

BGSZC Szász Ferenc Kereskedelmi Technikum és Szakképző Iskola

Szakképzés neve: Szoftverfejlesztő és -tesztelő

A szakma azonosító száma: 5-0613-12-03

Szakdolgozat



NFT GALLERY

NFT Gallery

Témavezető:

Kátay Magdolna

Készítette:

Berke Zsanett

Nagy Liliána

Somogyi Márió

Budapest, 2022



Bevezetés

Az NFT Gallery projekt fejlesztő csapata azért döntött a szoftver megvalósítása mellett, mert közelebb szeretnénk hozni az emberek minden napjaiba a jövő egyik „pénzügyi” aspektusát: **az NFT-k világát.**

Projektünk célja, hogy a program felhasználói számára saját, zárt, bárhonnan elérhető, NFT értékeket (képeket) megőrző platformot biztosítsunk. Alkalmazásunk célja megkönnyíteni felhasználóinknak **NFT értékeik megőrzését, nyomon követését, felügyeletét és rendszerezését egy átlátható és könnyen kezelhető, bárhonnan elérhető felületen.** A platform használata során a felhasználók csak a saját képeikhez férnek hozzá, azt kezelhetik.

A fejlesztés életszerűségét adja a különböző alkternatív valuták, fizetési lehetőségek, tokenek megjelenése és fokozott térnyerése a pénzügyi szektorban. A világ, világrend folyamatos változásban van, a technológiai újítások, vadonatúj vívmányok pedig megkövetelik tőlünk fejlesztőktől a nyitottságot új, progresszív technológiák alkalmazása, témaiban rejlő lehetőségek kipróbálása iránt.

Az elkészült szoftverünk hasznos, hiszen piaci részt fed(het) le a végleges fejlesztést követően. Egyre növekszik az NFT-k iránti kereslet és kínálat – a közeljövőben várhatóan növekedni fog a felhasználói igény alternatív tárolási lehetőségek megjelenése, használata iránt is.

Szoftverünk megfelel a modern kor követelményeinek, a **reszponzív tervezésnek** köszönhetően a felhasználó szinte bárhonnan képes **mobil vagy asztali eszközről** egyaránt csatlakozni saját, egyedi rendszeréhez.

A felhasználói élmény fokozása érdekében a megvalósítás során törekedtünk nem csak az esztétikus, könnyed megjelenésre (ezt segítette a semleges színvilág alkalmazása is), hanem az egyszerű, **letisztult felhasználói felület megvalósítására is.** Szoftverünkben csak a felhasználó által szükségesnek ítélezhető funkciók érhetőek el, **könnyen átlátható az oldalstruktúra,** nincs „felesleges” figyelem elterelő tartalom.



A felhasználói élmény fokozását segíti elő a **reszponzivitás**, a weboldal minden eszkösről és nézetből kényelmesen olvasható, megtekinthető.

Az elkészítés során eleget tettünk a REST (**Representational State Transfer**) architectúra megszorításainak, így joggal nevezhetjük „RESTful” architektúrának. Alkalmaztuk a MVC (**Model-View-Controller**) tervezési mintát, ezáltal szébbé, karbantarthatóbbá és átláthatóbbá téve a szoftverünk kódját mellyel megfelelünk a mai kor elvárásainak.

Dokumentációnkban megtalálhatnak **egy általános felhasználói leírást**, megismerkedhetnek a felhasználói felülettel, illetve azok működésével. **Ezt követően egy fejlesztőknek szóló rész következik**, amelyben megtalálhatóak a fejlesztésnél figyelembe vett szempontok, a kód felépítése, kiemelt példák kódrészletekre-megoldásokra, illetve a tesztelésnél vizsgált esetek.

Webalkalmazásunk további fejlesztés alatt van, hogy biztonságtechnikai szempontból a későbbiekbén képes legyen kezelni a fejlett blokklánc technológiát is. mindenkorai terveink között szerepel, hogy projektünk képes legyen egy zárt, megbízható rendszert alkotni az arra vágyóknak, ahol a felhasználók képesek biztonságosan tárolni digitális értékeiket.

A csapatmunka megvalósítása

A projektmunka során a 3 fős fejlesztési teamünk csapatmunka keretében **GITHUB verziókezelő használatával**, rendszeres meetingek, írásos és szóbeli egyeztetésekkel fejlesztette az NFTGallery productot.

A **Trello programon** ticketeken felvettük a fő prioritási pontokat, feladatokat. A fájlmegosztás és a meetingek pedig a Teams-en valósult meg, így haladtunk előre a projekt különböző fejlesztési folyamat ciklusaiban.

A végső projekt publikása a GITHUB mellett a **HEROKU rendszerében** is megvalósult.

PROJEKTÜNK ELÉRÉSE:	
A HEROKU az alábbi linken érhető el:	https://nftgalleryforprojekt.herokuapp.com/
A GITHUB repo elérése:	https://github.com/zsanber/NFTGallery

Adatbázisunk és a szerver is online elérhető (HEROKU), illetve a **képek tárolása felhőben (Cloudinary)** történik.



Commits on Apr 10, 2022
<div><p>teljes server mappa lecserélése. MVC megalósítás imetusalahurr1 committed 22 days ago ✘</p><p>e02ac00 e02ac00 e02ac00</p></div>
Commits on Apr 2, 2022
<div><p>reg/log/forg nagylilian96 committed on Apr 2 ✘</p><p>background zsanber committed on Apr 2 ✘</p><p>css2 nagylilian96 committed on Apr 2 ✘</p><p>background zsanber committed on Apr 2 ✘</p><p>react-router zsanber committed on Apr 2 ✘</p><p>react-router-dom zsanber committed on Apr 2 ✘</p><p>Merge branch 'main' of https://github.com/zsanber/NFTGallery zsanber committed on Apr 2 ✘</p><p>react-router-dom zsanber committed on Apr 2</p><p>bootstrap& axios install imetusalahurr1 committed on Apr 2 ✘</p><p>c3cf8a1 c3cf8a1 c3cf8a1</p><p>0aeefb34 0aeefb34 0aeefb34</p><p>51d9ee3 51d9ee3 51d9ee3</p><p>2eac7b4 2eac7b4 2eac7b4</p><p>05e4985 05e4985 05e4985</p><p>9e5e0c4 9e5e0c4 9e5e0c4</p><p>95a2672 95a2672 95a2672</p><p>490ddaa7 490ddaa7 490ddaa7</p><p>44bb54d 44bb54d 44bb54d</p></div>
<div><p>POSTMARK fix zsanber committed 5 hours ago</p><p>3ac76d3 3ac76d3 3ac76d3</p></div>
Commits on May 1, 2022
<div><p>never done, but build up & done imetusalahurr1 committed 20 hours ago ✘</p><p>Merge branch 'main' of https://github.com/zsanber/NFTGallery imetusalahurr1 committed 21 hours ago ✘</p><p>welcome + debug zsanber committed 21 hours ago ✘</p><p>teszt9 imetusalahurr1 committed yesterday</p><p>teszt8 imetusalahurr1 committed yesterday</p><p>99% imetusalahurr1 committed yesterday ✘</p><p>searchbar done imetusalahurr1 committed yesterday ✘</p><p>reg display update nagylilian96 committed 2 days ago ✘</p><p>form update nagylilian96 committed 2 days ago ✘</p><p>8e44efc 8e44efc 8e44efc</p><p>5587cec 5587cec 5587cec</p><p>9734553 9734553 9734553</p><p>4c75452 4c75452 4c75452</p><p>61f1bdd 61f1bdd 61f1bdd</p><p>e441da6 e441da6 e441da6</p><p>eb5a54a eb5a54a eb5a54a</p><p>47aac78 47aac78 47aac78</p><p>8ae4806 8ae4806 8ae4806</p></div>

I. GITHUB commitok – rendszeres komponens fejlesztés



Témaválasztás

Az alkalmazásunk alapötletét a bevezetőben leírt blokklánc, NFT technológia adta. Rengeteg szó esik manapság a blokkláncról, de még mindig sok félreértes övezi a technológiát. Az NFT definíciója egészen pontosan „*digitális hitelességi tanúsítvány, amely nem másolható és egyedi (vagy)tárgyak tulajdonjogát garantálja*”.¹

Az NFT a Non-fungible token rövidítése, magyar fordítása „nem helyettesíthető token”. A token jelentése „zseton”, ebben az esetben pedig az általában Ethereum blokkláncot használó alternatív kriptovalutákat (és azok egységeit) jelenti. **Az NFT-k esetében azonban nem egy új kriptovaluta a token, hanem az adott digitális, de véges darabszámú vagyontárgy egy egysége.** A nem helyettesíthető voltának itt kezdődik az értelme: az **NFT minden egyedi**. Nem helyettesíthetők egymással, mint ahogy helyettesíthető például 1-1 ezer forintos bankjegy, 1-1 bitcoin.

A témában gyakran emlegetett **Mona Lisa** példa így hangzik:

Attól függetlenül, hogy a világ leghíresebb festménye millió példányban le lett másolva, fotókon, másolatokon és képeslapokon szerepel, az eredeti alkotás értéke továbbra is felbecsülhetetlen, és továbbra is a Louvre vagy a francia állam tulajdona marad. Egy véges darabszámú NFT – legyen az egy digitális képhez, egy internetes játékbéli tárgyhoz, vagy egy valóságban létező oldtimer autóhoz kötve – attól még nem fog kevesebbet érni, hogy lehet róla másolatot vagy fotót készíteni. Ez a három példa már jól mutatja, hogy más-más felhasználási módjai lesznek a különböző típusú NFT-knek és csak évek távlatából tudjuk majd eldöntenи, hogy ezek közül melyik volt maradandó.

Hiszünk benne, hogy a világunk teljesen a digitális tér felé fog orientálóni s az embereknek szükségük lesz az általunk prezentált NFTGallery projekthez hasonló alkalmazásokra.

¹ <https://futuretrends.hu/nft-keszites-eladas-vasarlas/>



NFT GALLERY

Felhasználói dokumentáció

I. A program általános specifikációja

Alkalmazásunk célja megkönnyíteni felhasználóinknak NFT értékeik megőrzését, nyomon követését, felügyeletét és rendszerezését. Mindezt egy átlátható és könnyen kezelhető, bárhonnan elérhető felületen.

II. Hardverkövetelmények a szervergép számára

A backend és szervergép számára ajánlott követelmények

Ajánlott hardver	
Processzor:	4 x 1.6 GHz
Memória:	4GB
Merevlemez:	4GB szabad hely

Szükséges az állandó internetkapcsolat megléte. Az internetkapcsolat sebessége nagy befolyással lehet az alkalmazásunk működésére. A szerver felé irányuló kérések válaszideje a többszörösére nőhet gyenge internetkapcsolat esetén.

Hardverkövetelmények a kliens oldal számára

Ajánlott operációs rendszerek és készülékek mobilesközök esetén:

- Android 8.0 vagy annál magasabb verziószámú operációs rendszer minimum 1280x720 pixel felbontással és 3GB memoriával rendelkező táblagép vagy mobilkészülék.
- A kezelhetőség szempontjából ajánlott a nagyobb képernyőjű eszközök használata.
- Aktív internetes kapcsolat (wifi/mobilinternet)



Laptop, illetve asztali számítógép esetén a következő operációs rendszer és hardverkörnyezet szükséges:

Ajánlott hardver	
Processzor:	4 x 1.6 GHz
Memória:	8GB
Merevlemez:	4GB szabad hely
Operációs rendszer:	Windows 10

Aktív internetkapcsolat szükséges (vezeték nélküli/ vezetékes hálózat)

III. A program telepítése

Mivel programunk böngésző alapú így nincs szüksége a felhasználónak helyi telepítésre. Asztali számítógép, laptop, illetve mobil vagy táblagép esetén csak egy webböngészőre van szükség.

Ajánlott szoftverek:

- Google Chrome
- Mozilla Firefox
- Safari
- Opera

A böngésző megnyitását követően szükséges a keresőmezőbe beírnia a felhasználónak a szerveroldal címét.



IV. A program használatának részletes leírása

Bejelentkezés

Miután beírtuk a webböngészőnk keresőmezőjébe weboldalunk címét (<https://nftgalleryforprojekt.herokuapp.com/>) megjelenik a bejelentkezési felület. A felületen email cím és password megadásával tud a felhasználó belépni. A bejelentkezési felület alján a belépni, vagy regisztrálni kívánó felhasználó megtalálja a Terms and conditions is.

The screenshot shows a white rectangular login form centered on a background with a horizontal gradient from yellow to green. The form has a rounded top-left corner. At the top center, it says "Welcome". Below that are two input fields: one for "Email address" and one for "Password". Underneath the password field is a blue link "Forgot your password?". At the bottom of the form is a large blue button labeled "Login". Below the "Login" button is a smaller blue link "Not a member? Sign up Here!".

II. Login oldal

Elfelejtett jelszó

Ha a felhasználó elfelejtette volna a jelszavát, jelszóemlékeztető kérésére is van lehetőség. Az elfelejtett jelszó a login felületen található gombon érhető el. A felhasználó az email cím megadásával tud emlékeztetőt kérni elfelejtett jelszaváról. (Ez a komponens még fejlesztés alatt van.)

The screenshot shows a white rectangular password reset form centered on a background with a horizontal gradient from yellow to green. The form has a rounded top-left corner. At the top center, it says "Forgotten your password?". Below that is a small explanatory text: "No problem. Enter your email address and we will reset your password.". Underneath is an input field for "Email address". At the bottom is a large blue button labeled "Submit".

