



Webalkalmazás fejlesztése az MNIST adatbázis humán validációjához

Készítette

Vámos Ákos János

Programtervező informatikus BSc

Témavezető

Dr. Tajti Tibor Gábor

egyetemi docens

EGER, 2024

Tartalomjegyzék

Bevezetés	5
1. Bevezetés a statisztikába és az adatok világába	6
1.1. A statisztikáról	6
1.1.1. A statisztika eredete	6
1.1.2. Milyen egy jó statisztikai összefoglalás?	7
1.2. Használt statisztikai módszerek az alkalmazásban	8
1.2.1. Átlag szerepe	8
1.2.2. Maximum és Minimum szerepe	8
1.3. Adatvizualizáció és használt grafikonok az alkalmazásban	8
1.3.1. Táblázat	9
1.3.2. Oszlopdiagram	9
1.3.3. Kördiagram	10
1.3.4. Hőtérkép	11
1.4. Adatok gyűjtése	11
2. Használt technológiák	13
2.1. Tervezéshez használt technológiák	13
2.1.1. PlantUML	13
2.1.2. dbDiagram.io	14
2.2. Megvalósításhoz használt technológiák	15
2.2.1. MySQL	15
2.2.2. HTML és CSS	15
2.2.3. PHP	16
2.2.4. JavaScript	16
2.2.5. JSX	16
2.2.6. Tailwind CSS	17
2.2.7. Chart.js	17
2.2.8. Inertia	17
2.2.9. Laravel	18

3. Felhasználói dokumentáció	19
3.1. Dashboard	19
3.2. Overview	19
3.2.1. Generation Statistics	20
3.2.2. Misidentification Statistics	21
3.2.3. Responses Statistics	22
3.3. Graphs & Charts	22
3.3.1. Responses	23
3.3.2. Image Frequencies	25
3.4. Data Listing	26
3.4.1. Keresés	27
3.4.2. Rendezés	27
3.4.3. Adattörlés	28
3.4.4. Adatok exportálása	28
3.4.5. Kattintható sorok	29
3.4.6. Ugrás a tetejére	29
3.5. Image Generation	30
3.6. Feedbacks	31
4. Fejlesztői dokumentáció	32
4.1. Inicializálás	32
4.2. Adatbázis táblák	32
4.2.1. responses	33
4.2.2. images_frequencies	33
4.2.3. misidentifications	33
4.2.4. number_frequencies	34
4.3. Összefoglaló nézet lekérdezései	34
4.3.1. Components	36
4.4. Graphs & Charts fejlesztői szemmel	37
4.4.1. Oszlopdiaagram létrehozása	37
4.4.2. Hőtérkép megvalósítása	38
4.5. Adatok törlése	40
4.6. Telepítés	41
5. Tesztelés	42
5.1. Cypress tesztelés	42
6. Továbbfejlesztési ötletek	45
6.1. Személyre szabhatóság	45
6.2. Lekérdezések optimalizálása	45

6.3. Terheléses tesztek	45
Összegzés	46
Irodalomjegyzék	47

Bevezetés

Napjainkban az információ áradása rendkívül nagy mértékben történik. Ezen információkon való tájékozódás és értelmezés kihívást tud jelenteni, különösen akkor, ha célunk a releváns adatok megszerzése. Ebből kifolyólag kiemelt szerepet foglalnak a statisztikai kimutatások és felmérések, amelyek segítségével strukturált és felfogható formában tudjuk megközelíteni ezen problémákat.

Manapság egy másik meghatározó jelenség a mesterséges intelligencia és ennek egyik ága a gépi tanulás. Mit értek gépi tanulás alatt? Amikor a gép önmagától tanul újat egy bizonyos információhalmazból. Ehhez viszont először adatok kellenek neki, ráadásul minél többet kap, annál okosabb lesz. A cél tehát ezen információk begyűjtése, majd azok feldolgozása, amelyek felhasználhatóak lesznek a gépi tanulás egy szintjéhez.

A szakdolgozat címében szereplő MNIST adatbázis is egy, a gépi tanításhoz használtak közül. Milyen MNIST? Milyen adatbázis? Hadd válaszoljam meg ezeket a kérdéseket röviden. „Az MNIST adatbázis (kiejtve emniszt adatbázis) kézzel írt számjegyek képét tartalmazó adatbázis”.[1] „A számítástechnikában az adatbázis elektronikusan tárolt és elérhető adatok szervezett gyűjteménye”.[2] Azaz vannak képek amelyek számjegyeket ábrázolnak, vannak köztük könnyebben és nehezebben azonosíthatók.

A szakdolgozat címének „humán validációja” pedig arra utal, hogy az MNIST képekből egy felmérés áll elő, amelyek azonosítását a kitöltőkre bizzuk, majd válaszaik feldolgozásra kerülnek. Valójában a dolgozat témája két nagy részből áll, az emberek válaszainak gyűjtéséből, valamint a válaszok statisztikailag való elemzéséről és feldolgozásáról.

Részemről a hangsúly főként az utóbbin, tehát a válaszok feldolgozásáért felelős részekén van. Bemutatom az én megoldásaimat arról, hogy egyszerű számokból álló válaszokat hogyan dolgoztam fel annak érdekében, hogy később ezekkel további hasznos lépéseket lehessen tenni a gépi tanítás területén. Továbbá a legtöbb megoldás általánosítható más felmérésekből kinyert eredmények összegzésére, illetve azok feldolgozására.

Egy felmérés célja a válaszok összegyűjtése és az azokból megismert következtetések levonása vagy tények kimondása egy végső cél érdekében. Ebben a helyzetben a végső cél a gépi tanítás egy részének hatékonyabbá tétele.

A szakdolgozat további részében kitérek a használt technológiák kiválasztásától a tervezésen át a megvalósításig az összes olyan folyamatra, amelyet hasznosnak véltem.

1. fejezet

Bevezetés a statisztikába és az adatok világába

1.1. A statisztikáról

A statisztika egy olyan tudományos módszertan, illetve tevékenység, amely a valóságot számszerűen összegzi, majd ezt követően elemezni illetve megfigyelni lehet azt. Általánosságban három nagy összetevőről beszélhetünk.

1. **adatok:** olyan információk, amelyeket különböző gazdasági vagy társadalmi jelenségekről gyűjtünk és rögzítünk. Ezeket többféle módon, például számok vagy szöveges adatok formájában fejezzük ki.
2. **módszerek:** azok az eszközök és technikák, amelyeket az adatok feldolgozására és elemzésére alkalmazunk, annak érdekében, hogy információkat vonjunk le belőlük és megértsük a mögöttük álló jelenségeket. Ezen eszközök algoritmusokat és modelleket foglalnak magukba.
3. **statisztikai rendszer:** az adatok gyűjtésétől kezdve a közzétételig terjedő tevékenységek összessége, amelyek segítik az adatok definiálását, összegyűjtését, feldolgozását, elemzését és értelmezését.

