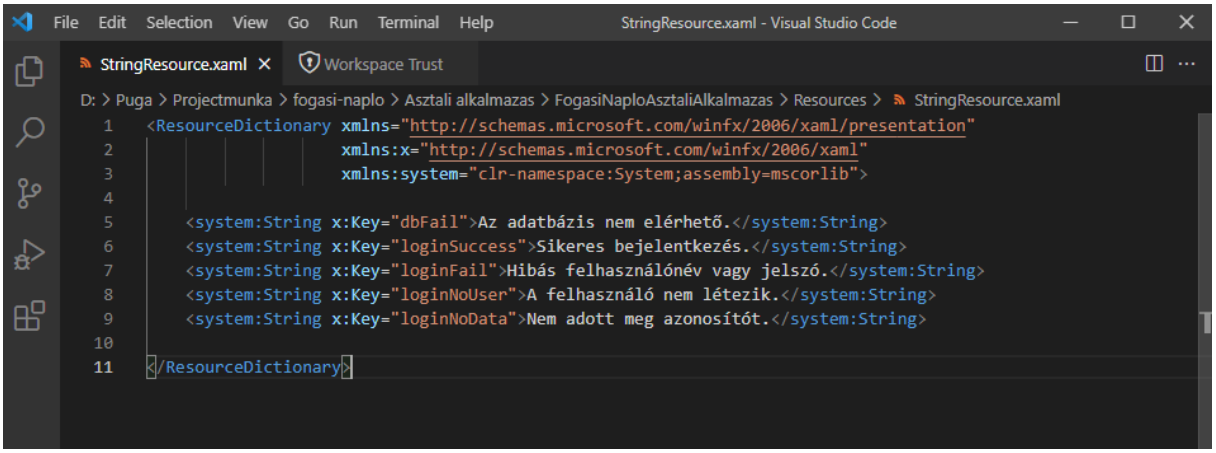
[16] Model-View-ViewModel – Fejlesztési architektúra, ahol a modell réteg képezi le az adatbázis tábláit, view réteg felel a megjelenítésükért, amivel a felhasználó interakcióba lép és a viewmodell réteg metódusokon, tulajdonságokon keresztül jeleníti meg a modell adatokat és az üzleti logikát adatkötésen keresztül.

Telepített NuGet-ek:

- Microsoft.EntityFrameworkCore (verzió: 5.0.12)
- Microsoft.EntityFrameworkCore.SqlServer (5.0.12)
- Microsoft.EntityFrameworkCore.Tools (5.0.12)
- Microsoft.VisualStudio.Web.CodeGeneration.Design (5.0.2)
- Pomelo.EntityFrameworkCore.MySql (5.0.3)
- BCrypt.Net-Next (4.0.2)
- Microsoft.AspNetCore.Authentication (2.2.0)
- Microsoft.AspNetCore.Authorization (6.0.0)

1.5.2 Program sajátosságai

Repository mintákat használtunk a program kódjának megírásánál. Ezek réteget képeznek az adathozzáférés és az üzleti réteg között. Az összes repository osztály egy helyen van kezelve egy generikus osztály használatával. Így csak egyetlen generikus repository-t kellett létrehozni, amely a teljes adatkezelésért felel. Ennek a mintának a fő előnye a kód újrafelhasználhatósága. Ezt az osztályt csak egyszer kellett megírni és csak örököltetni.



4. ábra Asztali alkalmazás rendszerüzenetei

A StringResource-ban megadtuk a különböző rendszerüzeneteket, amiket a bejelentkezésnél szerettünk volna értesítésként kiíratni. Így ezeket csak itt kellett megírni, és utána csak hivatkozni kellett rájuk az itt megadott „key” -ek alapján.

Az asztali alkalmazás a localhoston futó adatbázissal kommunikál. A program reszponzív, az összes tesztelt ablakméretben megfelelően jelentek meg az adatok.

A bejelentkező oldalon található egy, a beírt jelszót megjelenítő/elrejtő gomb. Ezt a WPF app sajátosságai miatt úgy tudtuk működésre bírni, hogy két beviteli mező készült. Az egyik (passwordbox) pontokkal helyettesíti a beírt jelszót a biztonság érdekében, a másik (textbox) pedig a valós beírt karaktereket jeleníti meg. A gombra kattintva vált a program a két mező között az ikonnal együtt. Amíg nem ad meg azonosítót és jelszót, addig a bejelentkezés gomb ikonja utal arra, hogy még nem fog tudni belépni. Kattintható a gomb, de hibaüzenet lesz látható.

A belépés után látható az alkalmazás lista nézete, ahol felsorolásra kerül az összes bejelentett fogás. Az egy lapon megjelenő bejegyzések száma módosítható 25-re, 50-re, vagy 100-ra. Az oszlopok címsorára kattintva rendezi az adott oszlop tartalmának a sorait abc szerinti, időrendi vagy a számok esetén csökkenő/növekvő sorrendben. A fogások adatainak módosítására létrehozott felület is itt található. A kijelölt rekord törölhető vagy az adatai módosíthatóak, valamint új rekord is itt hozható létre. Minden az alkalmazásban mentett változtatás eltárolásra kerül a localhost-on elérhető adatbázisban, amit a weboldal, és a mobil applikáció is elér a Web API-n keresztül. A keresési funkció, ahol a kereső mezőbe beírt karakterekre szűrést alkalmazhatunk. Ekkor csak azok a fogások jelennek meg amelyek tartalmazzák az adott karakterláncot.

Ezen a felületen található egy, az adatbázisban eltárolt felhasználók adatainak megjelenítésére szolgáló gomb is. Erre kattintva egy másik ablak nyílik meg, az előző pedig bezáródik. A program figyel arra is, hogy ha az aktuális ablak teljes képernyős módban volt, akkor az új ablak is teljes képernyőn jelenjen meg.

Az alkalmazás felületein, ahol lehetséges volt, ikonokat használtunk, modernebbé téve a megjelenést. Lekerekített sarkú ablakokat használtunk az alap helyett. Így az eredeti gombok helyett sajátot kellett készíteni. Mind kinézetben, mind funkcióban meg kellett írni ezeket.

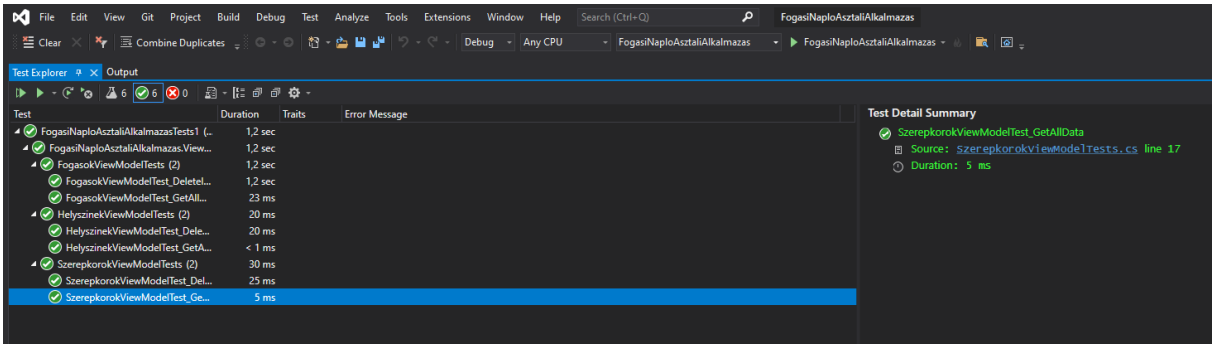
```
private void SulykorTat_PreviewTextInput(object sender, TextCompositi
{
    var regex = new Regex("^([1-9][0-9]?[0-9]?[.]?[5]?|1000)$");
    e.Handled = !regex.IsMatch((sender as TextBox).Text.Insert((sende
}
```

5. ábra Asztali alkalmazás hal súlyának ellenőrzése

A fogások szerkesztésénél a súly beviteli mezőnél szűrni szerettük volna a bevitt adatot, hogy csak megfelelő formátumot lehessen megadni, erre egy saját regex mintát hoztunk létre. Az elvárás, hogy 0.5 kg-tól kezdődően 0.5 kg-os lépésekkel lehessen megadni és csak egy pontot tartalmazhasson a szám. Egyetlen hibája, hogy a legkisebb bevihető súly 1-től indul nem pedig 0.5-től.

1.5.3 Tesztelés

A tesztelés végigkísérte a fejlesztés teljes folyamatát. Minden elkészült funkciót egyből alávetettük az ellenőrzésnek. Mind manuális, mind automata teszteket végeztünk. Ha hibára derült fény, akkor ennek kijavítása volt az elsődleges cél, még mielőtt tovább fejlesztettük volna a programot.



6. ábra Asztali alkalmazás sikeres egységtesztek

A Visual Studio beépített tesztkörnyezetével készítettük el az egységteszteket, ahol a ViewModel osztályokat vetettük teszt alá. Összesen 6 darab unit teszt készült el az asztali alkalmazáshoz, ezek mindegyike hiba nélkül fut le.

A manuális tesztelés során pedig figyelmet fordítottunk a jól olvashatóságra, és helyes megjelenítésre is. Számos esetben a manuális tesztelés során vettünk észre olyan hibát, amit automata teszteléssel nem lehetett kiszűrni. A fejlesztés során megtapasztaltuk, hogy mindkét módszerre szükség van ahhoz, hogy lehetőleg az összes felmerülő hibát kiszűrjük.

1.6 MOBIL ALKALMAZÁS (KOVÁCS FRUZSINA RÉKA)

1.6.1 Fejlesztői környezet bemutatása

A mobil applikációnk Ionic keretrendszerben íródott React alkalmazás. Ennek egyik nagy előnye, hogy kész komponenseket is fel tudtunk használni. A felhasznált komponensek rugalmasan változtathatóak és bővíthetőek. A fejlesztés TypeScript[17] nyelven történt, ami segített elkerülni a JavaScript[18] atipusosságának hátrányait. A VSC[19] kódfejlesztő programmal készült a mobil alkalmazás, hála a platformfüggetlenségének és a több nyelvű támogatásának, valamint a hozzá letölthető kiterjesztéseknek, sokat segítettek munkánk során. Az alkalmazással kapcsolatos hibák, „bugok" feltárásában a VSC, Ionic-React compiler[20] és a Google Chrome Developer Tools [21]volt segítségünkre.