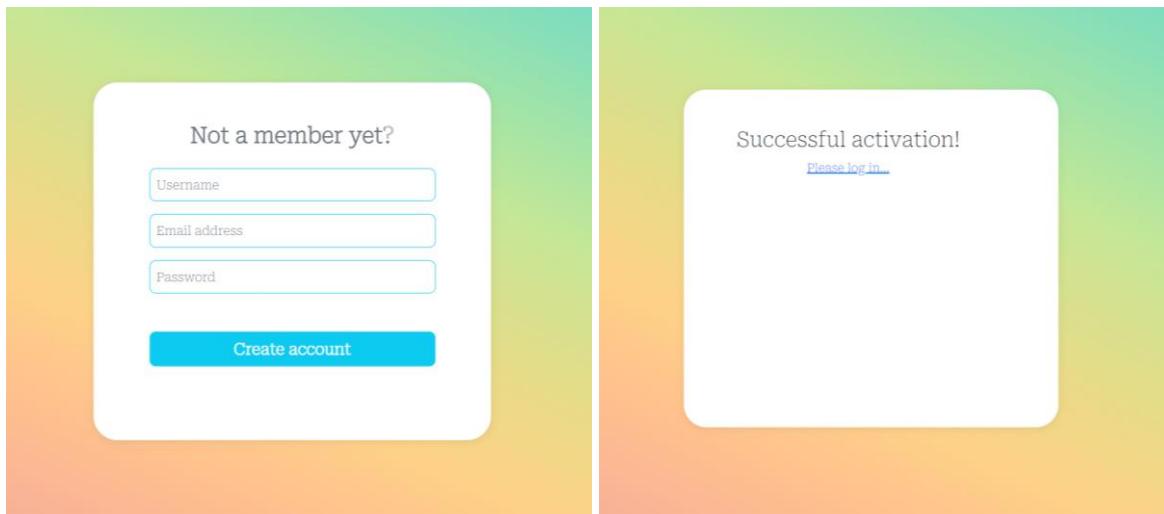
III. Elfelejtett jelszó oldal



NFT GALLERY

Regisztráció

Ha új felhasználó érkezik oldalunkra, a login oldalon található gombra kattintva tovább kerül a regisztrációs felületre, ahol felhasználónév, email cím és jelszó megadásával tud regisztrálni. A regisztrálást követően kap egy emailt, az abban található link segítségével tudja visszaigazolni regisztrációja hitelességét.



IV. Regisztráció oldal, illetve a sikeres emailes visszaigazolást követően az aktivációs link verifikálása

Kezdőlap

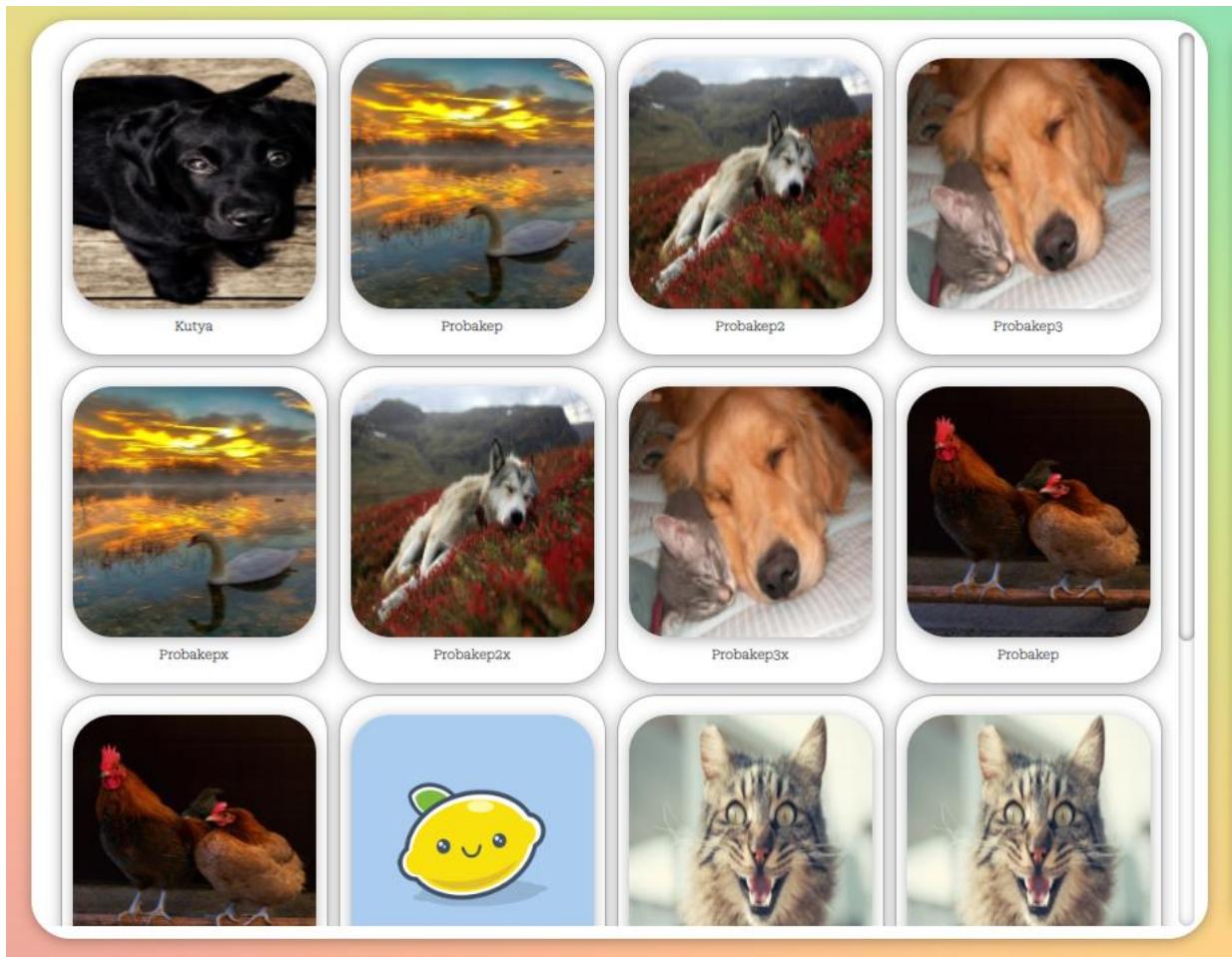
Miután sikeresen beléptünk az oldalra, a következő felület jelenik meg.

III. Főoldal – Home oldal



NFT GALLERY

Itt, főoldalunkon találhatóak oldalunk fő komponensei. Fent a felhasználó a navigációs sáv (navigation bar) használatával tudja kategóriánként szűrni a már feltöltött saját NFT-jeit, amelyek automatikusan frissülnek a weboldal közepén található boxban. Ide bármilyen mennyiségű képet feltölthet a felhasználó, a weboldal görgető sáv használatával könnyíti az esztétikus megjelenést és kezelhetőséget.



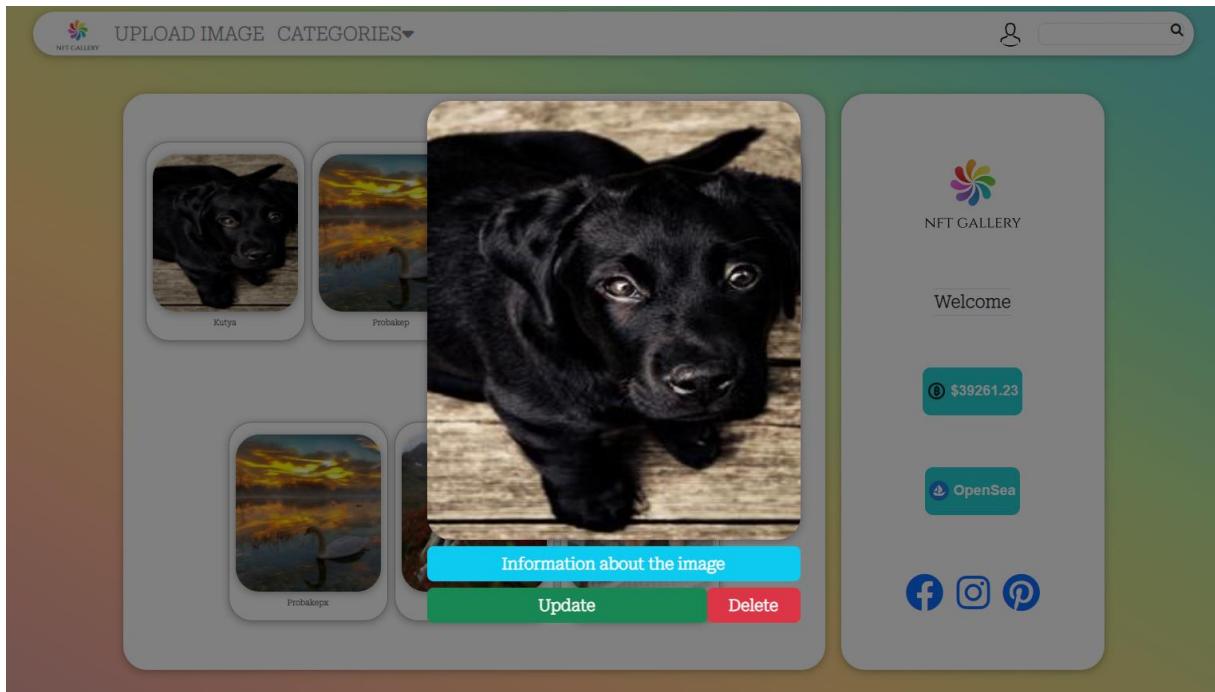
I VI. Homebox - itt éri el, és tudja kezelní a felhasználó az NFT-jeit

Fontos, hogy mivel NFT értékekről van szó, minden felhasználó CSAK a saját képeit éri el. (Az adminisztrátornak sincs joga az összes kép elérésére, hiszen ezek zárt fiókok.)



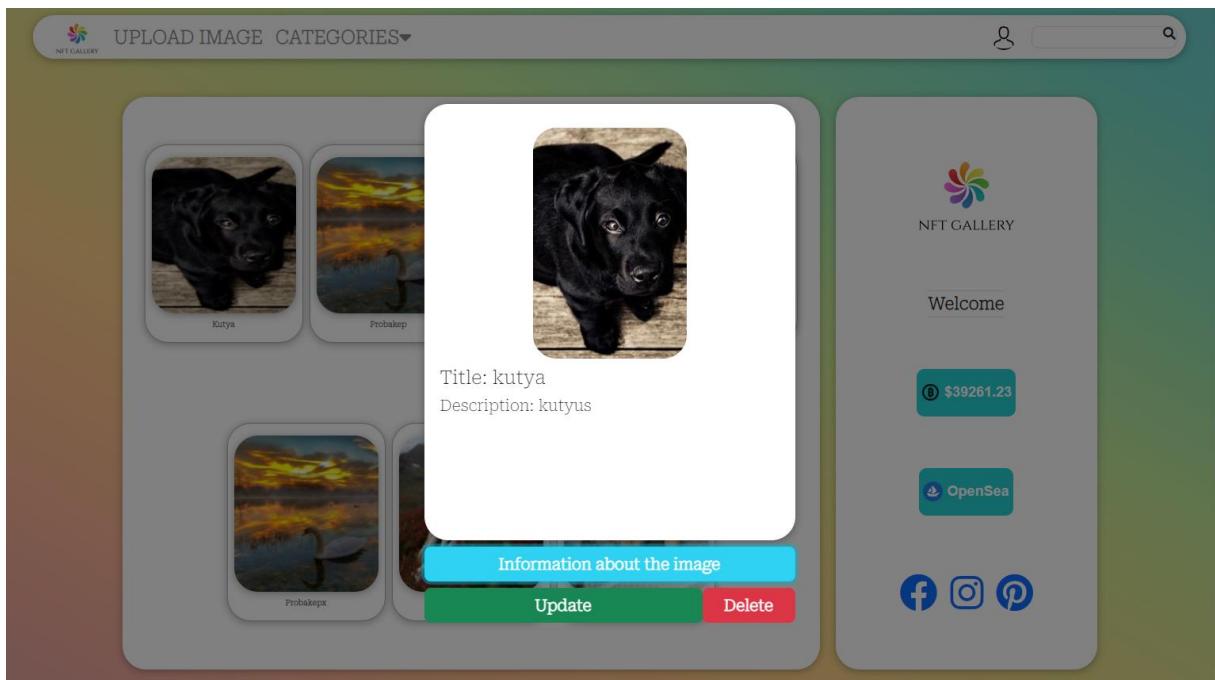
NFT GALLERY

A képekre kattintva a kép felugrik egy ablakban, a kép fölé helyezve az egeret a kép megnagyobbodik, így könnyebben láthatjuk annak részleteit. Ezalatt 3 gomb található.



VI. NFT nagy megjelenés, illetve kép-információs, - módosítási, - törlési lehetőségek

Az „Information about the image” gomba kattintva a képleírás, illetve a kép címe a kép helyén flipelve jelenik meg. Az Update vagy a Delete gomb kiválasztásával pedig a kép szerkesztésére és törlésére is van mód.