A statisztika célja általában az objektív és megbízható adatok biztosítása az adatokat felhasználók, illetve a társadalom számára a döntéshozatal elősegítése érdekében.[3]

1.1.1. A statisztika eredete

A statisztika eredete még az emberiség kezdeti időszakára vezethető vissza. Úgy tartják, hogy az első államformák és központi hatalmak kialakulásával jelent meg a fogalom. Méghozzá az egyik legősibb ágával a népességstatisztikával, hiszen az adók meghatározásához ismerniük kellett az adott területen élők számát. Továbbá a statisztika

segítségével volt nyilvántartható a különböző erőforrások mennyisége is. A statisztika szót értelmezése szerint „az újlatin *statisticum collegium* („államtanács”) és az olasz *statista* („államférfi”, politikus) kifejezésekből származtatják.”[4]

A statisztika más ágai jóval később, a XIX. században jelentek meg, amikor rendszeresen kezdték gyűjteni a gazdasági és társadalmi adatokat világszerte. Ez idő tájékán nyerte el mai értelmezését is, minthogy („az adatgyűjtés és adatfeldolgozás általános tudománya”)[3]

1.1.2. Milyen egy jó statisztikai összefoglalás?

A statisztikai összefoglalások elengedhetetlen tartozékai egy adatelemzés eredményeinek közlésére. Ennek megírása viszont nehézkes lehet, főleg ha azt szeretnénk, hogy a közönség számára egyértelmű és letisztult legyen.

Saját tapasztalataim, illetve egy részben ideillő idegennyelvű cikk[5] szerint a következő szempontokra érdemes figyelni.

Realizáljuk illetve keressünk választ azokra a kérdésekre, hogy kik és mire fogják használni az összegyűjtött és feldolgozott adatokat. Ha például azt vesszük, hogy az eredményeket majd egy élelmiszer-cég piacutatói fogják elemezni és értelmezni, akik egy új terméket szeretnének piacra vinni, akkor nekik releváns statisztikák lehetnek a vásárlási szokások, illetve a hasonló termékek fogyasztása. Azaz tisztában kell lennünk a céllal és a célközönséggel.

Következő fontos pontja egy jól strukturált statisztikai összefoglalásnak a szisztematikus felépítés, amely segíti az olvasót az adatelemzés lépéseinek követésében. Az egyik gyakori formája az áttekintő összefoglaló (*angolul*: „Overview”), amely összefoglalja a fő eredményeket, feldolgozott adatokat. Emellett elengedhetetlen az eredmények egy részletesebb bemutatása, ahol az adatokat táblázatokkal, grafikonokkal és leíró statisztikákkal is megjelenítjük.

Érdemes megemlítenem a formázást valamint a stílus használatot. Törekedni kell, hogy a különböző diagramcímek, oldalcímek, figyelem felkeltő hatást keltsenek. Rendkívül meghatározó lehet a különböző komponensek, mint például grafikonok, táblázatok elhelyezkedése. Egy a középpontban elhelyezkedő grafikon sokkal hatásosabb lehet, mint sok apró egybeömlesztése. A különböző szöveg és grafikák tekintetében használjunk letisztult betűtípusokat, színeket. Továbbá a méreteket esztétikusan kell alkalmazni. A fehér tér és a margók kiegyensúlyozott elrendezést hozhatnak létre. A grafikonok könnyebb átláthatóságáért különböző tengelycímekkel és mértékegységekkel legyenek ellátva.

Mindezeket szemelőt tartva igyekeztem felépíteni a webalkalmazásban lévő statisztikai felületet.

1.2. Használt statisztikai módszerek az alkalmazásban

Azt már tudjuk, hogy mi is nagyjából a statisztika és hogy mi kell egy jó statisztikai összefoglalás megírásához, de milyen módszerek állnak a rendelkezésünkre és mire érdemes használni azokat? Az alábbi alszakaszokban a webalkalmazásban használt statisztikai módszereket és szerepüket mutatom be röviden.

1.2.1. Átlag szerepe

Az átlaggal és átlagszámítással már biztos mindenki találkozott. Angolul a szót többféleképpen is említik, egyik a „mean”, de ennél elterjedtebb a statisztikai oldalakon gyakrabban előforduló „average” szó lesz számunkra meghatározó a dolgozat további részében.

Mikor és mire érdemes használni? Az átlag a vizsgált adathalmaz általános tendenciájáról próbál információt nyújtani. Például az olyan adathalmazok elemzésekor a leghasznosabb, amelyekhez kevés kiugró érték tartozik.

1.2.2. Maximum és Minimum szerepe

A maximum az adatok közül a legnagyobb értéket, míg a minimum a legkisebbet jelöli. Mikor és mire érdemes használni? Az adatok elemzésében leginkább az extrém értékek azonosításában és az adathalmaz jellemzésében van szerepe.

1.3. Adatvizualizáció és használt grafikonok az alkalmazásban

Az adatvizualizáció másképp az adatok ábrázolása olyan általános grafikák segítségével, mint a diagramok, ábrák, táblázatok vagy akár animációk. Ezek alkalmazása segítik az információk könnyebben való megértését. A legelterjedtebbek közé tartoznak a következők:[6]

- Táblázatok (*Tables*)
- Kör és oszlopdiagramok (*Pie and bar charts*)
- Vonaldiagramok (*Line charts*)
- Hisztogramok (*Histograms*)
- Szórásdiagramok (*Scatter plots*)


- Hő térképek (*Heat maps*)
- Fatérképek (*Tree maps*)

Ezek közül négy darab található meg az alkalmazásban, melyeket az alábbi alszakaszokban ismertetek.

1.3.1. Táblázat

A táblázatot az adatok strukturáltan való megjelenítésére használom. Az alkalmazásban többféle nézetben is megtalálható és ez szolgál például a válaszok és hozzátartozó információk kelistázására, valamint az adott válaszok törlésére. Továbbá az exportálás mint funkció az adott táblázat nézetéből történik.