A fejlesztés során segítségünkre volt az online elérhető Ionic dokumentáció, valamint a natív react dokumentációja is.

1.6.2 Program sajátosságai

RESTful alapú architektúrát használtunk fel a projektmunka során, amely esetünkben egy szerverből és kettő kliensből áll. A mobil kliens egy aszinkron[22] kérést küld a http://localhost:5000/api/ cím felé, amit a RESTful alkalmazás keretei között a Web API (szerver) dolgoz fel, és a megfelelő választ küldi vissza. A mobil alkalmazás így kapja meg a felhasználók, fogások és a helyszínek adatait, amiket a kliens a horgászok modelljében kialakított struktúrában dolgoz fel és az objektumokat listákban jelenít meg a halőr számára.

---

[17] TypeScript – Erősen típusos programozási nyelv, ami a javascript nyelve épül

[18] JavaScript – Webfejlesztésnél használt atípusos programozási nyelv

[19] Visual Studio Code – Forráskód szerkesztő, leginkább webfejlesztésre használt

[20] Compiler – Szoftver, ami a programozási kódot egyszerűbb gépi kódra fordítja

[21] Google Chrome Developer Tools – Böngészőbe megtalálható segédeszköz fejlesztők számára

[22] Aszinkron - A program nem várja meg a kérés végrehajtását, hanem fut tovább. A válaszra történő várakozás nem akasztja meg a program futását, a várakozás alatt a program egyéb funkciói hajtódnak végre.

A lista betöltése kapott egy késleltetést, amelyet React Hooks használatával tettünk működőképessé. A „Horgok” a React 16.8-as verziójától elérhetők, lehetővé teszik az állapotok és más beépített React horgok használatát osztályok írása nélkül, leegyszerűsítve a programozók munkáját. A UseEffect hook hatás biztosítja a késleltetést, itt azt is meg tudtuk adni pontosan hány másodperces legyen a késleltetés. Mindez esztétikusabb működést biztosít a program számára, nem azonnal jelennek meg az adatok a képernyőn. A töltés vizualizálásához a React loader spinner-t használtuk. Ezt a terminálba beírt npm paranccsal telepítettük, majd importálva tudtuk felhasználni. A React keretrendszerhez sok már kész könyvtárat fel lehet használni, ami a fejlesztést gyorsítja és egyszerűsíti, ezeket ki is használtuk alkalmazásunk fejlesztése során. A töltés után indul el az aszinkron (GET)kérés a szerver felé, hogy megkapja a program a kért adatokat. A RESTful server dolgozza fel a kliens által küldött kérést, majd arra válaszol. A mobil alkalmazásban modellezzük az objektumokat a horgaszok.ts fájlban, ezáltal azokat vizuálisan is meg tudjuk jeleníteni a képernyőn. A kilistázandó adatok struktúráját egy külön fájlban (komponensben) írtuk meg, és a „map()” beépített függvénnyel jártuk be az adat-tömböt. A késleltetés után az Ionic keretrendszer által biztosított lista elem jeleníti meg a lekért adatokat a komponensben meghatározott módon. Ez lehetővé teszi a későbbiekben a komponens használatát anélkül, hogy újra meg kelljen írni az egész kódot. Egyszerűen tudjuk implementálni másik alkalmazásba is, de alkalmazásunkon belül másik oldalon is fel tudjuk használni a kész komponenst.

A lista elemeit kattintható útvonalakká kellett konvertálni, hogy ha a halőr rákattint az egyik felhasználóra megkapja a felhasználó fogásait. A felhasználóra klikkelés esetén elindul egy újabb Ionic lista komponens késleltetéssel, ami már az adott felhasználó fogásait fogja kilistázni egy másodperc után. A komponens aszinkron kéréssel kapja meg az adatokat a szervertől GET metódussal.

A fogások listát is a felhasználók lista mintájára át kellett alakítani, hogy a halőr láthassa egy-egy fogás részletes adatait. A részletes adatok betöltése szintén kapott késleltetést és az előre meghatározott aszinkron kérést használja, hogy megkapja a szervertől az adatokat.

A program kapott egy reszponzív hamburger menüt is, ahol megtalálható minden oldal: a horgászok, keresés, profil, fogás jelentése és a kijelentkezés funkció is egyaránt. Mindezt egy külön komponensben írtuk meg, így mindegyik oldalon egyszerűen implementálható. Minden elemnek külön van kattintható útvonala, amely az adott oldalra irányít át.

A menüt az Ionic ikonok segítségével tettük színesebbé. Ezek is importálás után alkalmazhatóak a komponensekben.

A bejelentkezési adatokat külön változókban tároljuk el és az input mezőn változtatásra inicializáljuk az értéküket. Mindezt a webAPI szervernek küldi el a kliens. Ha az adatok helyesek és megtalálhatóak az adatbázisban, akkor a szerver OK(http 200 státusz kód) választ és a felhasználó azonosítóját küldi vissza, hogy azt ellenőrizhesse. A bejelentkezési adatokat a localstorage-ban tároljuk el, így ezek a profil oldalon keresztül elérhetőek és módosítást is végre lehet hajtani rajtuk PUT metódussal, így mentve az adatbázisba azokat.

POST kérést is képes küldeni a kliens a szerver felé, ez a fogás regisztrálása oldalon érhető el. Itt minden adatot az input mezőkbe írt értékkel inicializálunk. Minden kötelező input kitöltése esetén tudjuk csak elküldeni a szerveren keresztül az adatbázisba az adatokat. Validálás is történik, ha nem megfelelő az azonosító, nem találja a http lekérdezés az adott azonosítójú horgászt a tömbben, akkor nem tudja elküldeni az adatokat az adatbázisba az alkalmazás. Erre egy figyelmeztető üzenet is tájékoztatja a felhasználót.

Ha a helyszín mező üres, akkor a „Nincs adat” -tal küldi el adatbázisba, így az értéke nem null lesz. Erre egy külön függvényt írtunk a komponensben.

A szoftvert használó képes a horgász azonosítójára is rákeresni így nem kell a listát átnéznie vagy abból kikeresnie. A keresés funkció képes az összes felhasználó kapásainak megjelenítésére, vagy csak a keresett horgász kapásainak kilistázására is. Így gyorsabban ellenőrizhető egy-egy horgász kapásainak nyilvántartása. Betűket nem tud írni az input mezőbe a felhasználó, csak számokat. Mivel az azonosító integer típusú adat kell legyen, így volt a legcélszerűbb kiküszöbölni az alkalmazás esetleges összeomlását.

Az Ionic keretrendszernek hála a vele fejlesztett program képes Android, IOS és webes reszponzív alkalmazásként is funkcionálni. Nagyfokú rugalmasságot biztosít a keretrendszer a fejlesztők számára és felgyorsítja a programozást a kész komponensek segítségével, és a React Hooks használatával.

1.6.3 Tesztelés

Tesztelés folyamatosan végig kísérte a fejlesztést. Minden funkció és aszinkron kérés tesztelésen esett át amint egy-egy modul kódolása befejeződött. Könnyebb volt így kis részletekben megtalálni a hibákat, mint a program elkészültével keresni, hogy hol akad el az

adatfolyam. A lekérések tesztelésében a Google Chrome Developer Tools segített a böngészőben. Lekéréseknél egyszerűen a konzolra ki lehetett íratni a kapott objektumokat, ha volt adat és helyes volt, akkor a lekérés működött. A funkciók tesztelése is nagy hangsúlyt kapott, mivel azok felelnek az adatok vizualizálásáért. Minden funkció különféle tesztelés alá esett, hogy megtaláljuk a hibákat és később javítani tudjuk őket. Többek között az elemek, ikonok meglétét egységtesztekkel teszteltük az adott oldalon, amelyeknek jelen kell lenniük, hogy egy-egy funkció megfelelően működhessen. Ezekre cím HTML tulajdonság alapján kerestünk rá. Az adatok megfelelő betöltése is teszt alá esett: a horgászok tömb első elemének az e-mail címét kerestük, és ha ez megtalálható a homepage oldalon, akkor feltételezhetjük, hogy a többi elem is megfelelően betöltődik. Tehát a fetch működik, a szerverrel és adatbázissal is megfelelő a kapcsolata a kliensnek. További tesztekben a tömb többi elemének a betöltődését is tesztelés alá vetettük. Összesen 5 tesztet végeztünk a React Testing könyvtárral ily módon. Ehhez semmit sem kellett telepíteni, a React tartalmazza ezt a könyvtárat már a projekt létrehozásakor is. Importálással érhető el alkalmazáson belül a tesztelési könyvtár. Igyekeztünk mindent tesztelni, hogy az alkalmazás megbízható legyen minden felhasználó számára.

2. FELHASZNÁLÓI DOKUMENTÁCIÓ

2.1. MOBIL KLIENS (KOVÁCS FRUZSINA RÉKA)

2.1.1. A program célja

A mobil alkalmazás célja elsősorban, hogy a halőr megtekinthesse és ellenőrizni tudja a horgász fogásait, hogy azok naprakészen jelentve vannak-e. Minden adat a központi adatbázisban van tárolva ezáltal a horgásznak megnehezíti a csalás lehetőségét. Az ellenőrzést egyszerűsíti, papírmunkát csökkenti, és kizárja a fogási napló elvesztésének lehetőségét.

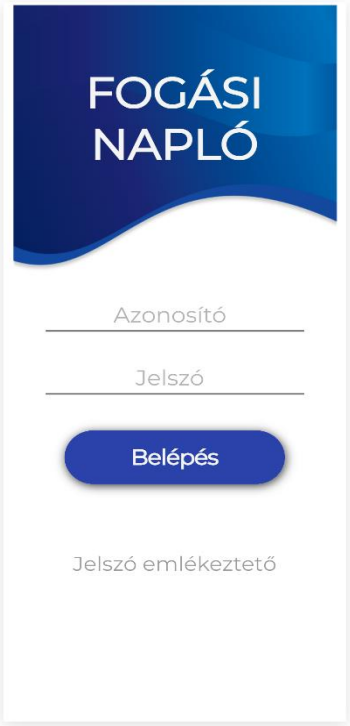
2.1.2. A program használatának bemutatása

Az alkalmazás működéséhez szükséges egy külső adatbáziskezelő használata a beimportált adatbázissal, amit az adatbázis mappában lehet megtalálni. Mi a XAMPP Control Panelt használtuk a fejlesztés során. Az Apache és MySQL szervereknek futnia kell. Elengedhetetlen a Nodejs legújabb verziójának megléte is a számítógépen és az Ionic React keretrendszerre is szükség van a futtatáshoz.