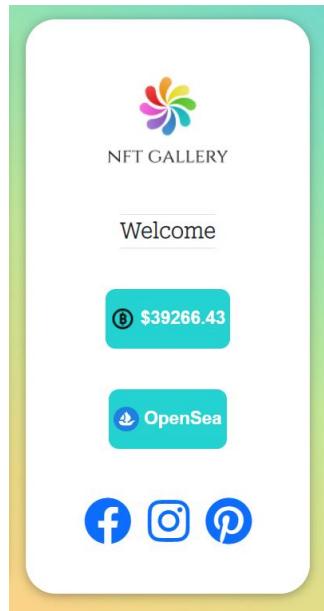


VII. NFT információs tartalom, adatbázisunkból a képhez kapcsolódóan név és leírás is renderelődik



NFT GALLERY

A főoldalon jobb oldalt egy sidebar található, ahol a felhasználónak lehetősége van ellenőrizni a Bitcoin (legismertebb és legnagyobb kriptovaluta) jelenlegi árfolyamát, illetve lehetősége van továbblépni az OpenSea weboldalra, ahol további kriptovaluta árfolyamokról tájékozódhat.



XI. Sidebar – a bejelentkezett felhasználó nevével, a Bitcoin árfolyamával és navigációs linkekkel

Illetve amennyiben a felhasználó ellenőrizni szeretné közösségi platformjait, a megfelelő ikonok kiválasztását követően a Facebook, Instagram, Pinterest oldalakra is átnavigál az oldalsáv.

Navigációs sáv

A főoldalunkon fent található a navigation bar, azaz navigációs sáv. Itt a felhasználó az NFTGallery logóra kattintva mindenkor visszajuthat az alapértelmezett, szüretlen home tartalomhoz.

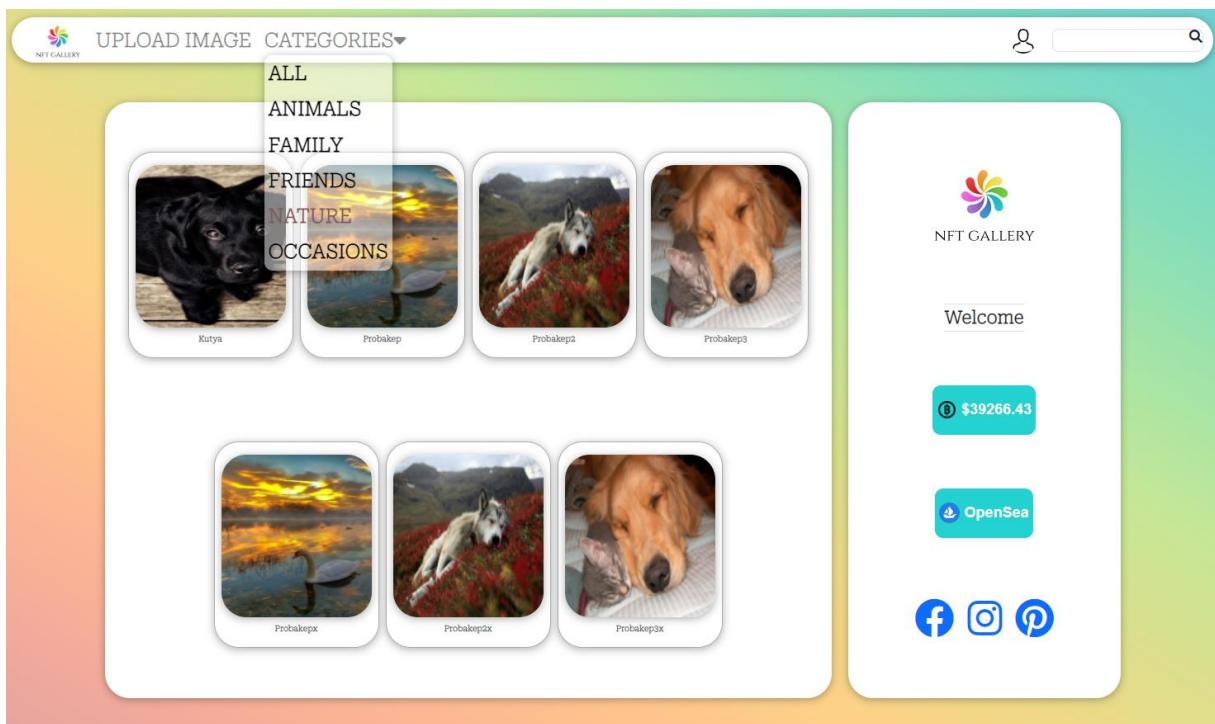
A felhasználó az Upload image gombra kattintva képfeltöltést valósíthat meg. Itt található a categories, illetve a felhasználói lehetőségek jobb oldalt. Valamint a felhasználónak keresési lehetősége is van.



X. Navigation bar – feltöltési, szűrési, keresési lehetőségekkel és felhasználói funkciókat ellátó menüponttal

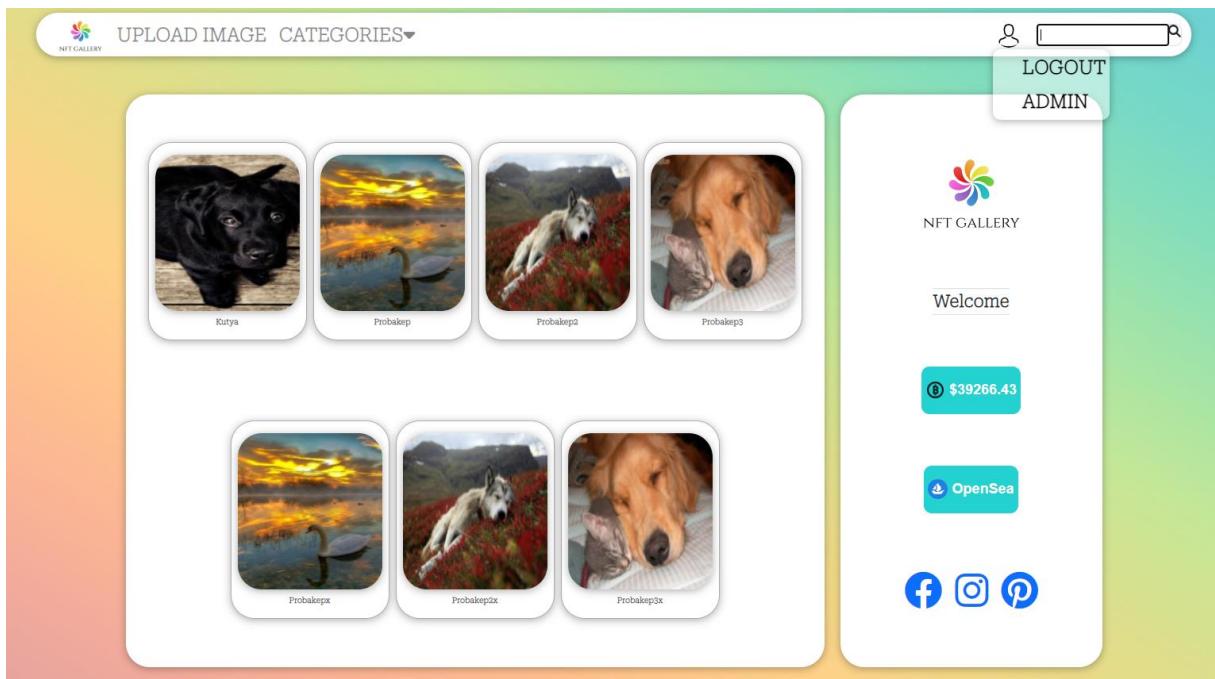


NFT GALLERY



XI. Categories – renderelt tartalom megjelenése a homeboxban a szűrési típus mentén

A navigation barban a Categories használatával a felhasználó megadott kategóriák mentén tudja szűrni a már feltöltött saját NFT-jeit, amelyek automatikusan frissülnek a weboldal közepén található homeboxban.



XII. Logout & Admin funkciók elérhetősége

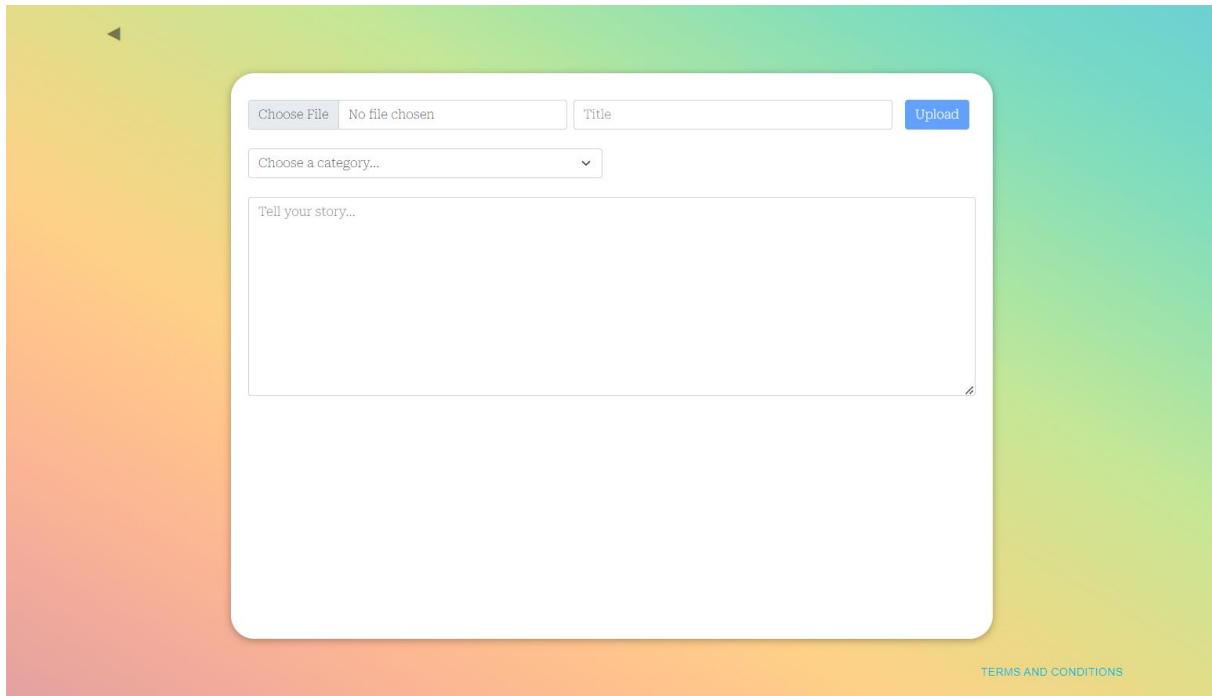
A navigation bar jobb oldalán a felhasználói ikonra kattintva lenyílik a kijelentkezési lehetőség, illetve ha a felhasználó admin, az adminisztrációs felület.



NFT GALLERY

Képfeltöltés

A felhasználó egyik fő lehetősége az NFT képfeltöltés.



XIII. File upload – képfeltöltési lehetőség

A felhasználó feltöltheti a képet, nevet és leírást ad a képnek, illetve kiválasztja a képhez kapcsolható kategóriát, mely alapján később csoportosítani tudja NFT-it.

A képfeltöltés sikereségét követően a felhasználó kap egy megerősítő üzenetet, majd két másodpercen belül visszairányításra kerül a főoldalra.

Amennyiben a felhasználó meggondolta magát, nem szeretne képet feltölteni, a bal oldalt található vissza ikonra kattintva lehetősége van a képfeltöltés nélküli főoldali visszatérésre is.

Fájl/képszerkesztés

A főoldalon található képek bármelyikére kattintva felugrik a kép nagyban, illetve alatta egy Update gomb. A gomra kattintva betölt egy szerkesztési felület, ahol a felhasználó módosíthatja a kép címét, kategóriáját, leírását, illetve magát a képet is. A módosítás mentését követően a felhasználó kap egy megerősítő üzenetet, majd két másodpercen belül visszairányításra kerül a főoldalra.



NFT GALLERY

A screenshot of a file upload dialog box. At the top left is a search bar containing the text "kutya". Below it is a dropdown menu set to "Nature". A large text area contains the word "kutyas". In the top right corner of the dialog box is a blue "Save" button. At the bottom right of the dialog box is a small link labeled "TERMS AND CONDITIONS".

XIV. Update – a kép, illetve a képhez kapcsolódó információk frissítését szolgálja

Amennyiben a felhasználó meggondolta magát, a bal oldalt található vissza ikonra kattintva lehetősége van a képfeltöltés nélküli főoldali visszatérésre is.

Adminisztrációs felület

Ha a felhasználó admin jogosultságokkal bír, a navigation barban az Admin kiválasztva megjelenik a lenti felület, ahol az adminnak lehetősége van a felhasználók áttekintésére, illetve törlésére.

A screenshot of an Admin user list interface titled "Users". The table has columns: USERID, USERNAME, EMAIL, CREATED AT, UPDATED AT, ROLE, and DELETE. There is one row of data: USERID 10, USERNAME valamike, EMAIL haho@haho.hu, CREATED AT 2022-04-25T22:00:00.000Z, UPDATED AT 0000-00-00, ROLE (empty), and a red "Delete user" button in the DELETE column.

USERID	USERNAME	EMAIL	CREATED AT	UPDATED AT	ROLE	DELETE
10	valamike	haho@haho.hu	2022-04-25T22:00:00.000Z	0000-00-00		<button>Delete user</button>

XV. Admin felület – a felhasználók áttekintésére

Amennyiben az admin meggondolta magát, a bal oldalt található vissza ikonra kattintva lehetősége van a képfeltöltés nélküli főoldali visszatérésre is.



Fejlesztői dokumentáció

I. Alkalmazott fejlesztői eszközök, technológiák és programok

A fejlesztés során az alábbi programok, fejlesztői eszközök és technológiák lettek alkalmazva.