IMAGE_ID ↑	RESPONSE	SESSION_ID	TIME	HAND	MAJOR	CREATED_AT	<input type="checkbox"/>
16480	9	t58M1sMcHYXJNe93hZqfO...	870	Unknown hand	Unknown major	2024. 04. 12. 4:59:39	<input type="checkbox"/>
22732	5	t58M1sMcHYXJNe93hZqfO...	908	Unknown hand	Unknown major	2024. 04. 12. 4:59:39	<input type="checkbox"/>
30458	4	t58M1sMcHYXJNe93hZqfO...	1182	Unknown hand	Unknown major	2024. 04. 12. 4:59:39	<input type="checkbox"/>
32741	1	t58M1sMcHYXJNe93hZqfO...	879	Unknown hand	Unknown major	2024. 04. 12. 4:59:39	<input type="checkbox"/>
33720	4	t58M1sMcHYXJNe93hZqfO...	923	Unknown hand	Unknown major	2024. 04. 12. 4:59:39	<input type="checkbox"/>
43393	3	t58M1sMcHYXJNe93hZqfO...	1003	Unknown hand	Unknown major	2024. 04. 12. 4:59:39	<input type="checkbox"/>
45000	3	t58M1sMcHYXJNe93hZqfO...	1778	Unknown hand	Unknown major	2024. 04. 12. 4:59:39	<input type="checkbox"/>
45217	3	t58M1sMcHYXJNe93hZqfO...	1022	Unknown hand	Unknown major	2024. 04. 12. 4:59:39	<input type="checkbox"/>
57586	0	t58M1sMcHYXJNe93hZqfO...	855	Unknown hand	Unknown major	2024. 04. 12. 4:59:39	<input type="checkbox"/>
61736	8	TYXCPSik5zo3V9mWPkvPp...	735	Unknown hand	Unknown major	2024. 04. 12. 4:58:33	<input type="checkbox"/>

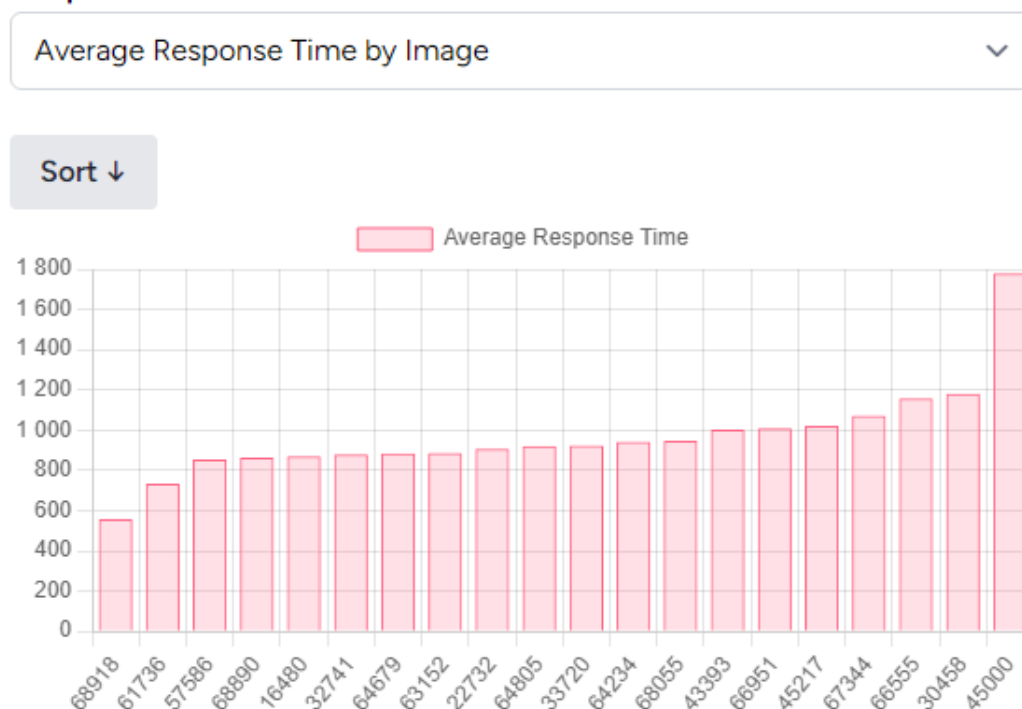
10 / 20
Show more


1.1. ábra. A Response táblázat nézete a webalkalmazásban

1.3.2. Oszlopdiaagram

Az oszlopdiaagramok szolgálnak a különböző kategóriák közötti összehasonlításban és az adatok vizuális megjelenítésében. Több nézetben is alkalmazom, az egyik példa erre a válaszadásra fordított idő átlagos megjelenítése azonosításonként.