A program fájlrendszerének letöltése után a program mappájába a node modulok telepítése szükséges, ezzel telepítjük a futtatáshoz szükséges csomagokat. A mappából megnyitott PowerShell, vagy parancssori ablakba kiadott npm install segítségével a csomagok könnyen telepíthetőek.

A Web API szerver futtatása is nélkülözhetetlen, amihez a Visual Studio telepítése szükséges a számítógépen. A szerver fogja az adatbázissal összekapcsolni a klienst, így futása a háttérben nélkülözhetetlen a megfelelő működés érdekében. A PowerShell ablakban vagy a Visual Studio Code termináljában az ionic serve parancs segítségével lehet futtatni a programot a localhost:8080 -as portján, majd automatikusan bedobja a Google Chrome böngészőben az alkalmazásunk bejelentkezési felületét.

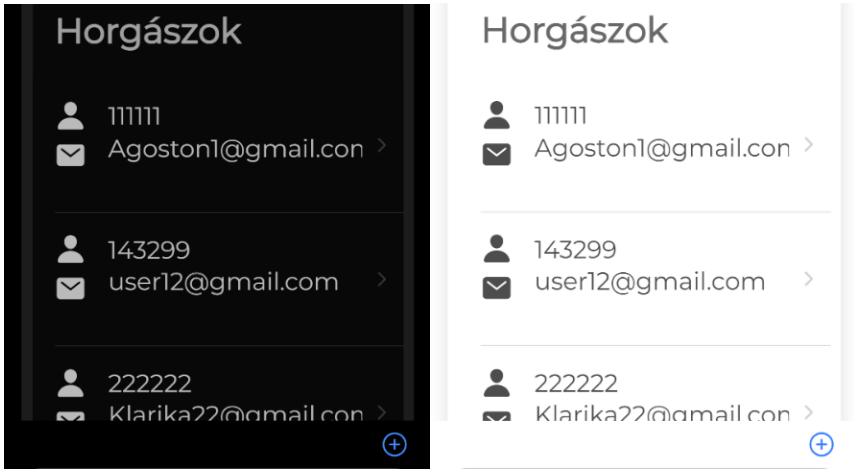
A halőr a bejelentkezés oldalon tud belépni az applikációba. Ez az oldal jelenik meg alapértelmezettként. A halőrnek külön azonosítóval és jelszóval kell rendelkeznie, hogy megtekinthesse a horgász adatokat. Ezek az adatbázisban vannak eltárolva, ahonnan adatokat kér le a kliens a szerveren keresztül. Ha az azonosító vagy jelszó nem megfelelő vagy üres valamelyik, akár mindkét mező, akkor az alkalmazás erre figyelmeztet és nem engedélyezi a belépést. Ha a belépési adatok helyesek, akkor a belépés gombra nyomva fog a kliens a főoldalra átirányítani. A minél egyszerűbb dizájn volt szem előtt tartva, mivel mobiltelefonokon a képernyő kisebb, korlátozottabb, így a letisztult, de stílusos, könnyen értelmezhető struktúra volt a cél.



7. ábra Bejelentkezés felület

Belépési adatok:

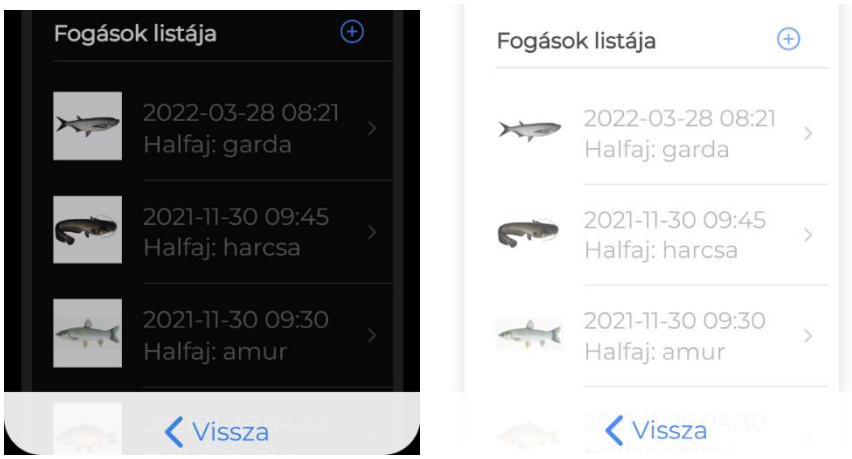
- Halőr: 20000 / halor1987



8. ábra Horgászok listája

Betöltés után a horgászok listája, a mobil applikáció főoldala látható. Minden regisztrált horgász megtalálható a listában azonosítószámmal és email címmel ellátva. A lista gördíthető, reszponzív. A bal fenti sarokban megtalálható a hamburger menü ikonja, amiben minden oldal és funkció elérhető, alatta pedig egy csúszka, aminek funkciója a téma változtatása.

Az alkalmazás kapott egy sötét és egy világos témát, hogy a felhasználó tudjon váltani a kettő között ízlése szerint.

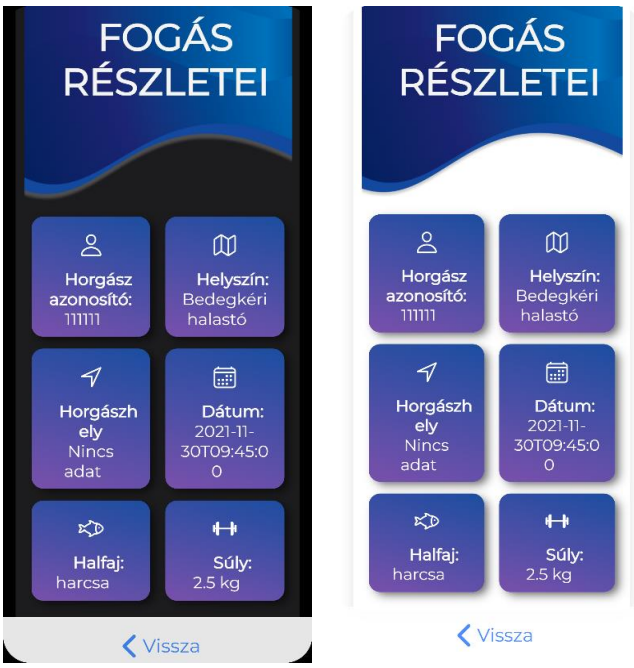


9. ábra Horgász fogásainak listája

Ha valamelyik kattintható lista elemet kiválasztja a halőr, a felhasználó regisztrált fogásainak listáját látja. A fogások listájában a dátum és a fogott halfaj látható.

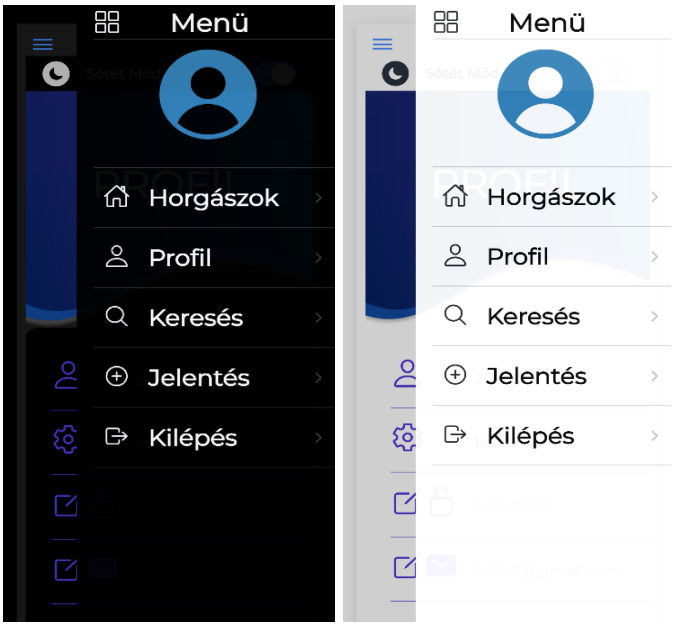
Amilyen halat fogott a horgász, olyan halfajnak a képe jelenik meg bal oldalt a listaelemben. Előfordulhat, hogy egy regisztrált felhasználónak még nincsenek fogásai. Ilyenkor az oldalon betöltés után egy vissza gomb jelenik meg, amivel az előző oldalra tud lépni a halőr, hogy további felhasználók fogásait tekinthesse meg. A Fogások listája cím mellett egy plusz ikon látható, amire kattintva a fogás jelentése felületre tud eljutni a felhasználó.

Ha a fogások listájában választ ki egy lista elemet, a mobil kliens új oldalra irányít át és kiírja a kiválasztott fogás adatait. Itt nem listaként jelennek meg az adatok, hanem kártyákon kiíratva. Megjelennek a részletes adatok: a horgász azonosítója, helyszín, horgászhely, dátum, a fogott hal faja és a hal súlya is kg-ban. Ha horgászhely nincs megadva az alkalmazás megjeleníti, hogy nincs adat.