Microsoft Visual Studio Code²

A Visual Studio Code (rövidítve: VSCode vagy VS Code) egy ingyenes, nyílt forráskódú kód szerkesztő, melyet a Microsoft fejleszt Windows, Linux és OS X operációs rendszerekhez. Támogatja a hibakeresőket, valamint beépített Git támogatással rendelkezik, továbbá képes az intelligens kódkezelésre (intelligent code completion) az IntelliSense segítségével. Ezen felül testre szabható, így a felhasználók megváltoztathatják a kinézetet (témat), megváltoztathatják a szerkesztő gyorsbillentyű-kiosztását, az alapértelmezett beállításokat és még sok egyebet.

A kódszerkesztőt használtuk a fejlesztésre.

XAMPP³

Egy szabad és nyílt forrású platformfüggetlen webszerver-szoftvercsomag, melynek legfőbb alkotóelemei az Apache webszerver, a MariaDB (korábban a MySQL), valamint a PHP és a Perl programozási nyelvek értelmezői, azaz végrehajtó rendszerei. Ez a szoftvercsomag egy integrált rendszert alkot, amely webes alkalmazások készítését, tesztelését és futtatását célozza, és ehhez egy csomagban minden szükséges összetevőt tartalmaz. A rendszer egyik nagy előnye az összehangolt elemek könnyű telepíthetősége.

Projektünk keretében a MySQL adatbázisunk futtatásához használtuk.

² <https://artsandculture.google.com/entity/m0134xwrk?hl=hu>

³ <https://hu.wikipedia.org/wiki/XAMPP>



MySQL Workbench⁴

Egy vizuális adatbázis-tervező eszköz, amely integrálja az SQL fejlesztést, adminisztrációt, adatbázistervezést, -készítést és -karbantartást egyetlen integrált fejlesztői környezetbe a MySQL adatbázisrendszer számára. Ez a fabFORCE.net DBDesigner 4 utódja, és felváltja a korábbi szoftvercsomagot, a MySQL GUI Tools Bundle-t.

A MySQL adatbázis rendszerünk felépítéséhez és teszteléséhez használtuk.

PHPMyAdmin⁵

Egy nyílt forráskódú eszköz, amit PHP -ban írtak a MySQL menedzs selésére az interneten keresztül. Jelenleg képes készíteni és eldobni adatbázisokat, készíteni/eldobni/módosítani táblákat, törleni/módosítani/hozzáadni mezőket, SQL parancsokat futtatni és a mezőkön kulcsokat kezelní.

Képes az egész MySQL szerver kezelésére (szuper-felhasználót igényel) épp úgy, mint egyetlen adatbáziséra. Az utóbbi megvalósításához be kell állítani a MySQL felhasználót, hogy csak a kívánt adatbázist tudja írni/olvasni.

A MySQL adatbázisunk további fejlesztéséhez és teszteléséhez használtuk.

Postman⁶

A Postman egy komplett eszköztár API fejlesztők részére, a programmal gyorsan és hatékonyan lehet dolgozni az API-val, mivel támogatja a fejlesztők minden munkafolyamatát, továbbá elérhető Mac OS X, Windows, Linux és Chrome felhasználók számára is. A program használata lehetővé tette számunkra, hogy kész front end nélkül tesztelhessük a szerver oldali kódot. Kérést tudtunk indítani a szerver megfelelő end point-jaihoz, ezzel tesztelve az adott controllerek elérhetőségét.

A backendről érkező információk tesztelésére használtuk.

⁴ <https://www.mysql.com/products/workbench/>

⁵ <https://www.phpmyadmin.net/>

⁶ <https://www.postman.com/>



Heroku⁷

A Heroku egy felhőplatform szolgáltatás (PaaS), amely számos programozási nyelvet támogat. Az egyik első felhőplatform, 2007 júniusa óta fejlesztés alatt áll. Hosszú évek fejlesztésnek köszönhetően egyre több nyelvet támogat, így a Java, a Node.js , a Scala , a Clojure, a Python , a PHP és a Go nyelveket is.⁸

A Herokun került megosztásra weboldalunk jelenlegi fejlesztési verziója.

Cloudinary⁹

Egy SaaS technológiai vállalat, amelynek központja a kaliforniai Santa Clarában található, és irodái vannak Izraelben, Angliában, Lengyelországban és Szingapúrban. A cég felhő alapú kép- és videókezelési szolgáltatásokat nyújt. Lehetővé teszi a felhasználók számára, hogy képeket és videókat töltsenek fel, tároljanak, kezeljenek, kezeljenek és szállítsanak webhelyekre és alkalmazásokra. A Cloudinary-t 1 millió webes és mobilalkalmazás-fejlesztő használja több mint 8000 vállalatnál. Az Inc. Magazine a Cloudinary-t a webes képkezelés "arany standardjának" nevezte.

Projektünk keretében a felhasználók által feltöltött képek tárolására szolgál.

JawsDB¹⁰

Egy adatbázis-szolgáltatás (DBaaS) szolgáltató, amely egy teljesen működőképes, teljesen felügyelt, relációs adatbázist biztosít az alkalmazáshoz. Ahelyett, hogy az adatbázisok tárolásával, konfigurálásával, javításával és kezelésével járó gondokat kellene átélnie, a JawsDB egy kattintással biztosítja a relációs adatbázisok felhőben történő kézbesítését és kezelését. A JawsDB könnyen használható, és nincs szükség egyedi kódolásra vagy könyvtárakra az interakcióhoz.

A végső adatbázisunk megosztását szolgáló platform.

⁷ <https://dashboard.heroku.com/>

⁸ <https://en.wikipedia.org/wiki/Heroku>

⁹ <https://cloudinary.com/>

¹⁰ <https://www.jawsdb.com/>



Postmark¹¹

Egy ügyfélkommunikációs platform tranzakciós és marketing e-mailekhez. A Postmark felhőalapú szolgáltatást nyújt, segíti a vállalkozásokat az e-mailek kézbesítésében. A szolgáltatás különféle típusú e-maileket kezel, beleértve a szállítási értesítéseket, barátkéréseket, regisztrációs visszaigazolásokat és e-mailes hírleveleket. Azt is lehetővé teszi a vállalatok számára, hogy nyomon kövessék az e-mailek megnyitását, leiratkozását, visszapattanását és a spamjelentéseket.

A felhasználóval való email kapcsolattartást kezelő komponens (pl. regisztráció).

React¹²

A React (más néven React.js vagy ReactJS) egy ingyenes és nyílt Javascript könyvtár, amely UI-komponenseken alapuló felhasználói felületek építésére szolgál. Karbantartója a Meta (korábban Facebook), valamint az egyes fejlesztőkből és cégekből álló közösség. A React alapul szolgálhat egyoldalas, mobil vagy szerver által renderelt alkalmazások fejlesztéséhez olyan keretrendszerrel, mint a Next.js. A React azonban csak az állapotkezeléssel és az állapotnak a DOM számára történő megjelenítésével foglalkozik, így a React alkalmazások létrehozásához általában további könyvtárak használatára van szükség az útválasztáshoz, valamint bizonyos kliensoldali funkciókra.

Projektünk fejlesztési környezetét a React biztosította. A React eszközrendszerét felhasználva, komponenseken keresztül valósítottuk meg projektünk könnyen átlátható struktúráját.

Node.js¹³

A Node.js egy szoftverrendszer, melyet skálázható internetes alkalmazások, mégpedig webszerverek készítésére hoztak létre.¹⁴ Egy olyan futtató környezet, ami

¹¹ <https://postmarkapp.com/send-email/node>

¹² <https://reactjs.org/>

¹³ <https://nodejs.org/en/>

¹⁴ <https://hu.wikipedia.org/wiki/Node.js>



lehetőséget nyújt JavaScriptben írt programok futtatására szervereken. Szintaktikája egyszerű, érthető.

A szerver oldal kiépítését és működtetését segítette.

NPM¹⁵

Egy csomagkezelő Javascript programozási nyelvhez. A Node.js alapértelmezett csomagkezelője. Egy parancssori kliensből, más néven npm-ből, valamint egy nyilvános és fizetős privát csomagokat tartalmazó online adatbázisból, az úgynevezett npm-nyilvántartásból áll. A rendszerleíró adatbázis elérése a kliensen keresztül történik, az elérhető csomagok pedig az npm weboldalán keresztül böngészhetők és kereshetők. A csomagkezelőt és a rendszerleíró adatbázist az npm, Inc. kezeli.

A különböző csomagok segítségével hatékony kódelemeket, komponenseket tudtunk alkalmazni projektünkben, melyek célja mind a felhasználói élmény fokozása, mind - a tiszta kód elvét szem előtt tartva – az egyszerűen, átlátható kódolás párhuzamosan volt.

Express

Egy olyan webalkalmazás-keretrendszer Node.js-hez amely kiválló funkciókészletet biztosít mobil és webes alkalmazások fejlesztéséhez. Az express telepítése a Node Package Manageren (npm) keresztül történik.

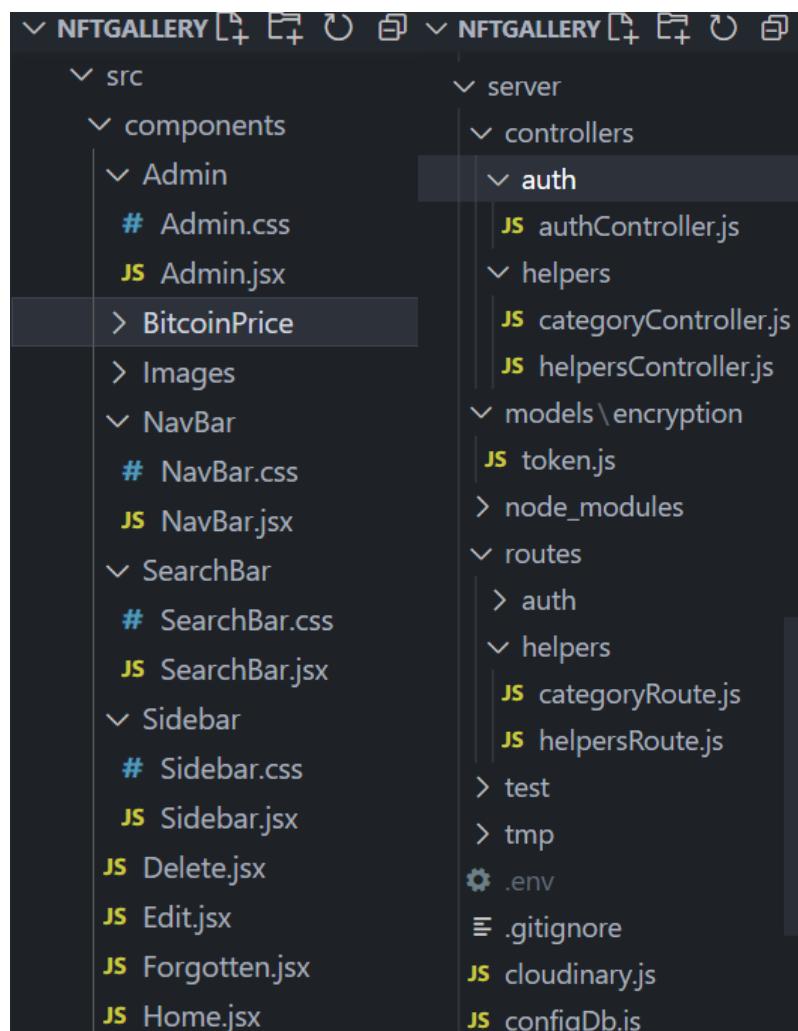
Segítségével többek között beállíthatunk köztes szoftvereket, hogy megválaszoljuk a HTTP kéréseket.

¹⁵ <https://www.npmjs.com/>

II. Tervezési szempontok

A tervezés, majd a fejlesztés során **kiemelt célunk volt, hogy a kód minősége megfelelő legyen.** Ezért kulcsfontosságúnak tartottuk, hogy a **tiszta kód elve alapján járunk el mind a megnevezések, mind az átláthatóságot segítő megjegyzések beszúrásán keresztül** is.

A kód kialakításakor **ugyancsak kulcsfontosságúnak tartottuk az átlátható kódstuktúra kialakítását.** Így mind a frontend, mind a backend oldalán mapparendszert alakítottunk ki a fő gondolati elemek mentén – például a kliens oldalon a fő komponensek és a hozzájuk kapcsolódó css fájl adott mappába került, a könnyebb átláthatóság és kezelhetőség végett.



XVI. Az átláthatóságot segítő mappastruktúra kialakítása mind a client, mind a server oldalon



Kliens oldali komponensek (frontend)

A kódunk megfelel a HTML5 szabványnak, használjuk az ECMAScript ES6 szabványt.