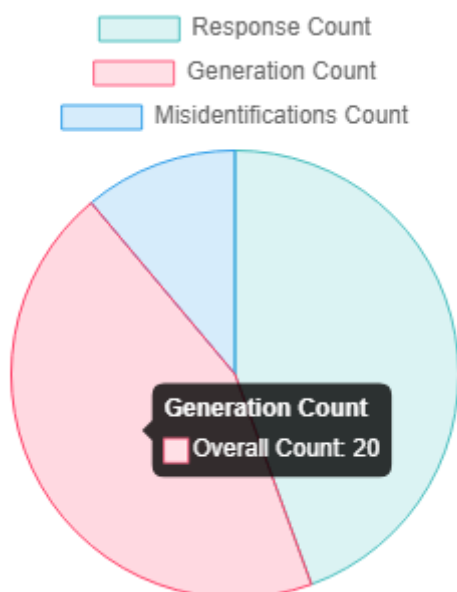
Responses Bar Chart



1.2. ábra. Válaszokra fordított átlagos idő oszlopdiagramja az alkalmazásban

1.3.3. Kördiagram

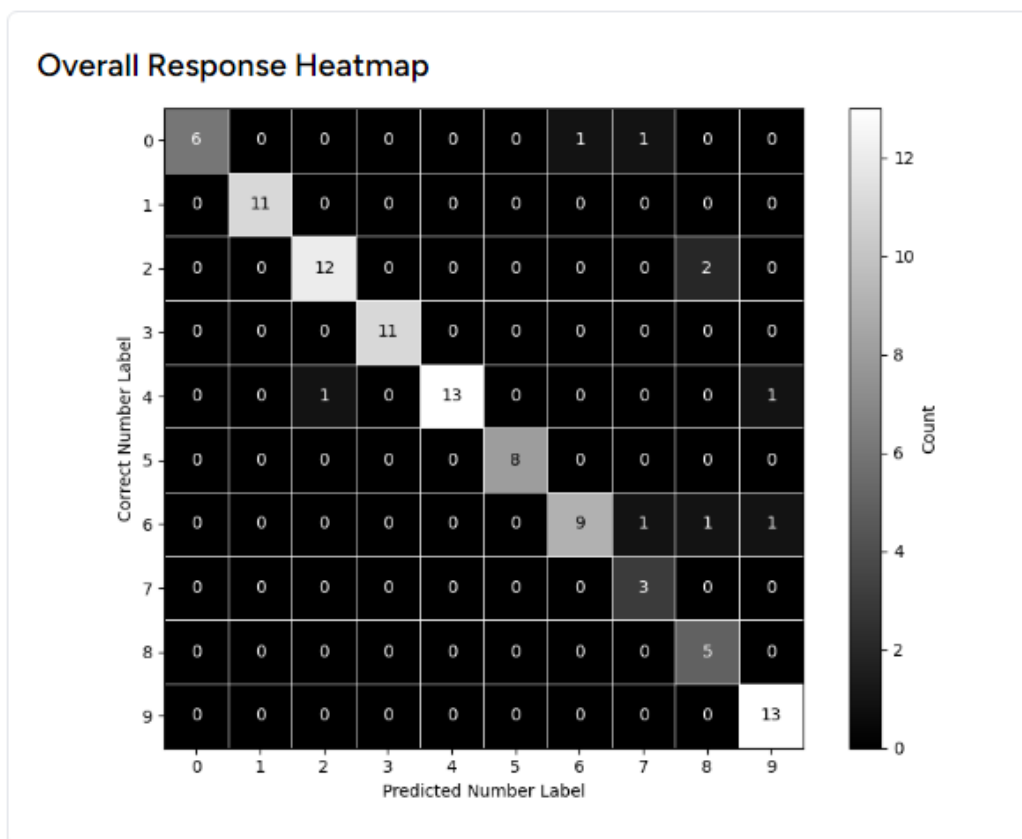
A kördiagram mutatja meg az adatok arányát a teljes mintához viszonyítva. Az én példámban három különböző szempontot jeleníttek meg egyszerre, majd ezek között figyelhető meg az arányok eloszlása.



1.3. ábra. Kördiagram az alkalmazásban

1.3.4. Hő térkép

A hő térkép elsőre táblázatnak tűnik, de emellett egyszerre egy összehasonlító nézetet mutat az adatokról. Ezt úgy éri el, hogy a különböző értékeket más-más színekkel színezi meg. Például a magasabb értékek színe világosabb, még az alacsonyabbaké sötétebb. Ezt alkalmazva könnyebben átlátható vizuális ábrázolást kapunk. Az alkalmazásban a válaszokból begyűjtött azonosításokat jelenítem meg egy hő térképen, amelyből egyszerűen leolvasható az adott számjegyre történt félreasonosítások száma.



1.4. ábra. Hő térkép az alkalmazásban

1.4. Adatok gyűjtése

Ahhoz, hogy még mielőtt bármilyen statisztikai megjelenítésről beszélhetnénk kellenek adatok. A webalkalmazásomnak is ez az alapja. Itt a begyűjtendő adatok a válaszok illetve a kérdések is. A gyűjtés egy online felmérés, más szóval teszt kitöltésén keresztül történik. A kitöltők MNIST képeket kapnak kérdésként amelyeket azonosítaniuk kell. Minden azonosítás egy adat, sőt a be nem érkezett válasz is egy adat, hisz a kérdés generálása a választól függetlenül megtörtént. Egy-egy ilyen válaszból kiderül, hogy ki volt a válaszadó, milyennek azonosította az adott képet (*helyesnek vagy helytelennek*),

mennyi idő alatt és hogy pontosan mikor történt az azonosítás. Továbbá mint említettem maga a kérdés is egy adat, ugyanis kiderül, hogy pontosan milyen számjegyet ábrázol, illetve hogy hányszor generálódott le az adott kérdés. Az utóbbi információt használva például manipulálhatjuk a kérdések generálásának folyamatát saját érdekeinknek megfelelően.

Az adatok gyűjtéséért felelős intézkedéseket a szakdolgozati társam készítette el, amelyekről bővebben az Ő szakdolgozatában lehet olvasni.



1.5. ábra. Képernyőkép a kérdőívről

2. fejezet

Használt technológiák

2.1. Tervezéshez használt technológiák

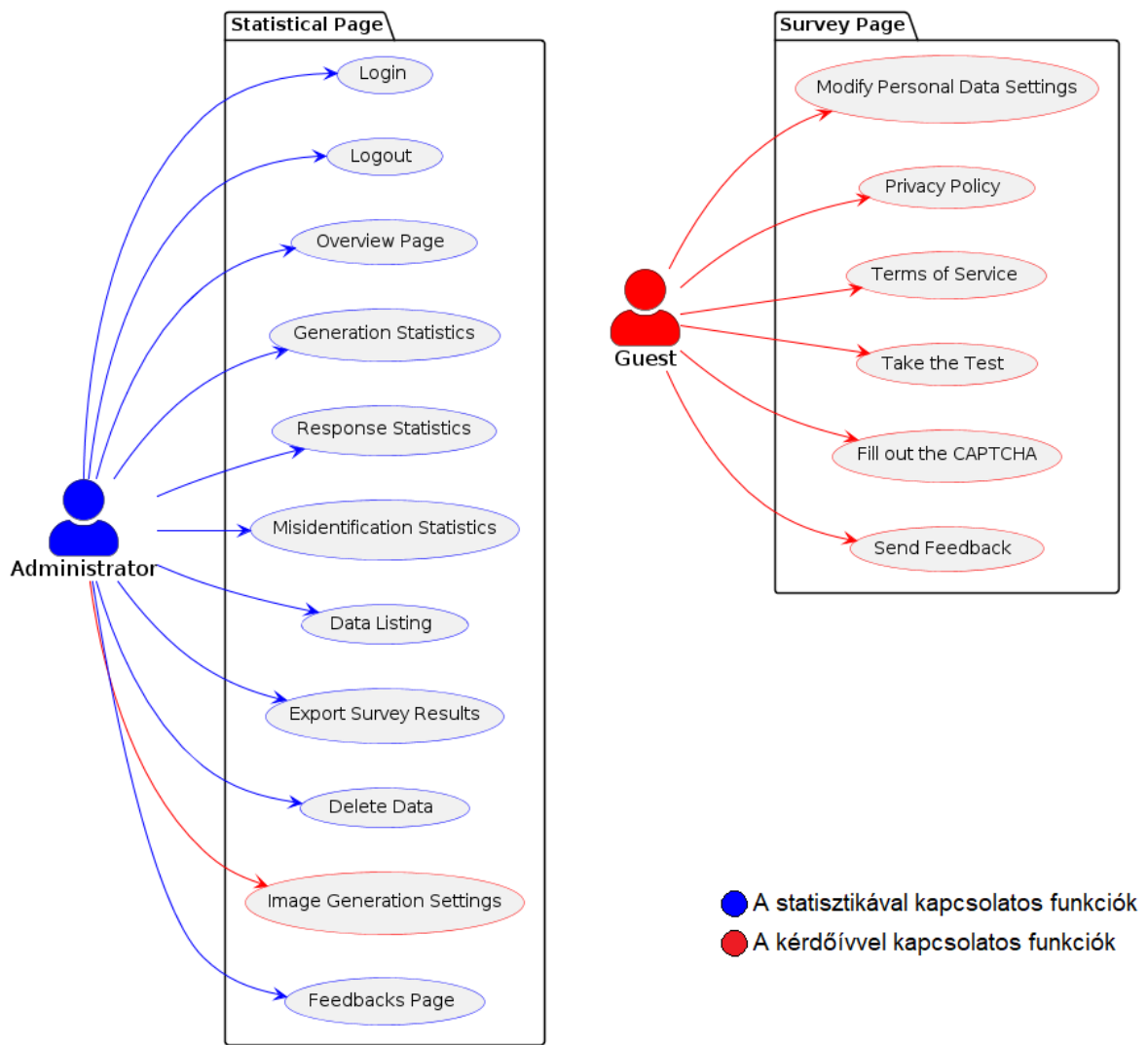
Ebben a szakaszban ismertetem azokat a technológiákat, amelyeket a webalkalmazás tervezési folyamatában alkalmaztam. A tervezés kulcsfontosságú szakasza az alkalmazás fejlesztésének, ugyanis ez teszi lehetővé, hogy hatékonyan modellezzük és megtervezzük az alkalmazás architektúráját és funkcionalitását.