10. ábra Adott fogás részletei

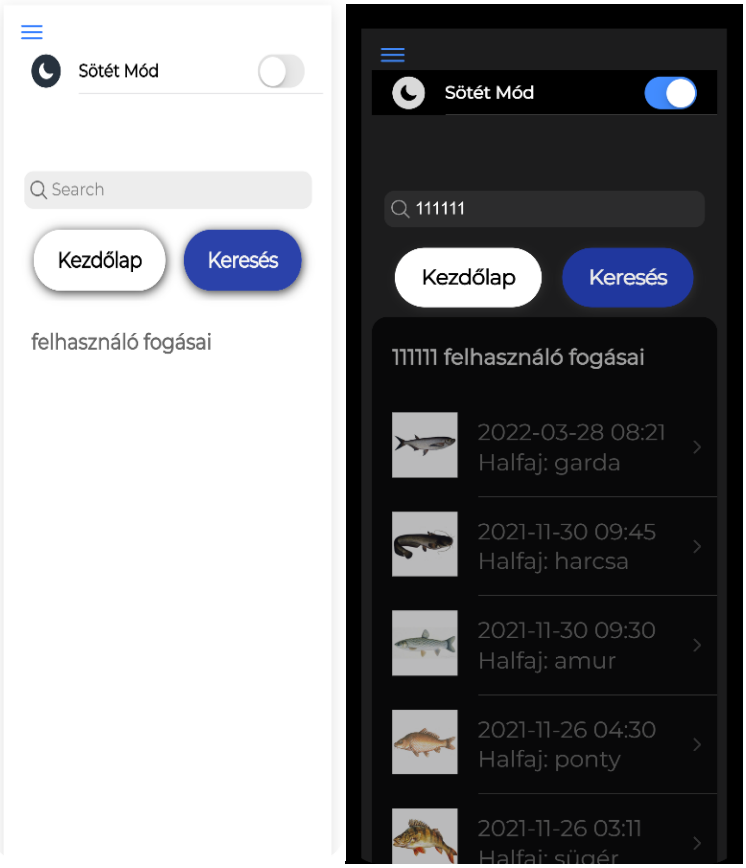
Mobil nézeten a menün keresztül elérhető az összes oldal, a kijelentkezés menüpont és a keresés funkció is. A menü reszponzív, nagy méretű kijelzőn nem hamburger-menüként jelenik meg. Ikonokat használtunk itt is a vizuális megjelenítésben. Ha a későbbi fejlesztésekben az adatbázisba bekerül egy felhasználói-profilkép, akkor az itt a menü tetején is meg fog jelenni.



11. ábra Hamburger-Menü mobil nézeten

Az oldalon a halőr a horgász azonosítóját a keresés mezőbe beírva tud adatokat lekérdezni. Ha megfelelő azonosítót vitt be, az adott azonosítóval rendelkező horgász fogásairól kap listát. Ez nagyobb mennyiségű felhasználó esetén lehet hasznos segítség a halőr számára, hogy gyorsabban tudjon navigálni az applikációban. Ha nem jó azonosítót ad meg, az alkalmazás erre figyelmeztet. A listaelemekre kattintva kapja meg a kiválasztott fogás részleteit.

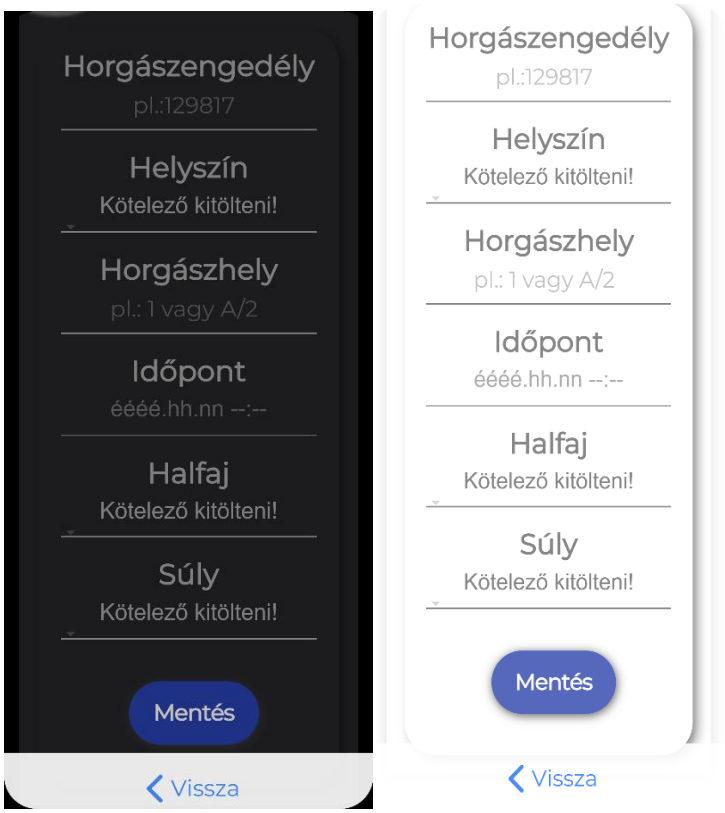
Egy kezdőlap gomb is van az oldalon, ami visszairányít a horgászok listájára.



12. ábra Keresés funkció

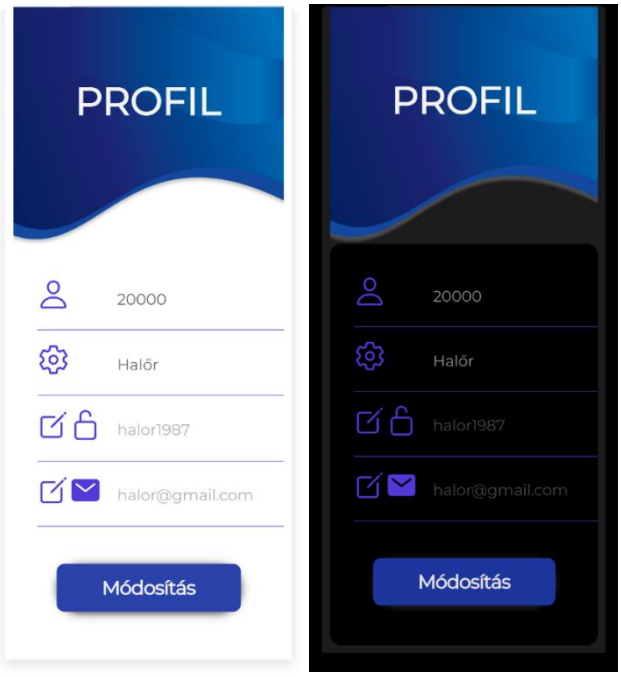
A mobil kliens kapott egy fogás jelentése felületet is, biztosítva a halőrnek, hogy a nem jelentett fogásokat a horgász helyett utólag jelentse. Ez egy űrlap, amin vannak kötelezően kitöltendő és opcionális mezők is. A halőrnek a horgászhely kivételével minden mezőt ki kell töltenie ahhoz, hogy elküldhesse. Az alkalmazás helyesen kitöltött űrlap esetén az adatbázisba elküldi az adatokat. A mentés gomb mindaddig inaktív, amíg minden kötelező mező nincs kitöltve.

Vissza gombbal itt is az előző oldalra tud lépni a felhasználó, elküldött űrlap esetén pedig visszairányít az alkalmazás a főoldalra.



13. ábra Fogás Jelentése oldal

A profil felület az első sorban a felhasználói adatok megjelenítésére készült. Az azonosító, szerepkör, jelszó és E-mail cím található ikonokkal ellátva az oldalon. A jelszó és email módosítható elem, sikeres módosítás az adatbázisba feltölti a módosításokat az alkalmazás. Ha az email és jelszó mező üres, és a felhasználó a módosítás gombra nyom a kliens figyelmeztet, hogy minden módosítható mezőt ki kell tölteni.



14. ábra Profil oldal

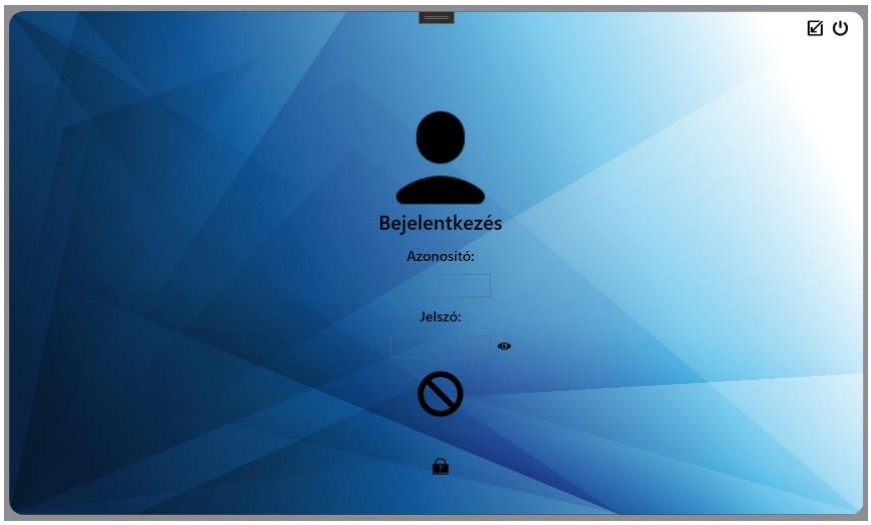
2.2. ASZTALI KLIENS (SZEKERES DÁNIEL)

2.2.1. A program célja

Az alkalmazás elsősorban a horgászegyesületeknek készült, irodai felhasználásra. A lejelentett fogásokkal való munka megkönnyítése, naprakész, friss adatok szolgáltatása volt a cél.

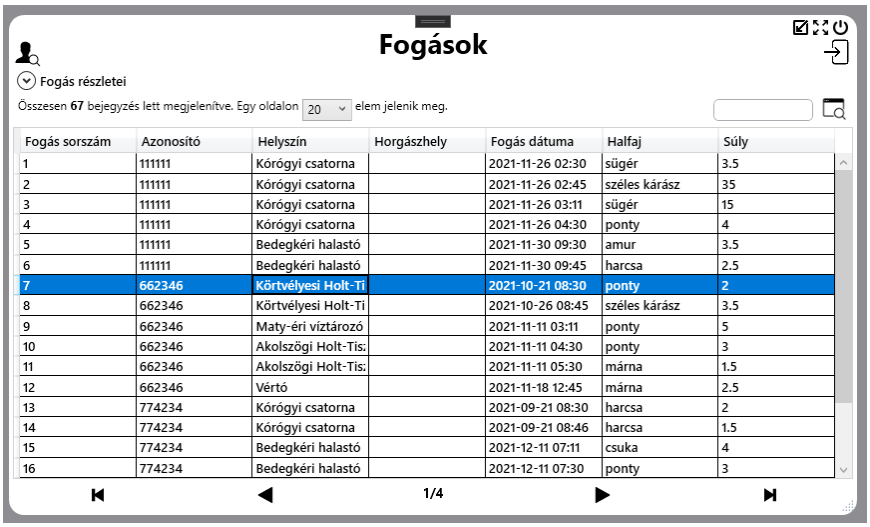
2.2.2. A program használatának bemutatása

A XAMPP Control Panel indítása után, a megjelenő menün belül az Apache, és a Mysql modulokat szükséges futtatni, hogy az adatbázissal történő kommunikáció meg tudjon valósulni. A mellékelt adatbázis fájlok importálása után az asztali alkalmazás a saját parancsikonjával indítható.