The screenshot shows the code editor interface with the file 'index.html' open. The code is a standard HTML5 document with meta tags, links to CSS and fonts, and a title. A noscript block provides fallback content for users who do not have JavaScript enabled. The code is annotated with several boxes highlighting specific sections: one around the title tag, another around the noscript block, and a third around the entire body content.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.1/css/all.min.css" integrity="sha512-KfkfYDsLkIlw..."/>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@100&family=Roboto+Serif:wght@100&display=swap" rel="stylesheet">
    <title>NFT GALLERY</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

XVII. Index.html fájlunk

A kódolás során a **CSS, SCSS mellett a Bootstrap 5 eszközrendszerét is kihasználtuk a formai megjelenés kialakításánál.**

The screenshot shows the code editor interface with the file 'App.css' open. It contains CSS rules for various components like scrollbars and a modal. Several sections of the code are highlighted with boxes: one for scrollbars, one for a modal content container, and one for a moving image component.

```
.infoImage::-webkit-scrollbar {
  width: 12px;
  margin-right: 10px;
}

.infoImage::-webkit-scrollbar-track {
  -webkit-box-shadow: inset 0 0 6px rgba(0,0,0,0.3);
  border-radius: 10px;
}

.infoImage::-webkit-scrollbar-thumb {
  border-radius: 10px;
  -webkit-box-shadow: inset 0 0 6px rgba(0,0,0,0.5);
}

.modal-content {
  background: none !important;
  border: none !important;
}

.movingImage {
  width: 468px;
  height: 550px;
  background: white;
  border-radius: 30px;
  box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.317);
  cursor: zoom-in;
}
```

XVIII. App.css fájl – példa a CSS/SCSS megjelenés manipulálásra



```
client > src > JS App.js > App > useEffect() callback
You, yesterday | 4 authors (You and others)
1 import React, { useEffect, useState } from 'react';
2 import { BrowserRouter, Routes, Route, useForm } from 'react-router-dom';
3 import './node_modules/bootstrap/dist/css/bootstrap.min.css';
4 import './node_modules/bootstrap/dist/js/bootstrap.bundle.js';
5 import './App.css';
6 import { Register } from './components/Register';
7 import { Login } from './components/Login';
8 import { Forgotten } from './components/Forgotten';
9 import { Home } from './components/Home';
10 import { Upload } from './components/Upload';
11 import { Admin } from './components/Admin/Admin';
12 import { Logout } from './components/Logout';
13 import { Edit } from './components/Edit';
14 import axios from 'axios';
15 import { useNavigate } from 'react-router-dom';

function App() {
  const [user, setUser] = useState(true)
  const [categoryList, setCategoryList] = useState([])
  const [selectedCategory, setSelectedCategory] = useState(0)
  const [userName, setUserName] = useState(localStorage.getItem('userName') ? localStorage['userName'] : '');
  const [userId, setId] = useState(localStorage.getItem('userId') ? localStorage['userId'] : 0);

  useEffect(() => {
    fetchCategoryList();
  }, []);
}
```

XVII. Bootstrap importálása az App.js-ben

A **Bootrstrap** használatával látványos felhasználói élményt vagyunk képesek nyújtani.

Az alapvető megjelenéseket Bootstrappal határoztuk meg, ezt egészítették ki a **CSS/SCSS megoldások**.

```
client > src > components > JS Login.js > Login > sendData
41
42   return (
43     <div className="reg_row">
44       <div className="col-12">
45         <div className="position-absolute top-50 start-50 translate-middle">
46           <div className="box">
47             <form onSubmit={handleSubmit}>
48               <h2 className="text-secondary fw-bold mb-3">Welcome</h2>
49               <input {...register('email', { required: true })} className="form-control mt-4 border border-info rounded p-2" placeholder="Email address" />
50               {errors.email && <p className="err text-danger small">Email address required</p>}
51               <input type="password" {...register('password', { required: true })} className="form-control mt-4 border border-info rounded p-2" placeholder="Password" />
52               {errors.password && <p className="err text-danger small">Faulty password</p>}
53               <a className="text-info fw-bold text-decoration-none mb-4 p-2" href="/forgotten">Forgot your password?</a>
54               <input type="submit" value="Login" className="btn btn-info form-control rounded mt-4 mb-1 fs-5 fw-bold text-white" />
55               <a className="text-info fw-bold text-decoration-none p-2" href="/register">Not a member? Sign up Here!</a>
56             </form>
57             <div>{msg}</div>
58           </div>
59         </div>
60       </div>
61     </div>
62   </div>
63   <div className="col-12 d-flex flex-column justify-content-end align-items-end fw-bold">
64     <Terms />
65   </div>
66 </div>
67 )
68 }
```

XX. Login.js – példa a bootstrap hatékony alkalmazására



Alkalmazásunk **reszponzív**, különböző viewportokon is kényelmesen változtatja a megjelenést, a **grid system alkalmazásával** (fluid grids).

The screenshot shows a portion of the `Home.js` file from a code editor. A specific section of the code is highlighted with a yellow box, demonstrating the use of a grid system for responsive design. The highlighted code is as follows:

```
59     return (
60       <>
61       <div className="homeBackground">
62         <NavBar photos={photos} setPhotosFiltered={setPhotosFiltered} categoryList={categoryList} setCategory={setCategory}
63           selectedCategory={selectedCategory} setSelectedCategory={setSelectedCategory} userName={userName} />
64         <div className="container">
65           <div className="row justify-content-center">
66             <Sidebar userName={userName} />
67             <div className="homeBox col-12 col-lg-8 col-md-8 col-sm-12 mt-5 mb-4 ">
68               <div className="homeBoxContent">
69                 {photosFiltered.map((item, i) => (
70                   <div
71                     key={i}
72                     className="homePictures d-flex d-inline-flex p-2"
73                     onClick={() => handleShowModal(item)}
74                   >
75                     <img
76                       className="homePicture"
77                       src={item.link}
78                       alt={item.title}
79                       loading="lazy"
80                     />
81                     <div className="homePictureTitle">{item.title}</div>
82                   </div>
83                 )))
84               </div>
85             </div>
86             <div className="col-12 d-flex flex-column justify-content-end align-items-end fw-bold">
87               <Terms />
88             </div>
89           </div>
90         </div>
91       </div>
92     </div>
93   </div>
94   <div className="col-12 d-flex flex-column justify-content-end align-items-end fw-bold">
95     You, 4 days ago • design ...
96   </div>
97 </div>
```

XXI. `Home.js` - Grid megvalósulás szabályozza a reszponzív megjelenést

Ezt a technikát alkalmaztuk például az admin felületen is, amely lehetővé tette az adatok elosztott – és reszponzív - megjelenését.

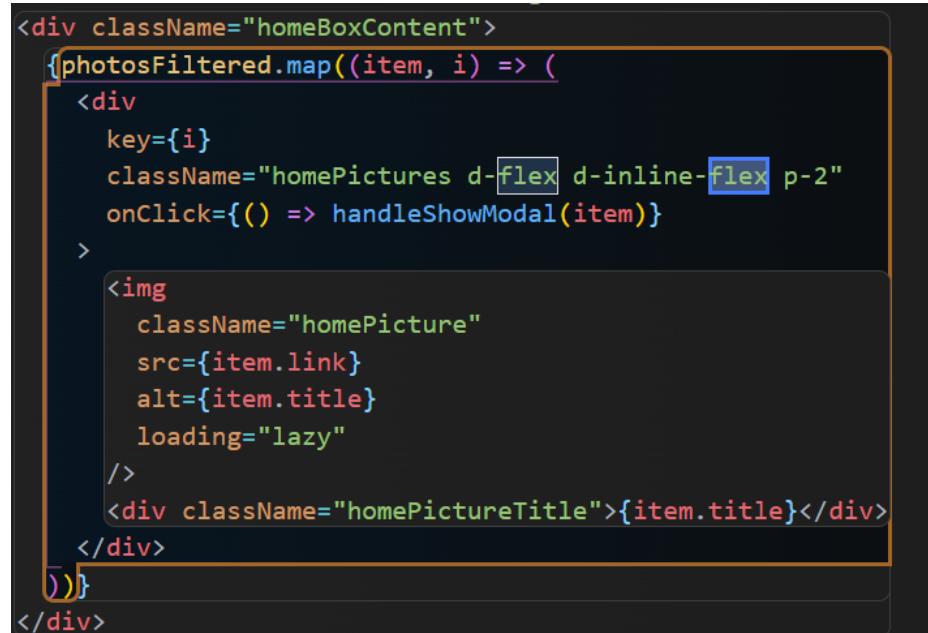
The screenshot shows a portion of the `Admin.js` file from a code editor. A specific section of the code is highlighted with a yellow box, demonstrating the use of a grid system for managing data in a table. The highlighted code is as follows:

```
33     return (
34       <div className="homeBackground">
35         <div className="container">
36           <NavLink to="/" className="nav-link" aria-current="page" href="#">&lt;&gt;</NavLink>
37           <div className="row justify-content-center">
38             <div className="adminBox col-12 mt-5">
39               <div className="adminContent">
40                 <h1 className="text-center">Users</h1>
41                 <table className="table table-striped text-center">
42                   <thead>
43                     <tr>
44                       <th className="adminNav col-1" scope="col">userid</th>
45                       <th className="adminNav col-2" scope="col">username</th>
46                       <th className="adminNav col-2" scope="col">email</th>
47                       <th className="adminNav col-2" scope="col">created at</th>
48                       <th className="adminNav col-2" scope="col">updated at</th>
49                       <th className="adminNav col-2" scope="col">role</th>
50                       <th className="adminNav col-1" scope="col">delete</th>
51                     </tr>
52                   </thead>
53                   <tbody>
54                     {list.map((item, index) => <tr key={index}>
55                       <td className="align-middle">{item.userid}</td>
56                       <td className="align-middle">{item.username}</td>
57                       <td className="align-middle">{item.email}</td>
58                       <td className="align-middle">{item.created_at}</td>
59                       <td className="align-middle">{item.updated_at}</td>
60                       <td className="align-middle">{item.role}</td>
61                       <td onClick={()=>handleDelete(item)} className="deleteUser text-align-center align-middle text-light bg-danger">
62                         &lt;&gt;&lt;&gt;
63                       </td>
64                     </tr>
65                   </tbody>
66                 </table>
67               </div>
68             </div>
69           </div>
70         </div>
71       </div>
72     </div>
73   </div>
74   <div className="col-12 d-flex flex-column justify-content-end align-items-end fw-bold">
75     You, 4 days ago • design ...
76   </div>
77 </div>
```

XXII. `Admin.js` Griddel szabályoztuk az oszlopok mefjelenését



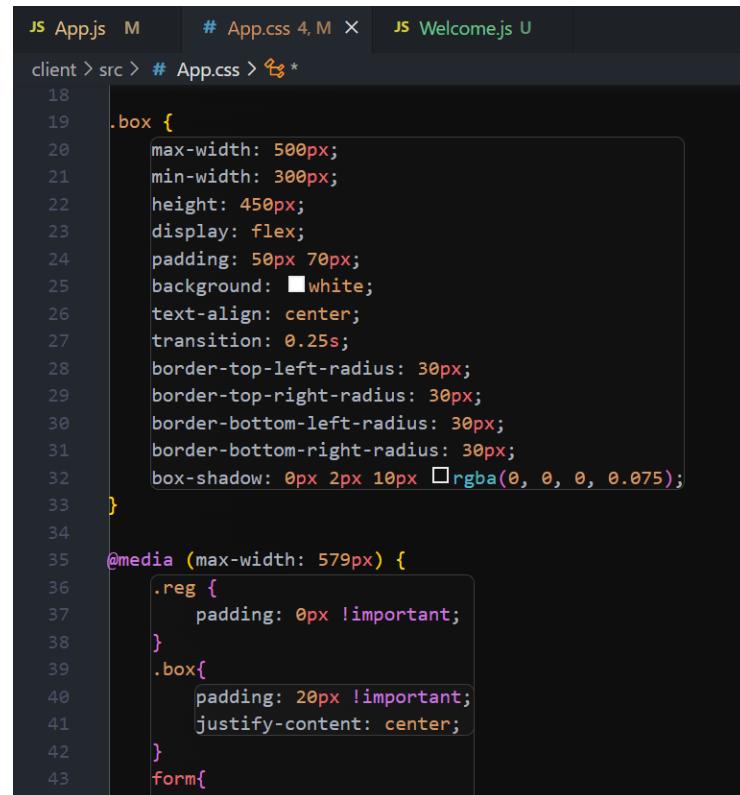
Ennek hatékonyságát kiegészítik a további flex megjelenítések. Így például a felhasználói élmény javítása érdekében a képek rugalmas megjelenésére is figyelmet fordítottunk.



```
<div className="homeBoxContent">
  {photosFiltered.map((item, i) => (
    <div
      key={i}
      className="homePictures d-flex d-inline-flex p-2"
      onClick={() => handleShowModal(item)}
    >
      <img
        className="homePicture"
        src={item.link}
        alt={item.title}
        loading="lazy"
      />
      <div className="homePictureTitle">{item.title}</div>
    </div>
  ))}
</div>
```

XXIII. Flex megjelenés szabályozza a reszponzivitást

A reszponzivitás érdekében ahol kellett, **media query-ket** is alkalmaztunk.



```
JS App.js M # App.css 4, M X JS Welcome.js U
client > src > # App.css > 43 *
18
19 .box {
20   max-width: 500px;
21   min-width: 300px;
22   height: 450px;
23   display: flex;
24   padding: 50px 70px;
25   background: white;
26   text-align: center;
27   transition: 0.25s;
28   border-top-left-radius: 30px;
29   border-top-right-radius: 30px;
30   border-bottom-left-radius: 30px;
31   border-bottom-right-radius: 30px;
32   box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.075);
33 }
34
35 @media (max-width: 579px) {
36   .reg {
37     padding: 0px !important;
38   }
39   .box{
40     padding: 20px !important;
41     justify-content: center;
42   }
43   form{
```