2.1.1. PlantUML

A PlantUML¹ egy olyan eszköz, amelynek segítségével diagramokat hozhatunk létre szöveges leírások alapján. Ez a szöveges alapú megközelítés lehetővé teszi a gyors és könnyű diagramkészítést, amelyek segítenek megérteni az alkalmazás tervezési részleteit. A PlantUML lehetőséget nyújt többek között a szekvencia-, használati eset-, osztály-, állapotgép- és más típusú diagramok elkészítéséhez.

Az egyik ilyen általam elkészített diagram a használati eseteket ábrázoló, úgynevezett Use case diagram volt. A használati eset diagram a szoftverfejlesztésben használt vizuális ábrázolás, amely a rendszer szereplői és maga a rendszer közötti kölcsönhatásokat ábrázolja. Ezek a diagramok hozzásegítenek a rendszer funkcionális követelményeinek meghatározásához és annak megértéséhez, hogy a felhasználók hogyan fognak kölcsönhatásba lépni a rendszerrel. Az alábbi ábrán a webalkalmazás teljes Use case diagramja látható, melyből az én feladatom a statisztikával kapcsolatos funkciók megvalósítása voltak (kék színnel jelölve).[7]

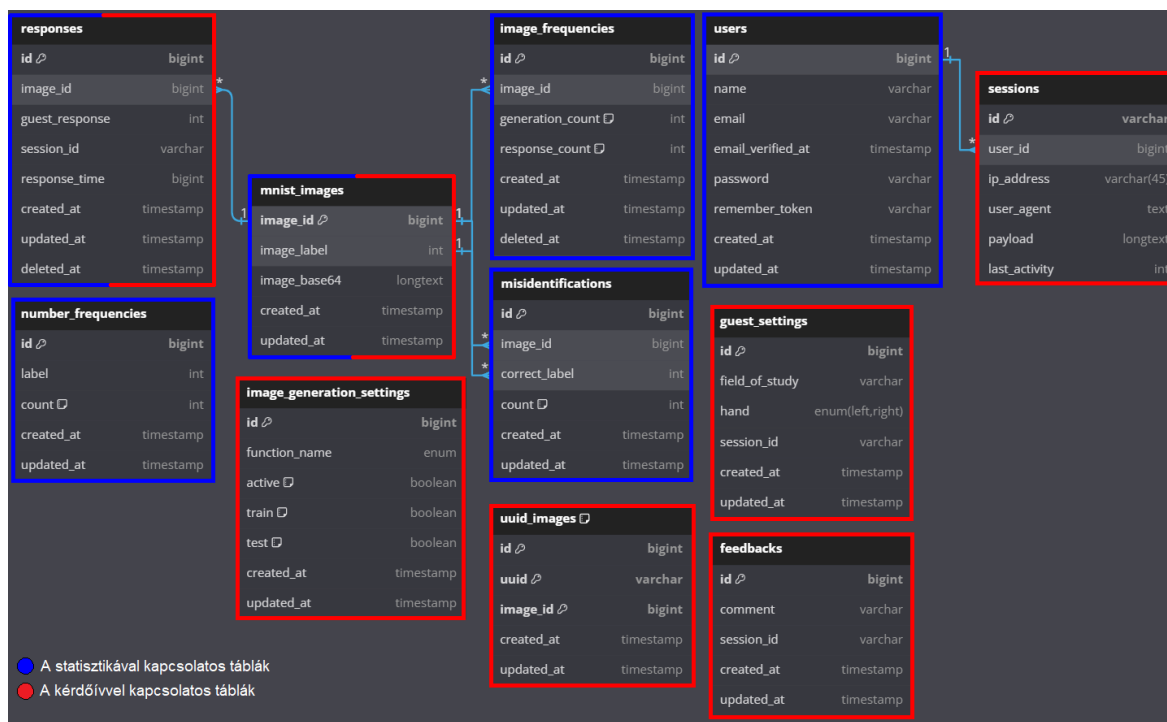
¹ UML - Unified Modeling Language



2.1. ábra. Use case ábra az alkalmazáshoz

2.1.2. dbDiagram.io

A dbDiagram.io[8] egy online felület, ahol adatbázis-sémákat tervezhetünk és modellezhetünk. Egy jól megtervezett adatbázis rendkívül elősegíti a hatékony munka folyamatát illetve a kezdeti félreértések elkerülését. Természetesen a fejlesztés során tapasztaltam egymásnak ellentmondó vagy megkérdőjelezhető megoldásokat, de menet közben ezeket javítani tudtam. Ugyanis itt valós adatbázis nem jön létre illetve az adatok közötti kapcsolatok egy pillanat alatt módosíthatóak. Az elkészült teljes adatbázis az alábbi ábrán megtekinthető, illetve a statisztikai oldalhoz kapcsolódó táblák egy részének leírása a 32. oldalon, az **Adatbázis táblák** szakaszban olvasható.



2.2. ábra. Adatbázis, külön jelölve a statisztikai és kérdőív tábláit

2.2. Megvalósításhoz használt technológiák

Ebben a szakaszban azokat a technológiákat mutatom be, amelyeket az alkalmazás megvalósításához használtam. Ezek a technológiák tették lehetővé az alkalmazás különböző részein való kódolást és fejlesztést, beleértve az adatbázis-kezelést és felhasználói felület létrehozását. Ezen technológiák kiválasztásában szerepet játszottak az egyetemi éveim során szerzett tapasztalatok.