15. ábra Asztali alkalmazás bejelentkezés

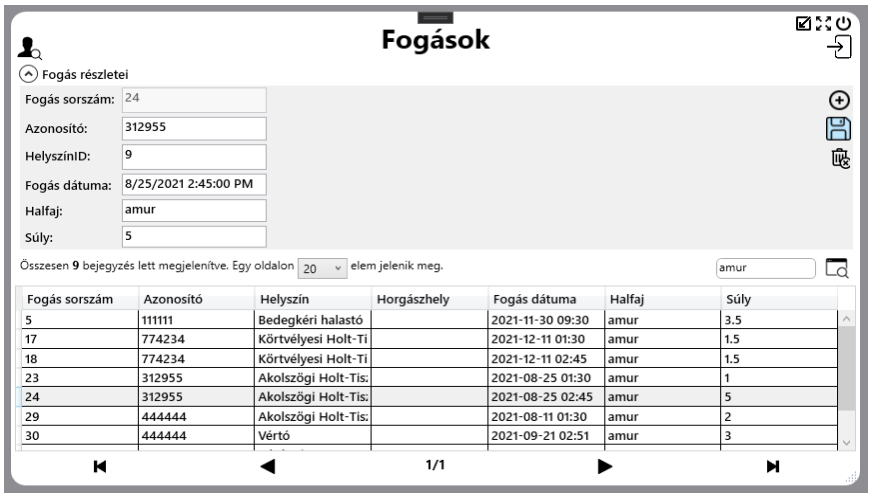
Először a bejelentkező ablak jelenik meg, ahol az adminisztrátor jogú felhasználó(k)hoz tartozó azonosítóval és jelszóval tudunk belépni. Nem megfelelő bevitel esetén hibaüzenet tájékoztatja a felhasználót. A jelszó beviteli mező mellett lévő „szem” ikonra kattintva a jelszó láthatóvá válik. Megkönnyítve a jelszó beírását például idősebb felhasználóknak.



Fogás sorszám	Azonosító	Helyszín	Horgászhely	Fogás dátuma	Halfaj	Súly
1	111111	Kórógyi csatorna		2021-11-26 02:30	sügér	3.5
2	111111	Kórógyi csatorna		2021-11-26 02:45	széles kárász	35
3	111111	Kórógyi csatorna		2021-11-26 03:11	sügér	15
4	111111	Kórógyi csatorna		2021-11-26 04:30	ponty	4
5	111111	Bedegkéri halastó		2021-11-30 09:30	amur	3.5
6	111111	Bedegkéri halastó		2021-11-30 09:45	harcsa	2.5
7	662346	Körtvélyesi Holt-Ti		2021-10-21 08:30	ponty	2
8	662346	Körtvélyesi Holt-Ti		2021-10-26 08:45	széles kárász	3.5
9	662346	Maty-éri víztározó		2021-11-11 03:11	ponty	5
10	662346	Akolszögi Holt-Tis:		2021-11-11 04:30	ponty	3
11	662346	Akolszögi Holt-Tis:		2021-11-11 05:30	márna	1.5
12	662346	Vértó		2021-11-18 12:45	márna	2.5
13	774234	Kórógyi csatorna		2021-09-21 08:30	harcsa	2
14	774234	Kórógyi csatorna		2021-09-21 08:46	harcsa	1.5
15	774234	Bedegkéri halastó		2021-12-11 07:11	csuka	4
16	774234	Bedegkéri halastó		2021-12-11 07:30	ponty	3

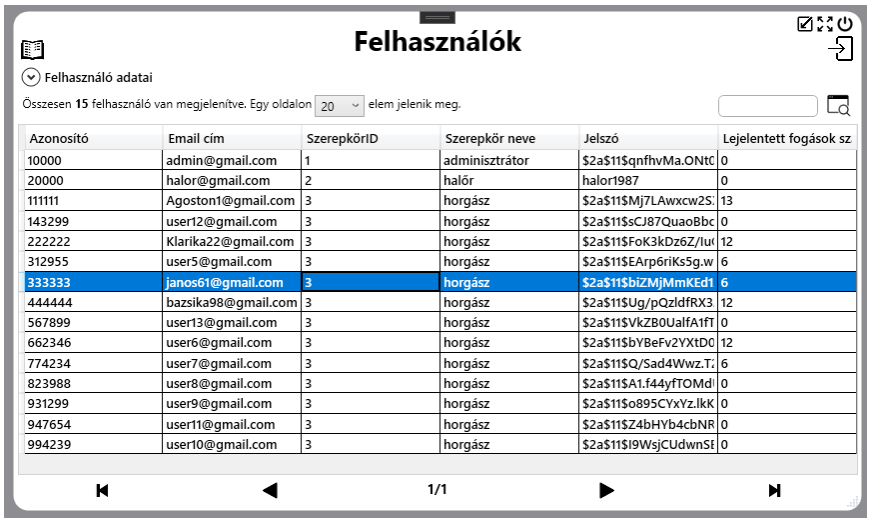
16. ábra Asztali alkalmazás fogások kezelőfelülete

Bejelentkezés után egy fogások listanézet ablak nyílik meg, ahol látszódik az összes bejelentett fogás. Az egy lapon megjelenő bejegyzések száma módosítható. Így mindig csak az egy oldalon megjeleníteni kíván mennyiségű adat töltődik be, nem kell várni amíg esetleg több száz bejegyzés egyszerre megjelenik. Az alsó részen elhelyezett lapozó gombokkal léphetünk előre/hátra, vagy az első/utolsó oldalra. Szintén ezen a területen jelenik meg az aktuális oldalszám is. A jobb felső részen megjelenő keresőmezőbe beírt karakterekre szűrést alkalmazhatunk. Ekkor csak azok a fogások jelennek meg amelyek valahol tartalmazzák a megadott karakterláncot. A jobb felső sarokban elhelyezésre került egy kijelentkezés gomb is, amely a bejelentkező oldalra irányítja a felhasználót. Az oszlopok címsorára kattintva rendezi az adott oszlop tartalmának megfelelően a sorokat. Ez lehet abc szerinti, időrendi sorrend vagy számadatoknál csökkenő/növekvő rendezés.



17. ábra Asztali alkalmazás fogások kezelőfelülete

Adatmódosítást vagy törlést végezhetünk, ha kijelölünk egy rekordot, és a bal felső területen elhelyezkedő „Fogás részletei” gombra kattintunk. Az itt lenyíló menüben tudjuk elvégezni a kívánt módosításokat, törlést vagy akár új fogás rögzítését is. Jelenleg nem minden adatot lehet módosítani. Ezt a felületet az adminisztrátor kezeli, aki tisztában van az adatok felépítésével, ezért a megkötéseket nem részleteztük.



18. ábra Asztali alkalmazás felhasználók kezelőfelülete

Bal felső sarokban található egy gomb, amely ikonjával is utal a funkciójára. Erre kattintva az aktuális ablak bezáródik és a felhasználók listanézet nyílik meg. Itt ügyeltünk arra, hogy ha az előző ablak teljes képernyős módban volt, akkor ez az új ablak is teljes képernyőn jelenik meg. Ellenkező esetben középen fog megjelenni.

Ha törölni szeretnénk egy felhasználót, ez abban az esetben engedélyezett, ha nincs hozzá kapcsolódó fogás. Ha van, akkor erről egy felirat tájékoztat és inaktívvá válik a törlés gomb.

A felhasználók kezelőfelületén a „könyv” ikonnal rendelkező gomb vált a fogások nézetre. Az ablakok egérrel való húzásra mozgathatóak, jobb alsó saroknál húzva a méretük változtatható.

2.3. WEBOLDAL (TOLDI RICHÁRD)

2.3.1. A weboldal célja

2.3.2. A weboldal használatának bemutatása

Az alkalmazás a belépés után lehetőséget ad a horgászoknak a fogásaikat lejelentésére. Gyorsan és egyszerűen a szükséges adatok megadása után le is tudják jelenteni a fogott halat, amit haza kívánnak vinni és ezzel elkerülik az esetleges büntetést, ami a jelentés hiányából következne.

2.3.3. A weboldal használatának bemutatása

Az alkalmazás működéséhez elengedhetetlen az adatbázist kezelő külső alkalmazás pl.: XAMP a beimportált adatbázissal, amit az adatbázis mappában meg lehet találni. A program működéséhez elengedhetetlen a Nodejs legújabb verziója és a Vuejs környezet telepítése.

A program letöltése után a program mappájába a node modulok telepítése szükséges, hogy leszedje a futáshoz szükséges csomagokat, amelyek felhasználásra kerültek a fejlesztés során. A mappából megnyitott PowerShell ablakba kiadott npm i vagy npm install segítségével a csomagok könnyen letölthetőek. A Web API kapcsolja össze a kliens programok funkcióit az adatbázissal így annak futtatása is nélkülözhetetlen, amihez a Visual Studio szükséges.

A program forráskódja a weboldal mappájából indított PowerShell ablakba a code . parancs kiadásával érhető el a Visual Studio Code-ban az npm run serve parancs segítségével lehet futtatni a programot a localhost:8080 -as portján, de erről a PowerShell ablak tájékoztat. Google Chrome böngészővel dolgoztunk, de más böngészőben is elérhető a localhost 8080-as porton vagy amire kiosztja a PowerShell.

19. ábra Weboldal bejelentkezési felülete

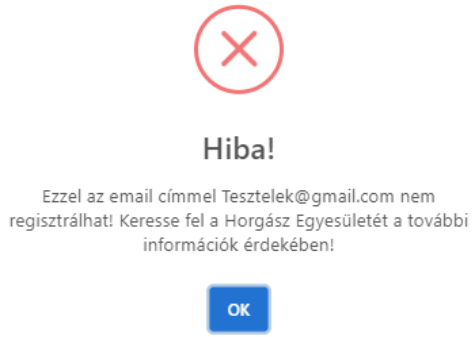
A felhasználó először a bejelentkezési oldallal fog találkozni. Három fiktív felhasználóval már be is lehet lépni, akiket előzőleg regisztráltunk a Firebase adatbázisába.