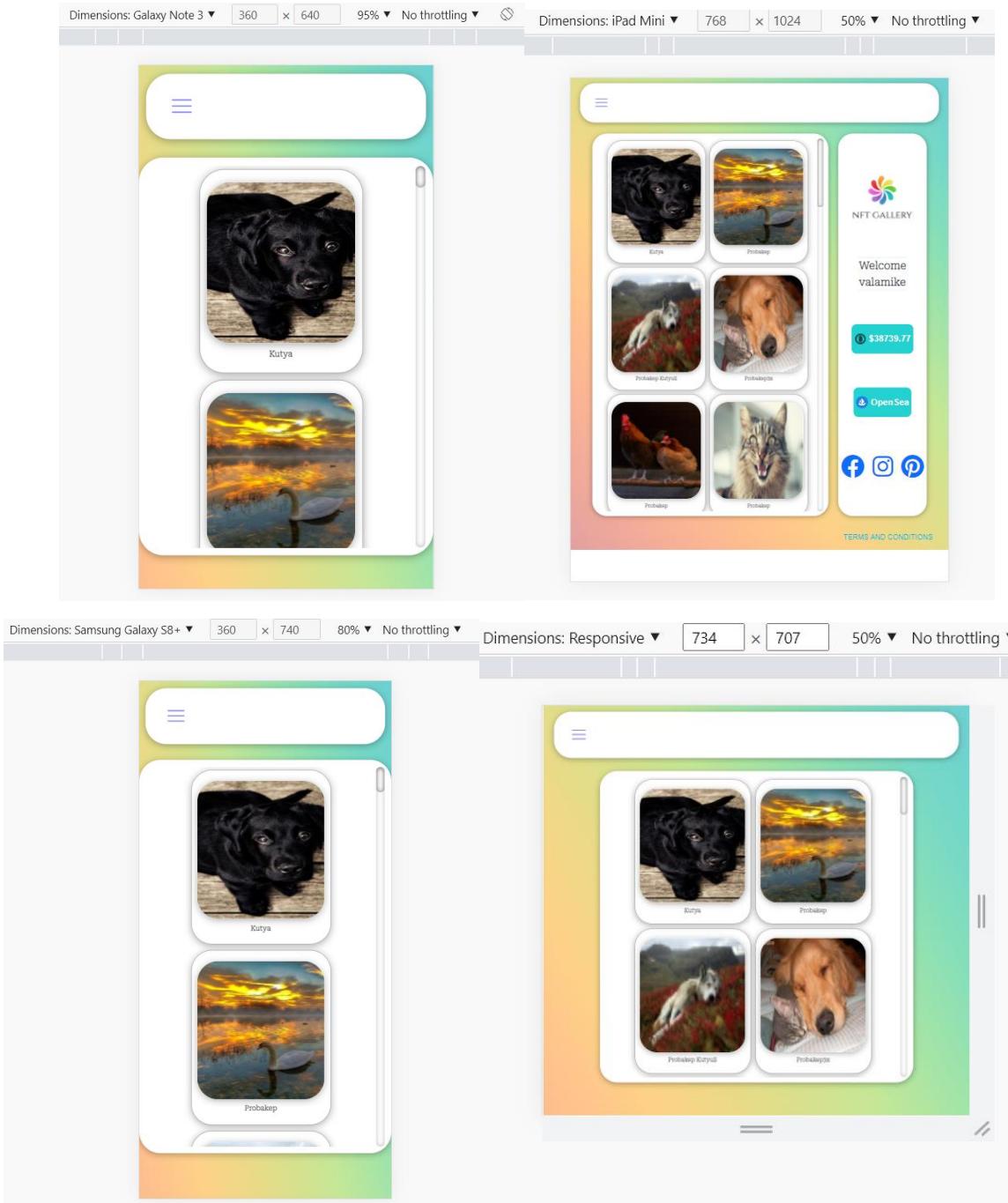
XXIV. App.css - media query alkalmazása a regisztráció oldalán



NFT GALLERY

A fent nevezett megoldások (grid, flex, media query, bootstrap) párhuzamos használatával tökéletesen reszponzív oldalmegvalósítást értünk el.

A **reszponzív tervezésnek** köszönhetően a felhasználó bármilyen készülékről képes **mobil vagy asztali eszközről** egyaránt csatlakozni saját, egyedi rendszeréhez. A felhasználói élmény nem sérül a különböző nézetek, képernyőméretek mellett.



XXV. Home megjelenése különböző viewportokon



A fejlesztés során JavaScript megoldásokat és könyvtárakat egyaránt használtunk.

```

EXPLORER      ... JS App.js
OPEN EDITORS   client > src > JS App.js
JS App.js client\src
JS App.js
JS NavBar.css
JS NavBar.jsx
SearchBar
JS SearchBar.css
JS SearchBar.jsx
Sidebar
JS Sidebar.css
JS Sidebar.jsx
JS Sidebar.test.js
JS Delete.jsx
JS Edit.jsx
JS Forgotten.jsx
JS Home.jsx
JS Login.jsx
JS Logout.jsx
JS Register.jsx
JS Terms.jsx
JS Upload.jsx
JS App.css
JS App.js
index.css
JS index.js

```

```

1 import React, { useEffect, useState } from 'react';
2 import { BrowserRouter, Routes, Route, useForm } from 'react-router-dom';
3 import '../node_modules/bootstrap/dist/css/bootstrap.min.css';
4 import '../node_modules/bootstrap/dist/js/bootstrap.bundle.js';
5 import './App.css';
6 import { Register } from './components/Register';
7 import { Login } from './components/Login';
8 import { Forgotten } from './components/Forgotten';
9 import { Home } from './components/Home';
10 import { Upload } from './components/Upload';
11 import { Admin } from './components/Admin/Admin';
12 import { Logout } from './components/Logout';
13 import { Edit } from './components/Edit';
14 import axios from 'axios';
15 import { useNavigate } from 'react-router-dom';
16
17 function App() {
18     const [user, setUser] = useState(true)
19     const [categoryList, setCategoryList] = useState([])
20     const [selectedCategory, setSelectedCategory] = useState(0)
21     const [userName, setUserName] = useState(localStorage.getItem('userName') ? localStorage['userName'] : '');
22     const [userId, setUserId] = useState(localStorage.getItem('userId') ? localStorage['userId'] : 0);
23
24     useEffect(() => {
25         fetchCategoryList();
26     }, [()]);
27 }

```

XXVI. App.js – példa a JavaScript könyvtárak importálására, illetve egy JavaScript függvényre, változókra

Komponenseink megírása során alkalmaztunk **horgokat (HOOKS)**, melyek által használhatjuk a React funkcióit osztályok írása nélkül. A horgok használatának további előnyei, hogy a **komponensek rövidebbek, és könnyebb állapotteljes logikát megosztani**, illetve újra felhasználni a komponensek között.

```

JS Home.jsx
client > src > components > JS Home.jsx > Home
1 import React, { useEffect, useState, useReducer } from "react";
2 import { motion } from "framer-motion";
3 import axios from "axios";
4 import { Modal } from "react-bootstrap";
5 import { Sidebar } from "./Sidebar/Sidebar";
6 import { NavBar } from "./NavBar/NavBar";
7 import { Terms } from "./Terms";
8
9 export const Home = ({ categoryList, setCategory, selectedCategory, setSelectedCategory, userName, userId }) => {
10     const [photo, setPhoto] = useState({});
11     const [photos, setPhotos] = useState([]);
12     const [showModal, setShowModal] = useState(false);
13     const [showInfo, setShowInfo] = useState(false);
14     const [photosFiltered, setPhotosFiltered] = useState([photos])
15
16
17     const handleCloseModal = () => setShowModal(false);
18     const handleShowModal = (item) => {
19         setPhoto(item);
20         setShowModal(true);
21         setShowInfo(false);
22     };
23
24
25     useEffect(() => {
26         setPhotosFiltered(photos);
27     }, [photos]);
28
29     useEffect(() => {
30         fetchPhotos(selectedCategory);
31     }, [selectedCategory]);
32
33     const fetchPhotos = async (selectedCategory) => {
34         console.log('home:', userId)
35         let url = selectedCategory == 0
36             ? `http://localhost:5000/photos/${userId}`
37             : `http://localhost:5000/photos/${userId}/categ/` + selectedCategory;
38         try {
39             const resp = await axios.get(url);
40             setPhotos(resp.data);
41         } catch (err) {
42             console.error(err);
43         }
44     };

```

XXVII. Home.js – példa a useState , useEffect használatára, illetve a fetchelésre



Alkalmazásunk használja a következő horgokat is: **useEffect**, **useReducer**, **useRef**, **useState**.

```
JS Terms.jsx  X
client > src > components > JS Terms.jsx > ...
1 import React, {useEffect, useForm, useRef, useState} from 'react';
2 import Button from '@mui/material/Button';
3 import Dialog from '@mui/material/Dialog';
4 import DialogActions from '@mui/material/DialogActions';
5 import DialogContent from '@mui/material/DialogContent';
6 import DialogContentText from '@mui/material/DialogContentText';
7 importDialogTitle from '@mui/material/DialogTitle';
8
9
10 export const Terms = () => {
11   const [open, setOpen] = useState(false);
12   const [scroll, setScroll] = useState('paper');
13
14   const handleClickOpen = (scrollType) => () => {
15     setOpen(true);
16     setScroll(scrollType);
17   };
18
19   const handleClose = () => {
20     setOpen(false);
21   };
22
23   const descriptionElementRef = useRef(null);
24   useEffect(() => {
25     if (open) {
26       const { current: descriptionElement } = descriptionElementRef;
27       if (descriptionElement !== null) {
28         descriptionElement.focus();
29       }
30     }
31   }, [open]);
32
33   return (
34     <div className="row pb-2">
35       <Button style={{color: '#00bcd4'}} onClick={handleClickOpen('paper')}>Terms and Conditions</Button>
36     </div>
37   );
38 }
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
```

XXVIII. Terms.js – további példa a useState , useEffect, illetve a useRef használatára

Illetve egyedi horgokat is használtunk, mint a **useForm** és **useNavigate**.

```
JS Logout.js  X
client > src > components > JS Logout.js > ...
impetusalahurr1, 4 days ago | 2 authors (You and others)
1 import React, { useEffect } from 'react'
2 import { Login } from './Login'
3 import { [useNavigate] } from 'react-router-dom';      You, las
4
5 export const Logout = ({setUser, setUserName, setId}) => {
6
7   const navigate = [useNavigate()];
8
9   useEffect(() => {
10     setUser(false)
11     setUserName('')
12     setId(0)
13     localStorage.removeItem('user')
14     localStorage.removeItem('userName')
15     localStorage.removeItem('userId')
16     navigate('/login');
17   }, []);
18
19   return (
20     <>
21     </>
22   )
23 }
```

XXIX. Logout.js –példa a useNavigate használatára



```

JS Register.jsx ×
client > src > components > JS Register.jsx > [Θ] Register
Lili, 5 days ago | 2 authors (Lili and others)
1 import axios from 'axios';
2 import React,{ useEffect, useState} from 'react'
3 import { useForm } from 'react-hook-form';
4 import {Terms} from './Terms';
5
6
7 export const Register=()=>{
8     const [successful,setSuccessful]=useState(false)
9     const [msg,setMsg]=useState('')
10    const [validUsername,setValidUsername]=useState(true)
11    const [validEmail,setValidEmail]=useState(true)
12
13    useEffect(()=>{
14        setMsg('')
15        if(!validUsername)
16            setMsg('username not available!')
17        if(!validEmail)
18            setMsg(msg+' email address already in use!')
19    },[validUsername,validEmail])
20
21    const {
22        register,
23        handleSubmit,
24        formState: { errors },
25    } = useForm();           Lili, 2 weeks ago * log, reg update ...
26    const onSubmit = (data) => {
27        console.log(data);
28        const url='/authRoute/register'
29        sendData(url,data)
    }
}

```

XXX. Register.js – példa a useForm használatára

Kódunk RESTful architektúrának megfelelő kliens oldali komponenseket tartalmaz.