2.2.1. MySQL

A MySQL egy nyílt forráskódú relációs adatbázis-kezelő rendszer, amelyet az alkalmazás adatbázisának tárolására és kezelésére használtam. A MySQL lehetővé teszi a hatékony adattárolást és lekérdezést az alkalmazás számára.

Itt említem meg a XAMPP, mint ingyenes, nyílt forráskódú szoftvert, amely egy integrált környezetet biztosít a webalkalmazások fejlesztéséhez helyi számítógépen. A XAMPP tartalmazza az Apache HTTP szerver, a MySQL adatbázis-kezelő rendszert és a PHP programozási nyelvet.[9]

2.2.2. HTML és CSS

A HTML (HyperText Markup Language) és CSS (Cascading Style Sheets) a webes fejlesztésben játszanak szerepet. A strukturált és esztétikus felhasználói felületek létrehozásáért felelnek.

A HTML egy alapvető nyelv a webes tartalom strukturálására és formázására. HTML elemekkel határozzuk meg az oldal struktúráját, például a címeket, paragrafusokat, listákat és linkeket.[10] A dolgozatban a HTML-es kódok főként JSX formátumban fognak megjelenni. Erről majd a 16. oldalon, **JSX** szakaszban olvashat.

A CSS a stílusok megadására szolgáló nyelv, amely szép és esztétikus megjelenést biztosít a weboldalnak. A CSS segítségével beállíthatjuk a színeket, betűtípusokat, elrendezéseket és egyéb vizuális tulajdonságokat az oldalon.[11] A kódok túlnyomó része a CSS-nek egy keretrendszerével van megvalósítva, amelyről bővebben a 17. oldalon, **Tailwind CSS** szakaszban olvashat.

2.2.3. PHP

A PHP az egyike a nyílt forráskódú szkriptnyelveknek, amely különösen alkalmas a webfejlesztésre. Ez teszi lehetővé az olyan dinamikus weboldalak készítését, amelyek például adatbázisból származó információkat jelenítenek meg vagy felhasználói interakciókat kezelnek. Emellett a PHP kód beágyazható HTML-be, ami lehetővé teszi a tartalom dinamikus generálását és az interaktív elemek létrehozását.[12]

2.2.4. JavaScript

A JavaScript egy olyan szkript- vagy programozási nyelv, amely lehetővé teszi az összetett funkciók megvalósítását a weboldalon. Legyen az akár egy olyan funkció amely reagál a felhasználói interakciókra és eseményekre. Például, ha azt szeretnénk, hogy egy gombra kattintásra megváltozzon a weboldal szövege, akkor egy egyszerű JavaScript kóddal ezt megtehetjük. A szakdolgozatomban ezen JavaScriptek hasonlóan a HTML-hez, majd JSX formátumban jelennek meg együttesen. Erről bővebben a 16. oldalon, **JSX** szakaszban olvashat.

2.2.5. JSX

A JSX egy szintaxis kiegészítő a JavaScripthez, amely lehetővé teszi a HTML-szerű jelölés használatát egy JavaScript-fájlon belül. Az én tetszésemet a maga tömörségével nyerte el, ugyanis például egy oszlopdiagram komponens létrehozásánál egyszerre tartalmazza mind a strukturális, mind a stílusos és a logikai részeket. Ennek segítségével gyorsabb és hatékonyabbnak éreztem a fejlesztést.

Amikor két külön nyelv egy formátumban jelenik meg kérdés lehet, hogy melyik konvenciót alkalmazzuk. A JSX esetében mivel közelebb áll a JavaScripthez ezért annak szintaxisát és konvencióit használja. Például a HTML attribútumok mint a *class* helyett *className* lesz, illetve ebből már látszódik, hogy a változónevek *camelCase* stílusúak.[13]

2.2.6. Tailwind CSS

A Tailwind CSS egy korszerű CSS-keretrendszer, amely lehetővé teszi a gyors és testre szabott webalkalmazások stílus megalkotását anélkül, hogy részletes egyedi CSS-kódokat kellene írni. Tehát egyszerűen a HTML kódon belül maradva módosítható például az elrendezés, a szín, a távolság, stb. Mindezt egy új CSS megírása nélkül. A következőekben bemutatok egy példán keresztül, hogy egy gombot miként lehet személyre szabni. `<button class="h-10 px-6 font-semibold rounded-md bg-black text-white" type="submit">Ez egy gomb</button>`

- **h-10:** A gomb 10 egységnyi fix magas
- **px-6:** A gomb vízszintes belső margója 6 egység
- **font-semibold:** A gomb szövege félkövér
- **rounded-md:** A gomb szegélye lekerekített
- **bg-black:** A gomb háttérszíne fekete
- **text-white:** A gomb szövegszíne fehér

Létezik egy már előre elkészített rendszer az osztálynevekkel, amelyeket használva személyre szabhatóak a kívánt komponensek. Így nemhogy egyéni CSS-t nem kell írni, de a „tökéletes” osztálynevek kitalálása sem okoz gondot.[14]

2.2.7. Chart.js

A Chart.js egy JavaScript alapú diagramkönyvtár, amely lehetővé teszi az interaktív és testre szabható diagramok készítését webes alkalmazásokhoz. Ennek a könyvtárnak a segítségével különböző diagramokat hoztam létre, például oszlopdiagramot, kördiagramot. Ezek tulajdonságai legyen az szín vagy stílus ízlés szerint változtathatóak. Az adatok vizuális ábrázolásához elengedhetetlennek tartom, egyrészt fejlesztői szemszögből, azért mert megkönnyíti a diagramokkal való munkát köszönhetően a részletes dokumentációjának[15]. Másrészt pedig a felhasználók szemszögből amiért az adat könnyebb értelmezését szolgálja.

2.2.8. Inertia

Az Inertia egy új megközelítés a klasszikus szerver vezérelt webalkalmazások készítéséhez vagy ahogy a fejlesztői hívják modern monolitikus².

² A „monolitikus” kifejezés olyan alkalmazásokra utal, amelyek kódbázisa egységes egységként épülnek fel, önállóan és más alkalmazásoktól függetlenül.[17]

Segítségével elkészíthetők kliensoldalon megjelenített (*renderelt*) egyoldalas alkalmazások (SPA)³ anélkül, hogy azok bonyolultságával foglalkozni kellene. Ezt úgy éri el, hogy kihasználja a már meglévő szerveroldali mintákat. Például egy Laravel backenddel összekötött alkalmazásban a kliens oldalról dinamikusan frissül az oldal anélkül, hogy újra kellene tölteni, hasonlóan az SPA-hoz, de az Inertia továbbra is a szerver oldali logikát használja a tartalom generálásához. Nem rendelkezik kliensoldali útvonalkezeléssel és nincs szüksége API-ra sem. Képes „bármilyen” backend keretrendszerrel együttműködni, de elsősorban a Laravelhez optimalizált.