Belépési adatok:

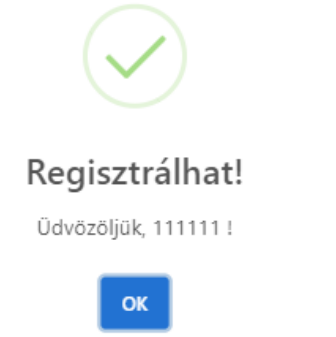
- Horgász: klarika22@gmail.com / Klarika12
- Adminisztrátor: admin@gmail.com / admin1987
- Halőr: halor@gmail.com / halor1987

A bejelentkezési oldal alatt található a regisztrációs oldal elérési útja és az elfelejtett jelszó funkcióé is, a funkció jelenleg fejlesztés alatt áll, ha rákattintunk tájékoztat is az alkalmazás minket. A még nem regisztrált felhasználókat a regisztrációs oldalon lehet regisztrálni a Firebase adatbázisába, az általunk készült adatbázisban megtalálhatóak a regisztrálható felhasználók, de az alkalmazás mappájában is van pár kiemelve egy bejelentkezési nevű fájlban. A regisztrációs oldalra történő lépéskor tájékoztat minket az oldal, hogy kik regisztrálhatnak. Fontos, hogy nem lehet akármilyen email címmel regisztrálni, csak azokkal, amik az adatbázisunkban is szerepelnek, a program ezt ellenőrzi.

20. ábra Weboldal bejelentkezés



21. ábra Regisztráció rossz emaillel



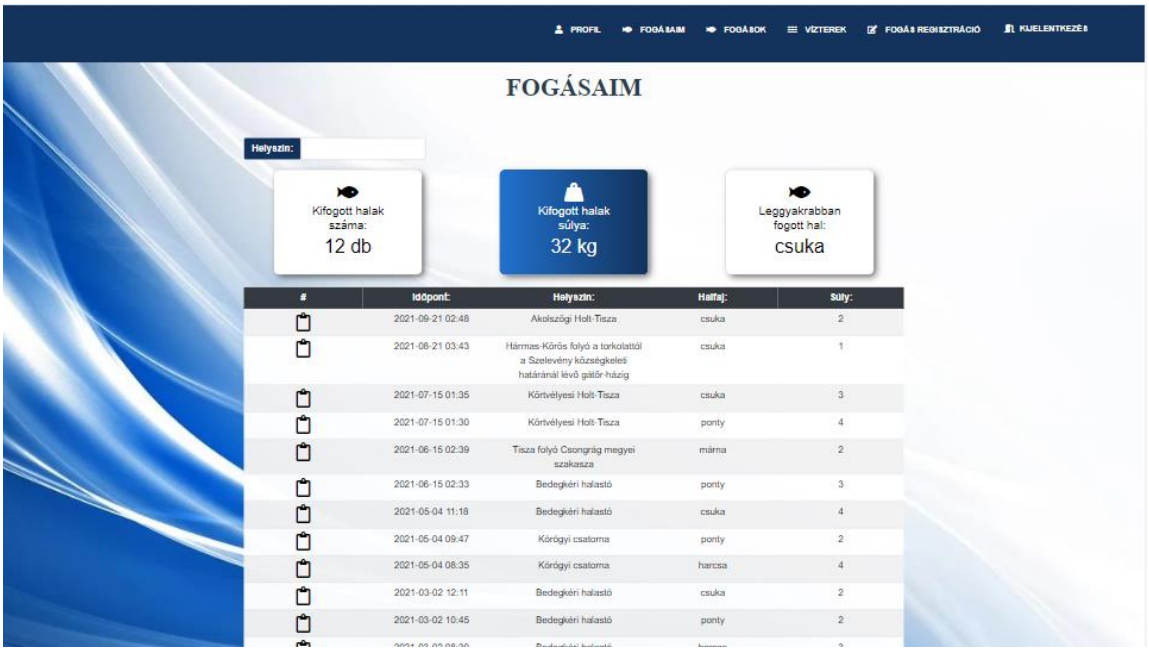
22. ábra Regisztráció megfelelő emaillel

Az oldal tájékoztatja a felhasználót, ha nem regisztrálhat a megadott email címmel.

23. ábra Profil oldal

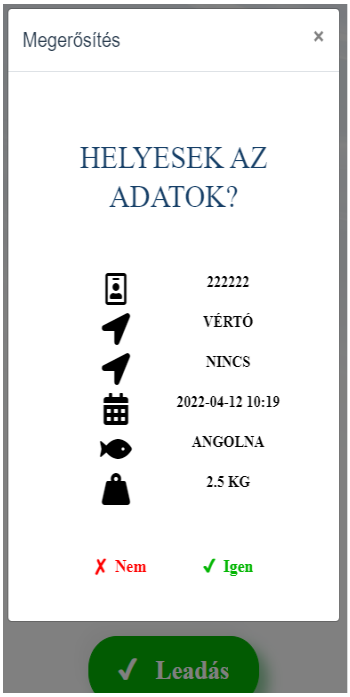
A sikeres belépés vagy regisztráció után a felhasználó a profil oldalt láthatja, ahol a horgász módosítani tudja az adatbázisunkban szereplő email címét a módosítási gombra kattintva. Az email cím jelenleg csak a központi adatbázisban módosul így belépni a módosítás után is az eredetivel fog tudni. A módosítás után az oldal frissítése szükséges, hogy lássuk a profilunkban az új címet.

Az oldal reszponzív így kisebb eszközökön is használható, mobilokon már a hamburger menüt láthatjuk.



24. ábra Horgász felhasználó fogásai

A fogásaim menüpont alatt láthatja a horgász a korábban lejelentett fogásait és a helyszínek alapján keresni is tud köztük, megtudja nézni, hogy az adott helyen milyen és mekkora halakat fogott ki és vitt el. A fogásai mellett lehet látni pár statisztikai adatot is, ami a későbbiekben bővíthető is. A horgász a fogások oldalon az országban az összes horgász által jelentett fogások között is tud keresni és a fogásaim oldalhoz hasonlóan statisztikai adatokat is láthat. Készült a vízterekhez is egy ilyen oldal táblázattal, hogy láthassa a horgász mely vizeken tud horgászni.



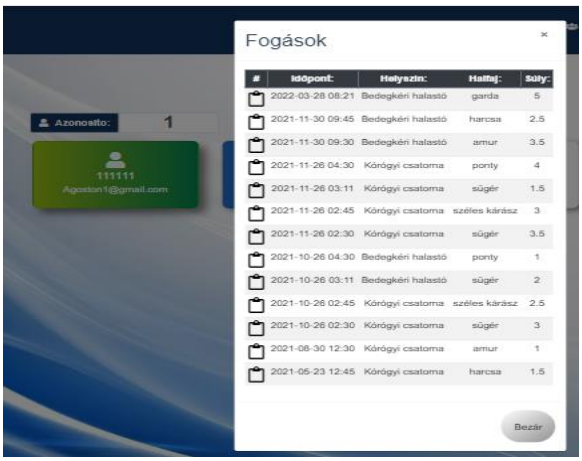
25. ábra Fogás jelentés teteje *26. ábra Fogás jelentés alja* *27. ábra Fogás jelentés elfogadása*

A horgászok felületének fő funkciója a fogás jelentése, az oldal egy űrlap kitöltésével válik leadhatóvá. Az oldal jobb felső sarkában található egy kis halacska amire, ha rákattint a felhasználó előjönnek a fogásai időrendi sorrendben. Megtudja nézni a fogásait még a jelentés előtt, hogy ha elbizonytalanodna, hogy lejelentett-e már amit szeretne vagy nem. Csupán a helyszínt kötelező megadni, ahol fogta a halat, a kifogott hal fajtáját és a súlyát 0.5 kg pontossággal. Minden más adatot a belépésekor rögzítünk, hogy ezekkel ne lehessen visszaélni. Elküldés előtt egy megerősítés figyelmeztetés felugrik, amit, ha elfogad a fogás megjelenik az adatbázisban.

28. ábra Halőr horgászok oldala

Halőrként is először a profil oldalt fogjuk látni, az email cím halőr és adminisztrátor esetében nem módosítható. Felhasználók menüpont alatt a jelenleg regisztrált horgászokat láthatjuk a központi adatbázisunkból és az engedélyszámuk alapján tudunk keresni, szűrni. Elég a keresés mezőbe az engedélyszámot beírni és már szűri is a kis kártyákat. A kártyára kattintás esetén a halőr lekérheti a horgász fogásait időrendi sorrendben így azonnal láthatja, hogy a horgász által kifogott hal le van-e jelentve. A legelső fogás adata lesz az utolsó fogás, amit a horgász jelentett.

A képen látható, hogy az egyes szám beírásával a megjelenített felhasználók le lettek szűkítve azokra, akiknek szerepek az egy az engedélyükben. A 111111-es felhasználó kártyájára kattintva már látni is lehet az általa jelentett fogásokat. Ez a megoldás kisebb monitorokon jobbnak bizonyult, mint a táblázatos, mobilon is jól kezelhető ez az oldal.

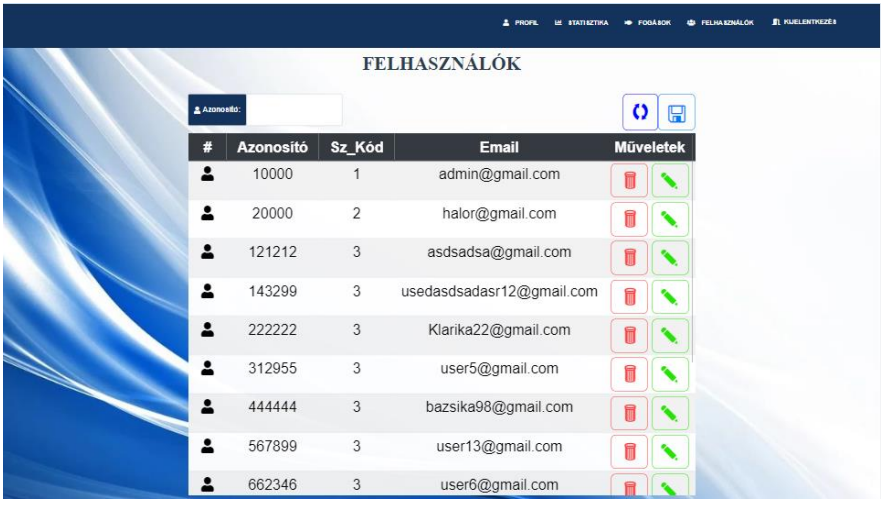


29. ábra Horgász fogásainak listája

30. ábra Halőr és Adminisztrátor fogások táblázata

A fogások menüpont alatt egy részletesebb táblázat található a jelentett fogásokról, amit a halőr és az adminisztrátor is láthat. A táblázatban lehet keresni, engedélyre, dátumra, helyszínre, halfajra, súlyra, horgászhelyre is. A megjelenő adatok számát lehet szabályozni, az adatok sorrendjét az oszlop nevére kattintva tudjuk csökkenő vagy növekvő vagy ABC sorrendbe rakni. Sok adat esetén a táblázat lapozhatóvá válik.