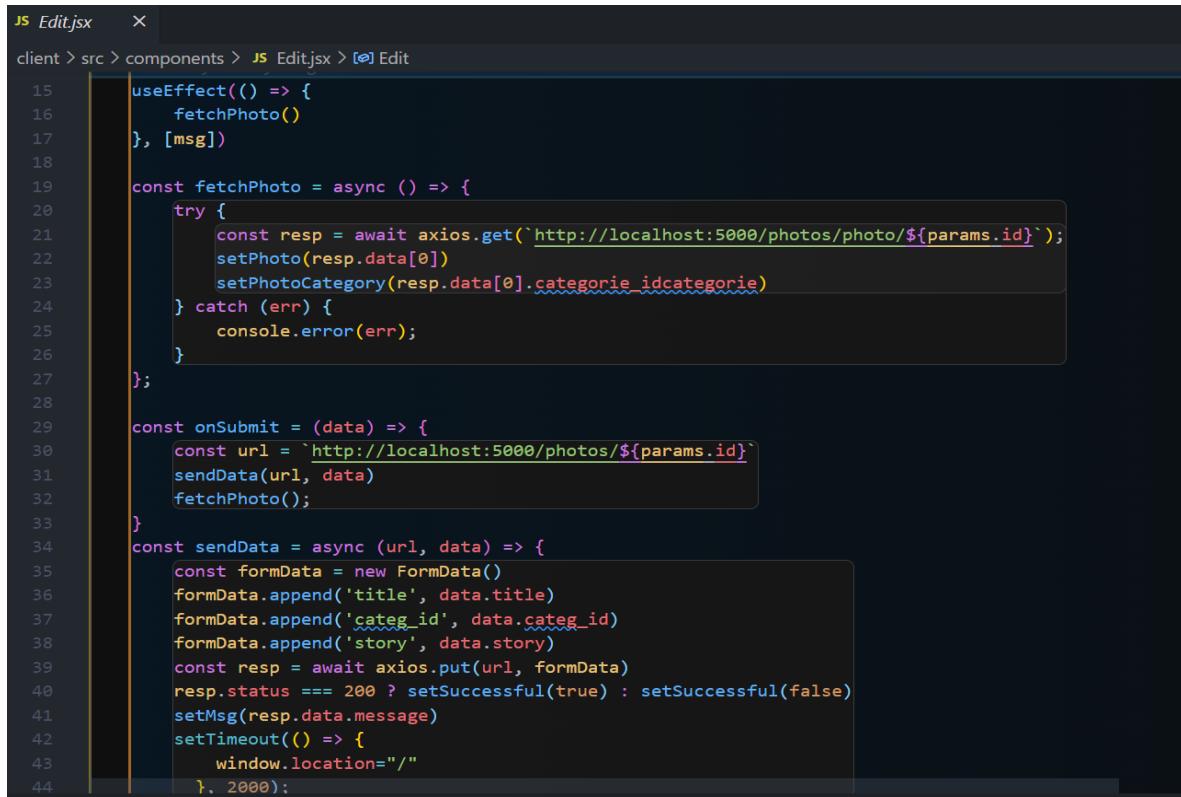
```

EXPLORER      ...
JS Edit.jsx ×
client > src > components > JS Edit.jsx > [Θ] Edit
Lili, yesterday | 3 authors (Lili and others)
1 import React, { useState, useEffect } from 'react'
2 import { useForm } from 'react-hook-form';
3 import axios from 'axios';
4 import { NavLink, useParams } from 'react-router-dom';
5 import { Terms } from './Terms';
6
7 export const Edit = ({ userId, categoryList }) => {
8     const params = useParams()
9     const { register, handleSubmit, formState: { errors } } = useForm();
10    const [successful, setSuccessful] = useState(false)
11    const [msg, setMsg] = useState('')
12    const [photo, setPhoto] = useState({})
13    const [photoCategory, setPhotoCategory] = useState(0)
14
15    useEffect(() => {
16        fetchPhoto()
17    }, [msg])
18
19    const fetchPhoto = async () => {
20        try {
21            const resp = await axios.get(`http://localhost:5000/photos/photo/${params.id}`);
22            setPhoto(resp.data[0])
23            setPhotoCategory(resp.data[0].categorie_idcategorie)
24        } catch (err) {
25            console.error(err);
26        }
27    };
28
29    const onSubmit = (data) => {
    }
}

```

XXXI. Edit.js – példa az axios használatra

Axiosokkal kapcsoltuk össze a kliens oldalt a szerver oldallal és kértünk le, illetve mentettünk fel adatokat.



```
JS Edit.jsx x
client > src > components > JS Edit.jsx > [e] Edit
15  useEffect(() => {
16    fetchPhoto()
17  }, [msg])
18
19  const fetchPhoto = async () => {
20    try {
21      const resp = await axios.get(`http://localhost:5000/photos/photo/${params.id}`);
22      setPhoto(resp.data[0])
23      setPhotoCategory(resp.data[0].categorie_idcategorie)
24    } catch (err) {
25      console.error(err);
26    }
27
28
29  const onSubmit = (data) => {
30    const url = `http://localhost:5000/photos/${params.id}`
31    sendData(url, data)
32    fetchPhoto();
33  }
34  const sendData = async (url, data) => {
35    const formData = new FormData()
36    formData.append('title', data.title)
37    formData.append('categ_id', data.categ_id)
38    formData.append('story', data.story)
39    const resp = await axios.put(url, formData)
40    resp.status === 200 ? setSuccessful(true) : setSuccessful(false)
41    setMsg(resp.data.message)
42    setTimeout(() => {
43      window.location="/"
44    }, 2000);

```

XXXII. Edit.js – adatküldés a server oldalra

Szerver oldali komponensek (backend)

RESTful architektúrának megfelelő szerver oldali komponenseket tartalmaz.

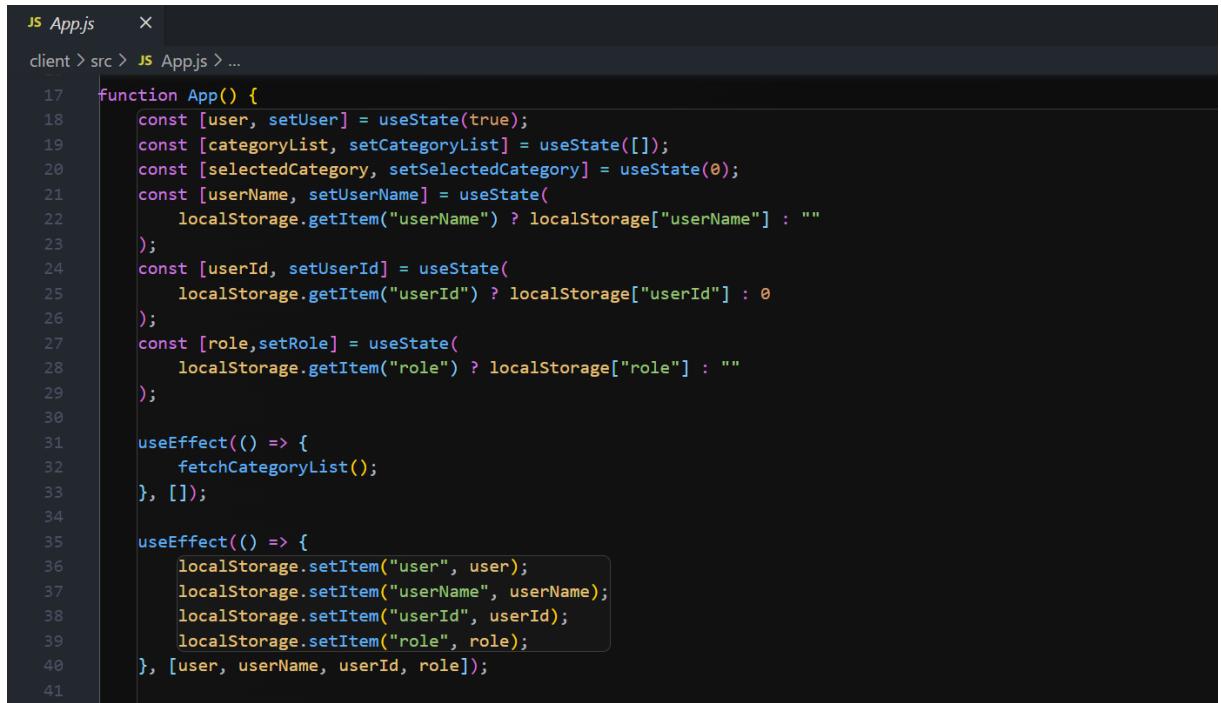
Adattárolási funkciókat megvalósít. Adatbázisba ment adatokat. Ez lehet, bármilyen felhasználói adat bevitel vagy regisztráció.

Adatkezelési funkciókat megvalósít. Adatbázis tartalmat megjelenít.

Adatkezelési funkciókat megvalósít. A felhasználok egyedi azonosítására alkalmas elemeket is tartalmaz.

A felhasználók számára később felhasználható adatokat adatbázisban tárolja. (pl: munkamenet azonosítók használata)

A munkamenet azonosítására Localstorege-et használtunk.



```
JS App.js X
client > src > JS App.js > ...
17 function App() {
18     const [user, setUser] = useState(true);
19     const [categoryList, setCategoryList] = useState([]);
20     const [selectedCategory, setSelectedCategory] = useState(0);
21     const [userName, setUserName] = useState(
22         localStorage.getItem("userName") ? localStorage["userName"] : ""
23     );
24     const [userId, setId] = useState(
25         localStorage.getItem("userId") ? localStorage["userId"] : 0
26     );
27     const [role, setRole] = useState(
28         localStorage.getItem("role") ? localStorage["role"] : ""
29     );
30
31     useEffect(() => {
32         fetchCategoryList();
33     }, []);
34
35     useEffect(() => {
36         localStorage.setItem("user", user);
37         localStorage.setItem("userName", userName);
38         localStorage.setItem("userId", userId);
39         localStorage.setItem("role", role);
40     }, [user, userName, userId, role]);
41 }
```

XXXIII. App.js – A munkamenet azonosítására Localstorege-et használtunk.

Megfelelően használja az MVC elv Modell és Controller közötti adatcserét.

Törekedett a biztonságos adatbevitelre, adatbázisban a jelszavakat titkosítva tárolja.

Használta a JavaScript alapú rendszerek bármelyikét a Backend programozásban.

Model - View - Controller

A tervezés során fontos volt, hogy a szerkezeti elem kitalálásakor az MVC mintát alkalmaztunk, ami nem más jelent, mint Model - View - Controller. A model az adatstruktúrát jelenti, a view a nézetet, megjelenítési elemeket és a Controller pedig a vezérlőt, ami összeköti ezeket egy egységé és irányítja őket.

Van a kliensünk, aki/ami valamilyen kérést küld a szerverünknek. Ezt a Controller kapja meg a Routeren keresztül, ezután a Controller, szépen bekéri az adatokat a Modeltől ami adott esetben egy adatbázissal kommunikál, majd pedig a Controller átadja ezeket a Viewnak és a View pedig elkészíti a kérésre feldolgozásának megjelenítését.



Többrétegű architektúra

A rendszereket melyek adat vezéreltek a modern kor irányelvai alapján többrétegű architektúra alapján építjük fel.

Megkülönböztetjük három fő réteget vagy másnéven komponenssaládját:

- Megjelenési réteg
- Üzleti logika rétegű
- Adatelérési réteg

III. Rendszerünk felépítése – Backend

A felhasználók adatait adatbázisban tároljuk, mivel kellőképpen biztonságos, hogy megőrizze számunkra az üzleti logikánkat, azáltal, hogy illetéktelenek nem férhetnek hozzá az engedélyünk nélkül.

Adatbázisunk minden szükséges információt tartalmaz, ami a rendszerünk működéséhez elengedhetetlen, egyszerűségének köszönhetően a későbbiekben kiegészíthető, átalakítható.

IV. Adatbázisunk felépítése

USER tábla

Elsődleges kulcsként az **iduser** vettük igénybe. Auto increment tulajdonságával minden felhasználónk egyedi azonosítót fog kapni.

A **felhasználó nevét, jelszavát és email címét** is ebben a táblában tároljuk el, mivel külön entitásként tekintünk egy-egy felhasználóra. Létrehoztunk egy **created_at** és egy **update_at** mezőt. Míg előbbi a profillétrehozás dátumát tárolja addig utóbbi az esetleges módosítások dátumát. Az **avatar** mező a lehetséges profilkép eltárolására szolgál, ha a felhasználó élni szeretne a profilkép beállítási lehetőségével. A **status** képes arra, hogy eltárolja a felhasználónkról, hogy aktív tag-e vagy pedig függőben lévő vagyis pending.



A **confirmationcode** mezőnk a maga varchar 255-ös tulajdonságával képes eltárolni egy myToken nevezetű változónk generált értékét, ami nullától egészen az abc végéig legenerál egy random értéket 25 karakterből.

```
JS token.js  X
server > models > encryption > JS token.js > ...
impetusalahurr1, 3 weeks ago | 1 author (impetusalahurr1)
1 const characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
2
3 const myToken=()=>{
4     let token = '';
5     for (let i = 0; i < 25; i++)
6         token += characters[Math.floor(Math.random() * characters.length )];
7     return token
8 }
9
10 module.exports={myToken}
```

XXXIV. *token.js – confirmationcode - myToken*

A blokklánc tulajdonságai miatt vezettük be a 255-ös értéket, hogy a későbbiekben ne legyen szükség változtatásra. A felhasználónk szerepkörét (user, admin) pedig a role mezőben tároljuk el. A későbbiekben tovább szeretnénk fejleszteni admin-superadmin lehetőségekre, hogy teljes egészében kiaknázzuk webalkalmazásunk lehetőségeit.

Az username és az email mezőket el kellett lássuk UNIQUE tulajdonsággal, mivel el szerettük volna kerülni, hogy a felhasználók regisztrációjánál összeütközés keletkezzen.

IMAGE tábla

Image táblánkba kerülnek eltárolásra a **felhasználók által feltöltött digitális tartalmak**. Természetesen ennek is elsődleges kulcsa egy **idimage** mező auto incrementtel. A **title** és **description** mezők a felhasználó által kitöltött digitális tartalom információjának a tárolására szolgálnak. Ezt későbbiekben a felhasználó természetesen tudja módosítani. A **link** mezőben tároljuk a clouddinaryra irányított elérési útvonalat. Üzleti logikánkban fontos szerepet játszott, hogy egy felhasználónak több képe is lehet, de egy adott digitális tartalomnak kizárolag egy felhasználója van. Ezért létrehoztunk egy kapcsoló kulcsot az image táblánkba aminek user_iduser nevet adtuk.