Az Inertia nem egy keretrendszer és nem is egy meglévő szerveroldali vagy kliensoldali keretrendszer helyettesítése. Inkább úgy tervezték, hogy velük együtt működjön. Ezt az adapterek segítségével teszi meg. Jelenleg három hivatalos kliensoldali adapterük van (React, Vue és Svelte) és két szerveroldali adapterük (Laravel és Rails).[16]

2.2.9. Laravel

A Laravel egy PHP alapú webalkalmazás fejlesztésére szolgáló keretrendszer, amely MVC⁴ architektúrával dolgozik. Köszönhetően a rendkívül részletes dokumentációjának könnyen tanulható és nem véletlenül a fejlesztők egyik kedvence. Többek között említésre méltó, hogy intuitív elegáns szintaxissal rendelkezik, beépített funkciók és kiegészítők széles választékát biztosítja. Támogatást biztosít az adatbázis-migrációkhoz, útvonalkezeléshez, egységbezáráshoz.[19]

Számomra azért volt egyértelmű, hogy Laravelt használom a backend fejlesztéshez, mert a telepítése egyszerű, több projektet volt szerencsém ezzel a keretrendszerrel megvalósítani, illetve a már említett részletes felhasználói dokumentáció egyféle biztonság érzetet tudott nyújtani.

³ SPA: Single Page Application

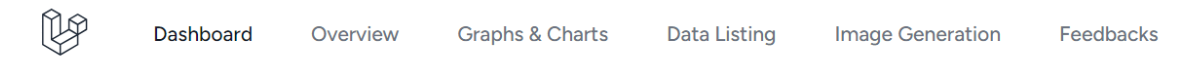
⁴ MVC: Model View Controller

3. fejezet

Felhasználói dokumentáció

3.1. Dashboard

A bejelentkezést követően a /dashboard útvonalon találja magát a felhasználó, ahol a különböző oldalak rövid ismertetéséről olvashat. A többi oldal navigálásához a navbar-on van lehetőség lépkedni.



3.1. ábra. Navbar a különböző nézetek navigálásához

3.2. Overview

Fontosnak tartottam egy átfogó nézetet, amely mindenféle diagram vagy táblázat nélkül megjeleníti az első ránézésre érdekesebb és kitűnőbb adatokat. Az Overview oldalra navigálva három külön kategóriában jelennek meg összesített és kiemelt adatok. Ezeket az alábbi listában felsorolom, illetve a szakasz további részében rövid leírást adok róluk.

1. Generation Statistics

- Total Generated Images
- Most Generated Image Id
- Most Generated Number

2. Misidentification Statistics

- Total Misidentifications
- Most Misidentified Image Id

- Most Misidentified Number

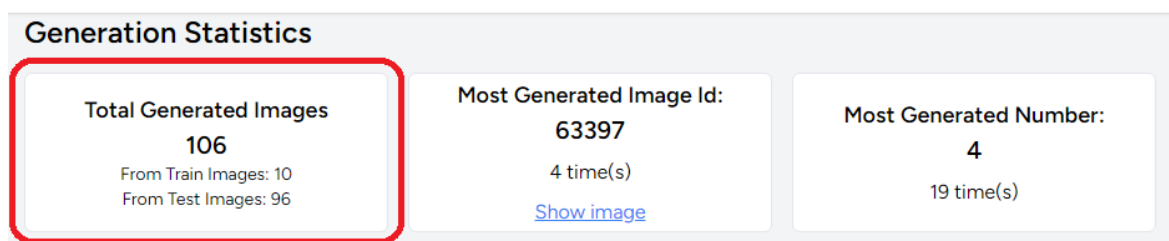
3. Response Statistics

- Total Responses
- Most Responded Image Id
- Average Response Time / Image
- Average Response / Day

3.2.1. Generation Statistics

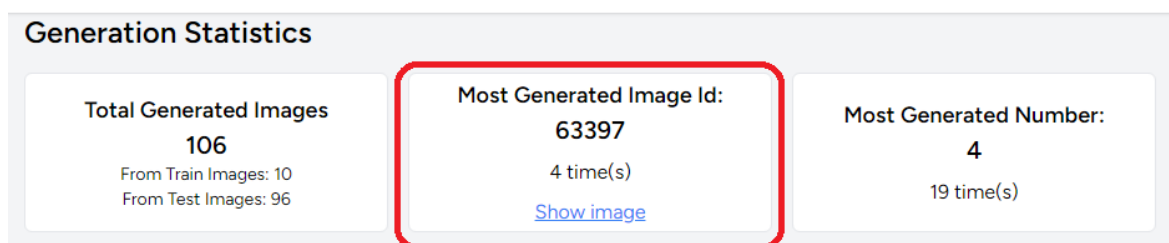
A **Generation Statistics** részleg tartalmazza a kérdőívben megjelenő „képek” generálási adatait. A képeket, azért idézőjelben használtam, mert a mi esetünkben a kérdőívben megkapott kérdések képek formájában történnek, lásd a 11. oldalon **Adatok Gyűjtése** szakaszban.

Tehát ez a részleg tartalmazza többek között az eddig legenerált képek számát (*Total Generated Images*). Ez az adat még lebontható egy kisebb részlegre, amely azt mutatja meg, hogy a generált képek közül melyek voltak a „train” és melyek a „test” adathalmazból. Ez fontos lehet a későbbi kép generálás manipulálásához, amelyről az **Image Generation** szakaszban olvashat.



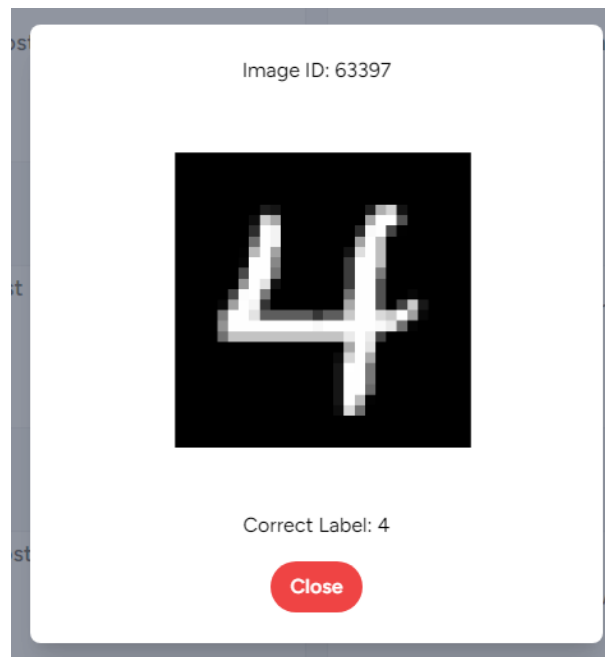
3.2. ábra. Total Generated Images az Overview nézetén

Emellett megtalálható a legtöbbször legenerált kép ID-ja (azonosítója) (*Most Generated Image Id*), illetve hogy pontosan hányszor generálódott le az adott a kép.