Az oldal reszponzív ahogyan a táblázat is, de kisebb monitorokon elkell húzni a táblázatot oldalra, hogy jól lehessen látni a végét, kisebb monitorokon a kártyás oldal ajánlott.



31. ábra Az adminisztrátor felhasználókat kezelő oldala

Az adminisztrátor extra oldala a felhasználók oldal, ahol a felhasználók láthatóak egy táblázatban. Itt az engedélyszámuk alapján lehet szűrni, keresni. A felhasználók kilistázásával műveleti gombok is előjönnek. A törlésnél a piros kukára kattintás után meg kell erősíteni, ha törölni akarunk. Minden művelet után frissíteni kell a táblázatot, amire figyelmeztet is az

oldal. Új felhasználót is tudunk felvinni a kis floppy-s gombbal és mellette található a táblázat frissítése gomb. A megadott információk után az input mezőben a tabulátor gomb megnyomásával vagy a felugró ablak fehér területére kattintva, de a szerepkör állítása után is lefut az ellenőrzés és ha minden adatot jól adtunk meg, leadhatóvá válnak. A már meglévő felhasználók adatait is tudjuk módosítani, a zöld ceruza gombra kattintva. A módosítás menete azonos az új felhasználó rögzítésével.

3. FEJLESZTÉSI LEHETŐSÉGEK

3.1 ADATBÁZIS FEJLESZTÉSI LEHETŐSÉGEI (TOLDI RICHÁRD)

A fejlesztés során láttuk a felhasználói tábla esetén, hogy az engedélyszám mint elsődleges kulcs nem célszerű, mivel így annak a változtatása problémákat okozhat. A jövőben érdemes külön azonosító számmal ellátni az egyedeket. Helyszínek bővítése, megyék bevezetése, felhasználókról több adattárolásának lehetősége név, telefonszám esetleg cím vagy személyi igazolvány szám, hogy a halőr letudja kérni a személyes adatokat is és az okmányokat ellenőrizni tudja, hogy tényleg a regisztrált személy használja a kiadott engedélyt és nem jogtalanul valaki más. Az adatbázisba külön táblán a kifogható halak fajtáját is lehetne tárolni. Hibajelentési funkció esetén a jelentéseket is lehetne tárolni. Külön megyékre bontani a helyszíneket. Adatbázisban tárolni a horgászhelyeket azoknál a víztereknél, ahol van ilyen bontás.

3.2 MOBIL KLIENS FEJLESZTÉSI LEHETŐSÉGEI (KOVÁCS FRUZSINA RÉKA)

Jelenleg a belépési funkció nem végleges, mivel a szerver nem képes a JWT kezelésére, így csak egy kezdetleges belépési funkciót kaphatott a program. A jövőben ez a biztonsági kérdés a JWT használatával megoldódik. Az útvonalak sem védettek még, de egy mobil telefonon nem tud a felhasználó előre vagy hátra lépkedni a lapok között mobil alkalmazás esetén. Az útvonalak védetté tétele is lehetséges lesz a jövőben a JWT implementálásával és az állapot kezeléssel.

Regisztrációs oldal fejlesztése a bejelentkezés miatt célszerű további fejlesztési lehetőség. Nélküle az ügyintézőnek kell majd újabb és újabb adatokat felvinni az adatbázisba, hogy aztán be tudjanak jelentkezni az alkalmazásba a felhasználók. E-mail cím és jelszó megadását követően a halőr a regisztrációs gombra kattintva tud majd regisztrálni, ha pedig az adatok helyesek, az e-mail cím valós, akkor a regisztráció sikeres. Ezután az alkalmazás elirányít majd a bejelentkezés oldalra.

Fogás részleteinek lapozhatósága is egy további fejlesztési lehetőség. Egyszerűsítené a navigálást az adott horgász fogásai között. Jelenleg egy fogás részlet látható amikor kiválasztjuk a listában, ha pedig másik fogás részletét meg akarjuk tekinteni vissza kell lépni a

fogások listába és ki kell keresni a megtekintendő fogást. Lapozható részletek esetén ez a folyamat egyszerűsödne, kevesebb kattintást és időt venne igénybe.

A hibaüzenetek megjelenítésének finomhangolása is jövőbeni fejlesztési lehetőség, Jelenleg különböző „alerteket", figyelmeztetéseket dob az alkalmazás hiba esetén. „Pop-up" -okkal sokkal felhasználó barátibb, és vizuálisan szebb lenne a hibakezelése az applikációnak.

3.3 SZERVER FEJLESZTÉSI LEHETŐSÉGEI (TOLDI RICHÁRD)

Jelenleg az azonosítás a klienseknél külön felhő alapú szolgáltatással és adatbázissal történik, mert a Web API nem képes még Json Web Token -nel kommunikálni a webes kliensekkel. Azonosítási funkció létrejöttével a felhasználói adatok módosítását már lehetővé lehet tenni a kliens programokon keresztül is, ha esetleg változna valamelyik adata.

Statisztikai adatok számítása és azok lekérdezésének lehetőségét megteremteni a kliensek számára. Akár halfajonként éves összesítő létrehozása a felhasználó számára vagy országos szinten is. Vízterületenként kifogott halak számának és súlyának nyomon követésére GET végpont létrehozása. Beérkező adatok szigorúbb ellenőrzése, jelenleg kevés ellenőrzésen esnek át a beérkező adatok ezért sérülékeny lehet a rendszer egy esetleges támadás esetén.

3.4 WEBOLDAL FEJLESZTÉSI LEHETŐSÉGEI (TOLDI RICHÁRD)

Belépés, azonosítás és regisztráció a Web API-n keresztül az új felhasználók számára, ha esetleg egy átmeneti időszakban a papír alapú és a mobilos jelentés is elfogadott lenne, akkor szabadon dönthet a felhasználó az alkalmazás használatáról. Jelenleg csak azok képesek regisztrálni, akik az adatbázisban is szerepelnek.

Adatok módosítása oldalon, ne csak a mi adatbázisunkba módosuljon az email cím, hanem a Google Firebase adatbázisában is, hogy az új email címmel is be tudjon lépni. Jelenleg a módosítás után a régi címmel tud belépni, mert a Firebase felhő alapú adatbázisban a változtatás nem történik meg. Az email cím módosításhoz hasonlóan a többi tárolt adat módosítását is bele lehet tenni, ha már több adatot is fogunk tárolni a felhasználóról.

Jelszó emlékezetető funkció, illetve email-cím ellenőrző funkció a regisztráció során. Fekete/fehér mód beépítése, hogy este és nappal is jól látható legyen az alkalmazás.

Több személyes adat kimutatása az adatbázis bővítésé esetén és azok módosítási lehetősége, ha valamelyik adata változna a felhasználónak.

Hibajelentési funkció vagy akár véleményezési lehetőség, esetleg későbbi lehetőségek írása a központ felé, hogy lássák a fejlesztők mire van igény vagy mit kellene módosítani a felhasználói klienseken így a kommunikáció is közvetlenebb, gyorsabb és hatékonyabb lehetne a fejlesztő és a felhasználók között.

Tooltip -ek[23] használata az oldalon, gombok és beviteli mezők esetén is a felhasználók segítésére. Az adminisztrátori felület bővítése, kártyás felhasználói oldallal, mint ami a halőrnek is van már, de műveleti gombokkal. A kártyás megoldás kisebb készülékeken, mobilon jobb és szebb, mint a táblázatos verzió. Fogások, helyszínek és szerepkörök kezelhetőségének felülete is belekerülhet a jövőben, mint jelenleg a felhasználókat kezelő felület. Azonnali online csevegő felület az adminisztrátor és a felhasználói vagy halőri szerepkörrel rendelkező között. Reszponzív menü fejlesztése, hogy ha az egér a képernyő jobb oldalához ér akkor is előjöjjön és ne csak a menü ikonjának aktiválásakor lehessen előhívni.

3.5 ASZTALI ALKALMAZÁS FEJLESZTÉSI LEHETŐSÉGEI (SZEKERES DÁNIEL)

Különböző összesítések készítésének lehetőség egy gombnyomásra. Például: összes kifogott hal súlya, vagy összes kifogott ponty súlya stb. Lehetne vízterületekre lebontva összesíteni az adatokat. Akár egy „lista kinyomtatása” funkció is elképzelhető, ami egy csatlakoztatott nyomtatóhoz továbbítja az utasítást. Megjeleníthetné, hogy mely fogások lettek esetleg tilalmi időszakban lejelentve. Ehhez az kellene, hogy megvizsgálja a lejelentett fogás dátumának alapján, hogy milyen halfajra volt abban az időintervallumban fogási tilalom. Ellenőrizné a vízterület alapján, hogy az országos, vagy esetleg speciális helyi szabályok vannak életben a tilalmi időszakokra vonatkozóan. Ezután lehetne megvizsgálni, hogy az adott fogás megfelel-e ezen szabályoknak.

A bejelentkező oldalon lévő jelszóemlékeztető funkció befejezése. Ez egy email-ben küldene további instrukciókat.

[23] Tooltip – Kis információs ablak, ahol felhasználóknak lehet információkat adni a gomb funkciójáról vagy a kitöltési mezőkről.