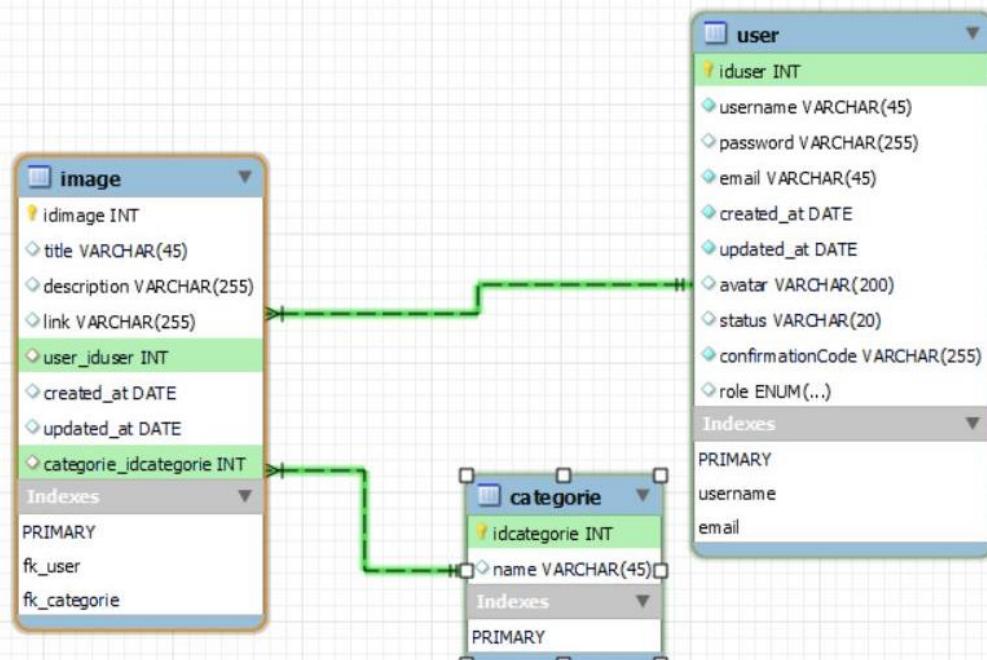
Létrehoztunk továbbá egy **created_at** és **updated_at** mezőt ami DATE típust hordoz magában. Erre azért volt szükség, mert el szerettük volna tárolni, hogy mikor lett létrehozva / feltöltve az adott tartalom és mikor volt frissítve. Legutoljára pedig fellelhető egy **categorie_idcategorie** kapcsoló kulcsunk, ami a categorie táblánkkal van szoros együttműködésben azáltal, hogy 1-N kapcsolattal vannak összepárosítva. Erre azért volt szükség, mert fontos szempont, hogy 1 kategória több képet foglalhat magában.

CATEGORIE tábla

A categorie táblánk minden összesen kettő mezőt tud magáénak. Az első mező az **idcategorie**, ami auto_incremente tulajdonságával képes az beavatkozás nélküli inkrementálásra. Továbbá ez a mező van szoros kapcsolatban az IMAGE táblában található categorie_idcategorie kapcsoló kulcsra a fentebb leírtak miatt (lásd 1-N kapcsolat). Szükségünk volt még egy **name** mezőre, mivel mindenkorban lehetőséget kellett találnunk arra, hogy eltároljuk a kategóriák nevét.

V. Adatbázis tervezés és megvalósítás

Adatbázis tervezésünkönél és megvalósításunknál fontos szerepet játszott, hogy minél egyszerűbben és átláthatóbban alkossuk meg adatbázisunkat. Biztosak voltunk benne, hogy legalább 3 entitásra szükségünk lesz, mivel külön szeretnénk őket kezelni és így a későbbiekbén is bármikor hozzá tudunk implementálni még több entitás típust amint szükség lesz rá és bővíteni szeretnénk webalkalmazásunkat.



XXXV. Tervezés - az UML diagram megfelel az adatbázis végső struktúrájának

A képen látható **UML Diagrammot** a MySQL Workbench által hoztuk létre, mivel a program egyszerűen kezelhető lehetőséget szolgáltat arra, hogy a fejlesztő lokális hálózaton lévő, vagy a mi esetünkben már felhőben lévő adatbázis kieportált .sql kiterjesztésű fájllából kinyerjük a nekünk szükséges EER vagy más néven UML diagrammot, ami könnyebb eligazodást szolgáltat egy fejlesztőnek, mint egy dump fájl.

Megvalósításnál igénybe vettük a PHPMyAdmin nyújtotta lehetőségeket és a MySQL Workbench által szolgáltatott opciókat. Mindaddig, míg lokális körülmények között volt szükség adatbázis lekérdezésekre a PHPMyAdmin felületét vettük igénybe, mivel úgy láttuk egyszerű, átlátható felülete lehetővé tette a gyors adatbázis fejlesztést és Konzol felülete lehetőséget adott arra, hogy gyors lekérdezésekkel ellenőrizhessük a létrehozott kapcsolatok élesben való működését. Amint felkerült adatbázisunk a Heroku által szolgáltatott ingyenes JAWS DB adatbázis felhőbe, jobbnak és egyszerűbbnek véltük használni a Workbench által nyújtott lehetőségeket. Így képesek voltunk arra, hogy egy projekten belül két adatbázis kezelő szoftverre ismerkedjünk meg mélyebben.



VI. A szoftver tesztelése

Szoftverünk tesztelésére – a manuális tesztelés mellett - több eszközt is alkalmaztunk, a hatékony ellenőrzés érdekében.

Unit tesztek

Unit teszteket írtunk a frontend oldalára.

```
JS Sidebar.js M JS Sidebar.test.js U
client > src > components > Sidebar > JS Sidebar.js > Sidebar
  act from "react";
  logo from "../images/myLogo.png";
  bitcoinPrice from "../BitcoinPrice/BitcoinPrice";
  openSea from "../BitcoinPrice/OpenSea";
  /sidebar.css"
  ...
  const Sidebar = ({userName}) => {
    ...
    return (
      ...
      

...
        ![AVATAR]({myLogo})
        Welcome {userName}
        ...
        fa-brands fa-facebook


    );
  };
  ...
  export default Sidebar;
```

```
PASS | src/components/sidebar/Sidebar.test.js
Sidebar component
  ✓ should render (149 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        4.329 s
Ran all test suites related to changed files.
```

```
PS D:\NFTGallery\server> nodemon
[nodemon] 2.0.15
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching path(s): "*"
[nodemon] watching extensions: js,mjs,json
[nodemon] starting 'node index.js'
Server running on port 5000
```

XXXVI. Sidebar.js –UNIT teszt meghívása

```
JS Sidebar.js M JS Sidebar.test.js U
client > src > components > Sidebar > JS Sidebar.test.js > describe('Sidebar component') callback > it('should render') callback
  import {render} from '@testing-library/react';
  import {Sidebar} from './Sidebar';

  const userName = 'test';
  describe('Sidebar component', () => {
    it('should render', () => {
      const getByTestId = render();
      const sidebarComponent = getByTestId('sidebar');
      expect(sidebarComponent).toBeInTheDocument();
    });
  });
  ...
}

  ...
  export default Sidebar;
```

```
PASS | src/components/sidebar/Sidebar.test.js
Sidebar component
  ✓ should render (149 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        4.366 s, estimated 5 s
Ran all test suites related to changed files.

Watch Usage
  > Press a to run all tests.
  > Press f to run only failed tests.
  > Press q to quit watch mode.
  > Press p to filter by a filename regex pattern.
  > Press t to filter by a test name regex pattern.
  > Press Enter to trigger a test run.
```

```
PS D:\NFTGallery\server> cd server
PS D:\NFTGallery\server> nodemon
[nodemon] 2.0.15
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching path(s): "*"
[nodemon] watching extensions: js,mjs,json
[nodemon] starting 'node index.js'
Server running on port 5000
```

XXXVII. - Sidebar.test.js – Kliens oldali sikeres UNIT teszt



A backend tesztelésére többek között a Postmant is használtuk.

The screenshot shows the Postman interface with a successful GET request to `http://localhost:5000/category`. The response body is a JSON array containing four categories:

```
1 [ { 2   "idcategorie": 3, 3     "name": "Animals" 4 }, 5   { 6     "idcategorie": 4, 7       "name": "Family" 8 }, 9   { 10    "idcategorie": 2, 11      "name": "Friends" 12 }, 13   { 14     "idcategorie": 1, 15       "name": "Nature" 16 } ]
```

XXXVIII. - Sidebar.test.js – Kliens oldali sikeres UNIT teszt

Az egész fejlesztés során szükségünk volt arra, hogy teszteljük a szerver és az alkalmazás kapcsolatát, működését. Ezért a **Postman** programot hívtuk segítségül. A program kiváló lehetőséget nyújt arra, hogy teszteljük az API-nkat. A tesztelés menete egyszerűen zajlott: kérést küldtünk a szerver felé, ehhez szükségünk volt arra, hogy a Postmanban beállítsuk a kérés típusát. (A lentebb látható képen egy GET kérés látható a /category route-ra) Amint megérkezett a szerver felé küldött adat a szerver képes volt feldolgozni és a Postman visszaszolgáltatta számunkra JSON Web Token formájában a tökéletes végeredményt.

VII. Publikáció

Projektünket a GITHUB-on (<https://github.com/zsanber/NFTGallery>), illetve a herokun (<https://nftgalleryforprojekt.herokuapp.com/>) pubikáltuk és tettük elérhetővé. (Bővebben fentebb: „A csapatmunka megvalósítása” fejezetben).



VIII. További fejlesztési lehetőségek

Rengeteg további fejlesztési lehetőséget látunk a projektben. Terveink szerint projektünk későbbi verziójában az elérhető funkciók, fejlesztési pontok a következők:

Mindenkorai terveink között szerepel, hogy projektünk képes legyen egy teljesen zárt, **megbízható rendszert alkotni** az arra vágyóknak, ahol a felhasználók képesek **biztonságosan tárolni** digitális értékeiket – ehhez szükséges, hogy minden, a felhasználók által elért és esetleg jelenleg még manipulálható tartalom a serveren legyen. Fontos, hogy a server lekérések megfelelően korlátozottak legyenek.

A képek tárolását **később biztonságos, belső serveren** oldanánk meg.

Webalkalmazásunk további legfontosabb fejlesztése lehet, hogy **biztonságtechnikai szempontból a későbbiekben képes legyen kezelni a fejlett blokklánc technológiát** is.

A későbbiekben tovább szeretnénk fejleszteni a jelenlegi user-admin jogosultsági köröket **user-admin-superadmin lehetőségekre**, hogy teljes egészében kiaknázzuk webalkalmazásunk lehetőségeit.

További kriptovaluta árfolyamok megtekintését is beépítenénk a projektbe.

Jelenleg webalkalmazásunk témaja és színvilága világos, emellett **sötét módban is elérhetővé szeretnénk tenni.**

A felhasználónak **Google bejelentkezési lehetőséget is biztosítani szeretnénk.**

A **design további letisztítása, finomítása**, akár különböző színvilágú nézet-lehetőségek beállítása is folyamatban van.



Felhasznált irodalom

- <https://artsandculture.google.com/entity/m0134xwrk?hl=hu> (Megtekintve: 2022. 04. 15.)
- <https://hu.wikipedia.org/wiki/XAMPP> (Megtekintve: 2022. 04. 15.)
- <https://www.mysql.com/products/workbench/> (Megtekintve: 2022. 04. 15.)
- <https://www.phpmyadmin.net/> (Megtekintve: 2022. 04. 15.)
- <https://nodejs.org/en/> (Megtekintve: 2022. 04. 25.)
- <https://hu.wikipedia.org/wiki/Node.js> (Megtekintve: 2022. 04. 25.)
- <https://www.npmjs.com/> (Megtekintve: 2022. 04. 15.)
- <https://reactjs.org/> (Megtekintve: 2022. 04. 12.)
- <https://www.postman.com/> (Megtekintve: 2022. 04. 20.)
- <https://www.jawsdb.com/> (Megtekintve: 2022. 04. 15.)
- <https://postmarkapp.com/send-email/node> Megtekintve: 2022. 05.01)
- <https://dashboard.heroku.com/> (Megtekintve: 2022. 04. 15.)
- <https://en.wikipedia.org/wiki/Heroku> (Megtekintve: 2022. 04. 15.)
- <https://cloudinary.com/> (Megtekintve: 2022. 04. 27.)
- KÁTAY MAGDOLNA tanárőr órai prezentációi (Megtekintve: 2022. 04. 01.)
- VASTAG ATTILA tanár úr órai prezentációi (Megtekintve: 2022. 04. 01.)

Tartalom

Bevezetés	2
A csapatmunka megvalósítása	4
Témaválasztás	6
Felhasználói dokumentáció	7
I. A program általános specifikációja.....	7
II. Hardverkövetelmények a szervergép számára	7
A backend és szervergép számára ajánlott követelmények	7
Hardverkövetelmények a kliens oldal számára	7
III. A program telepítése.....	8
IV. A program használatának részletes leírása	9
Bejelentkezés.....	9
Elfelejtett jelszó	9
Regisztráció.....	10
Kezdőlap	10
Navigációs sáv.....	13
Képfeltöltés.....	15
Fájl/képszerkesztés.....	15
Adminisztrációs felület	16



Fejlesztői dokumentáció 17

I. Alkalmazott fejlesztői eszközök, technológiák és programok 17

Microsoft Visual Studio Code	17
XAMPP	17
MySQL Workbench	18
PHPMyAdmin.....	18
Postman	18
Heroku	19
Cloudinary.....	19
JawsDB	19
Postmark.....	20
React	20
Node.js	20
NPM	21

II. Tervezési szempontok 22

Kliens oldali komponensek (frontend)	23
Szerver oldali komponensek (backend)	31
Model - View - Controller	32
Többrétegű architektúra	33



NFT GALLERY

III. Rendszerünk felépítése – Backend	33
IV. Adatbázisunk felépítése	33
USER tábla.....	33
IMAGE tábla	34
CATEGORIE tábla	35
V. Adatbázis tervezés és megvalósítás	35
VI. A szoftver tesztelése	37
Unit tesztek.....	37
VII. Publikáció	38
VIII. Továbbfejlesztési lehetőségek	39
Felhasznált irodalom.....	40
Tartalom	41



NFT GALLERY