4. Összegzés (Toldi Richárd, Szekeres Dániel, Kovács Fruzsina Réka)

A záródolgozat megvalósítása során sok tapasztalatot sikerült szerezni. A kezdeti tervek több helyen módosultak, kezdetben számtalan komponenst, dizájnt, illetve funkciót máshogy képzeltünk el. Azonban a fő irányvonalat sikerült tartanunk és az időbeosztást is meglepően szépen tartottuk. Többlet funkciókat is kaptak a programok, leginkább a weboldal és a Web API, kezdetben a weboldal csak a horgászok számára készült volna el, hogy azon tudják a fogásaikat jelenteni, de ez tovább bővült a szerepkörök szerinti funkciókkal és hozzá a szerver is folyamatosan növekedett, hogy ezek a funkciók működhessenek.

A csapatmunka során számtalan online felületet is használtunk. A program kódok megosztására, véleményezésére, illetve ellenőrzésére a GitHub[24] volt segítségünkre, egyszerűvé tette a komponensek fejlesztésének nyomon követesét. Kommunikációra gyakran személyesen került sor az iskolában és az iskolán kívül is, alapvető ötleteket, elképzeléseket, főbb dolgokat egyszerűbb volt személyesen kitalálni, megfogalmazni. Gyakran beszéltünk a Facebook csoportos „chat”[25] felületén is, a felület lehetővé tette az azonnali üzenetváltást és a segítségnyújtást közöttünk, illetve kisebb fájlok átküldését.

A dizájnnal kapcsolatban is eltérő volt az elképzelés, de igyekeztünk a témának megfelelő dizájnt kialakítani, a meghatározott szempontokat szem előtt tartva módosítottunk mindent, hogy időben működőképes programokat tudjunk prezentálni. A dizájn is gyakran változott, ha valaki látott valamit vagy új dologgal találkozott az bele is került a programba, így sok változáson esett át rövid időn belül. A különböző kliensek sajátosságai miatt a dizájnok eltérőek. Mivel egyes látványelemek az egyik kliensen jól néznek ki, de a másik kliensen már a méretbeli eltérés miatt vagy az eltérő elrendezés miatt kevésbé mutatós. Gyorsan és gyakran változtak a programok a fejlesztési időszakban, ezért végig aktív kommunikációra volt szükség.

---

[24] GitHub – Szoftverek fejlesztésénél használt online verzió kezelési felület, biztonságos tárolást és könnyű csapatmunkát, verziókövetést tesz lehetővé.

[25] Chat – Online felület, ami kis üzenetek küldését teszi lehetővé, gyorsan és egyszerűen lehet kommunikálni vagy adatokat és fájlokat küldeni egyszerre akár több személy számára.

A programok fejlesztése során az objektum-orientált programozási módszertan sokban megkönnyítette a munkánkat.

A legnehezebb az elvárások alacsonyan tartása, dokumentum írása volt, mindig találtunk új elemeket, funkciókat, ötleteket, amiket bele lehetne tenni az alkalmazásokba, de idővel rájöttünk, hogy egy jó ötleten alapuló alkalmazás vagy projekt sose lesz teljesen kész. Mindig lesz rajta fejleszteni és szépíteni való, akár csak a kód tisztítása, program optimalizálása vagy új dizájn készítése.

IRODALOMJEGYZÉK

- Reiter István – C# programozás lépésről lépésre Jedlik Oktatási Studio Budapest 2012
- Karsa Zoltán – C# Programozás Jegyzet

Források

- http://webprogramozas.inf.elte.hu/tananyag/wf2/lecke12_lap1.html
- https://datatables.net/manual/
- https://csharptutorial.hu/konyv/
- https://www.w3schools.com/
- https://docs.microsoft.com/en-us/dotnet/csharp/
- https://vuejs.org/guide/introduction.html
- https://v2.vuejs.org/v2/cookbook/using-axios-to-consume-apis.html?redirect=true
- https://firebase.google.com/docs/build
- https://fonts.google.com/
- https://ionicframework.com/docs/
- https://hu.reactjs.org/docs/getting-started.html
- https://icons8.com/

MELLÉKLETEK

1. melléklet Adatbázis tulajdonságai és egyedei

Egyedek: Szerepkorok Felhasznalok, Helyszinek, Fogasok

Tulajdonságok:

Szerepkorok (**SzerepkorID**, szerepkor_megnev)

Felhasznalok (**Azonosto**, SzerepkorID, jelszo, e-mail_cim)

Fogasok (**FogasID**, Azonosito, HelyszinID, datum_idopont, horgaszhely, halfaj, suly)

Helyszinek (**HelyszinID**, vizterulet_neve, vizterkod)

Egyedek és tulajdonságaik (adattípusok):

Szerepkorok

SzerepkorID	szám (INT) – auto increment – elsődleges kulcs – azonosító
szerepkor_megnev	szöveg (VARCHAR (20)) – not null – szerepkör megnevezése

Felhasznalok

Azonosto	szám (INT) – elsődleges kulcs – Magyar Horgászkártya száma (6 számjegy) illetve a halőrök (4 számjegy) /adminok azonosítója
SzerepkorID	szám (INT) – not null – idegen kulcs, jogosultság
jelszo	szöveg (VARCHAR (70)) – not null – vegyes karakterek
e-mail_cim	szöveg (VARCHAR (50)) –not null – unique –e-mail címe

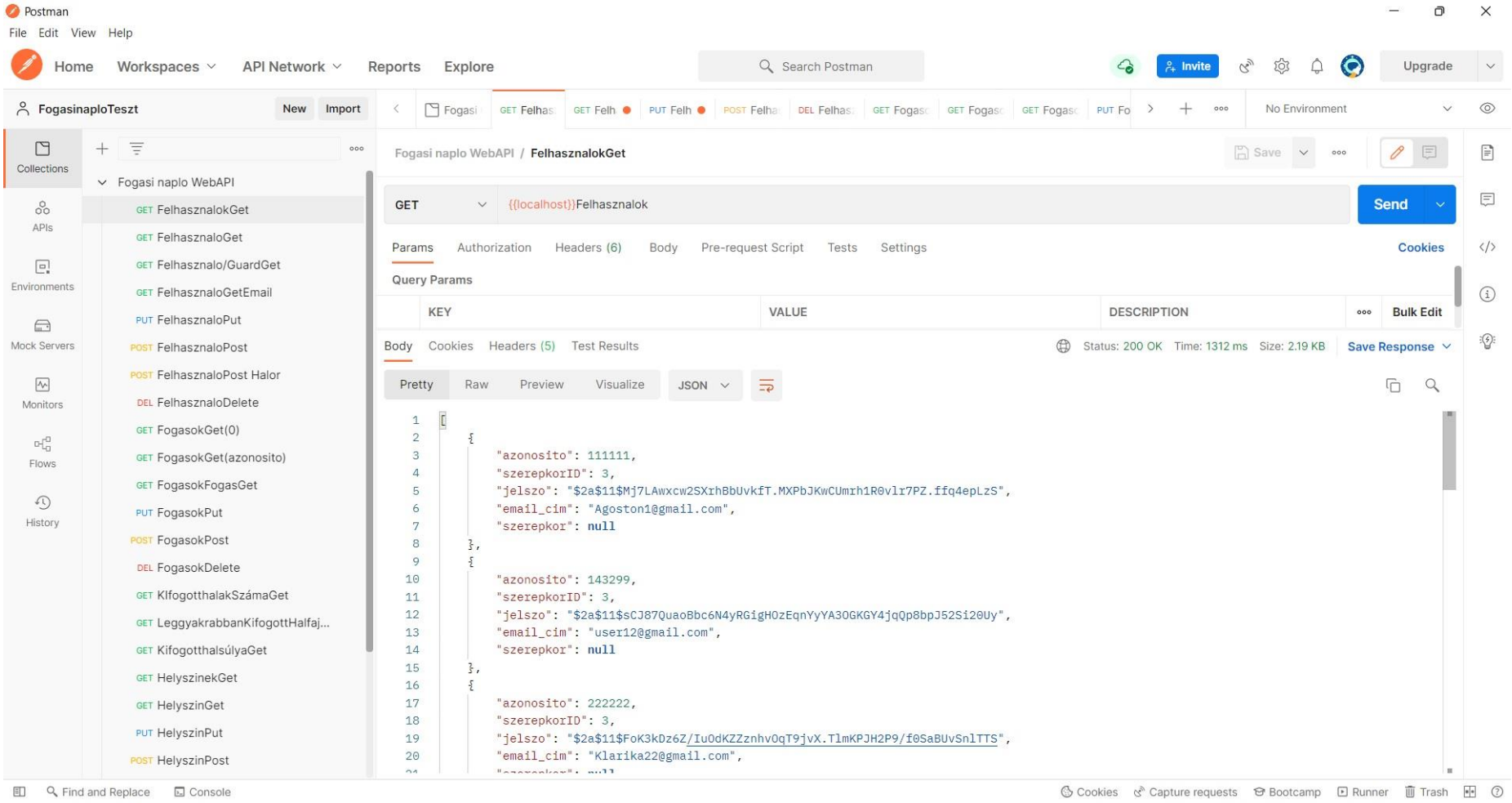
Fogasok

FogasID	szám (INT) – auto increment – elsődleges kulcs
Azonosito	szám (INT) – not null – idegen kulcs
HelyszinID	szám (INT) – not null – idegen kulcs
horgaszhely	szöveg (VARCHAR (10)) – adott vízterület számozott horgászhelyének a kódja, nincs mindenhol pl.:(1 vagy A1)
datum_idopont	dátum és időpont (DATETIME) – not null – unique – a fogás percre pontos ideje pl.:(2021-12-01 15:30)
halfaj	szöveg (VARCHAR (15)) – a felhasználó által fogott hal
suly	tizedes szám (FLOAT) – a fogott hal súlya

Helyszinek

HelyszinID	szám (INT) – elsődleges kulcs – előre meghatározott szám
vizterulet_neve	szöveg (VARCHAR (100)) – not null – unique – hely neve
vizterkod	szám (int) – not null – unique – 7 számjegy

2. melléklet Web API tesztelés Postmannel



Hallgatói nyilatkozat

Alulírott Kovács Fruzsina Réka, Szekeres Dániel, Toldi Richárd a Szegedi Szakképzési Centrum Vasvári Pál Gazdasági és Informatikai Technikum hallgatói kijelentjük, hogy a Fogási napló című záródolgozat a saját munkánk.

Kelt:

.....	.....	.....
Kovács Fruzsina Réka	Szekeres Dániel	Toldi Richárd