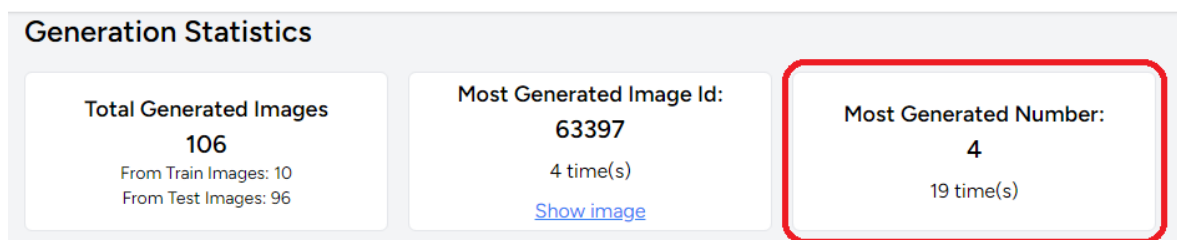
3.3. ábra. Most Generated Image Id az Overview nézetén

Található rajta egy (*Show image*) gomb is, amely egy felugró ablakban megjeleníti az adott képet ellátva azzal az információval, hogy pontosan melyik számjegyet ábrázolja („Correct label”).



3.4. ábra. Most Generated Image Id Popup-ja az Overview nézetén

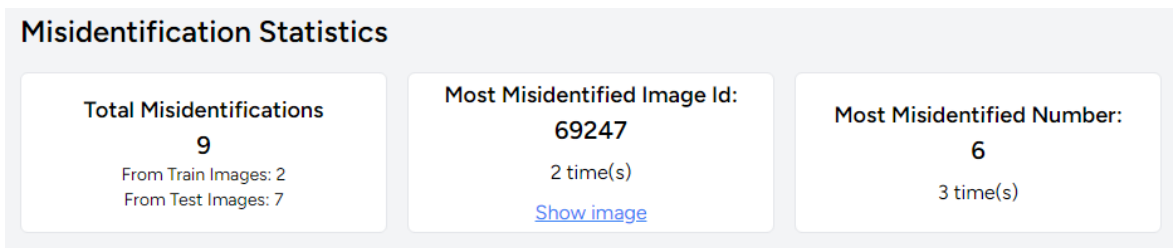
Végül a *Most Generated Number*, amely azt mutatja meg, hogy a leggenerált képek közül melyik az a számjegy ami a legtöbbször előfordult a képeken. Jól leolvasható az alábbi ábrán, hogy a legtöbbször (19-szer) olyan képeket kaptak a felhasználók, amelyek a négyes számot ábrázolták.



3.5. ábra. Most Generated Number az Overview nézetén

3.2.2. Misidentification Statistics

A félreazonosításokért felelős statisztikákat már nem külön-külön, hanem egyben ismertetem, ugyanis a felépítése megegyezik a **képgenerálási** statisztikákkal, viszont a jelentősége különbözik.

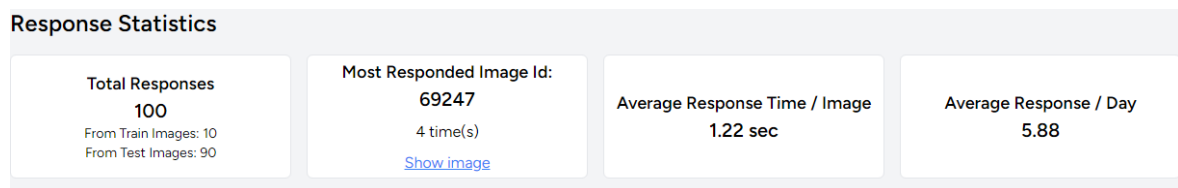


3.6. ábra. Misidentification Statistics az Overview nézeten

A fenti **ábráról** jól leolvasható a félreazonosítások száma (9), a legtöbbet félreazonosított kép ID-ja (69247), illetve az a számjegy amelyet a felhasználók a legtöbbször félreazonosítottak (6). Ebben az esetben a legutóbbi adatnak több jelentőséget lehet fordítani, mint a **generálásnál**, ha csak arra gondolunk, hogy ebből leolvasható, hogy melyik az a 0 és 9 közötti számjegy amelyet a legtöbbször félreazonosítanak a felhasználók.

3.2.3. Responses Statistics

Az utolsó részleg a válaszokért felelős statisztikák, amelynek ugyancsak hasonló a felépítése, de kiegészül két új fontosabb információval.

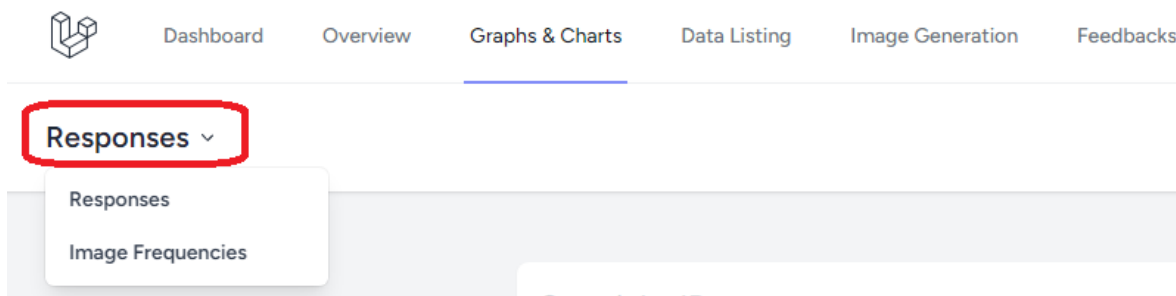


3.7. ábra. Responses Statistics az Overview nézeten

Az **ábrán** leolvasható az eddigi azonosítások száma (100). Ebből és a **Generation Statistics** összes legenerált képek számát leíró információból már következtethető, hogy hány olyan legenerált kép volt, amelyre nem érkezett válasz. Emellett itt is szerepel a legtöbbet megválaszolt kép ID-ja (69427). Továbbá egy összefoglaló adat arról, hogy átlagosan mennyi idő alatt történnek az azonosítások képenként. Végül arról is informálódhatunk, hogy átlagosan mennyi válasz érkezik naponta.

3.3. Graphs & Charts

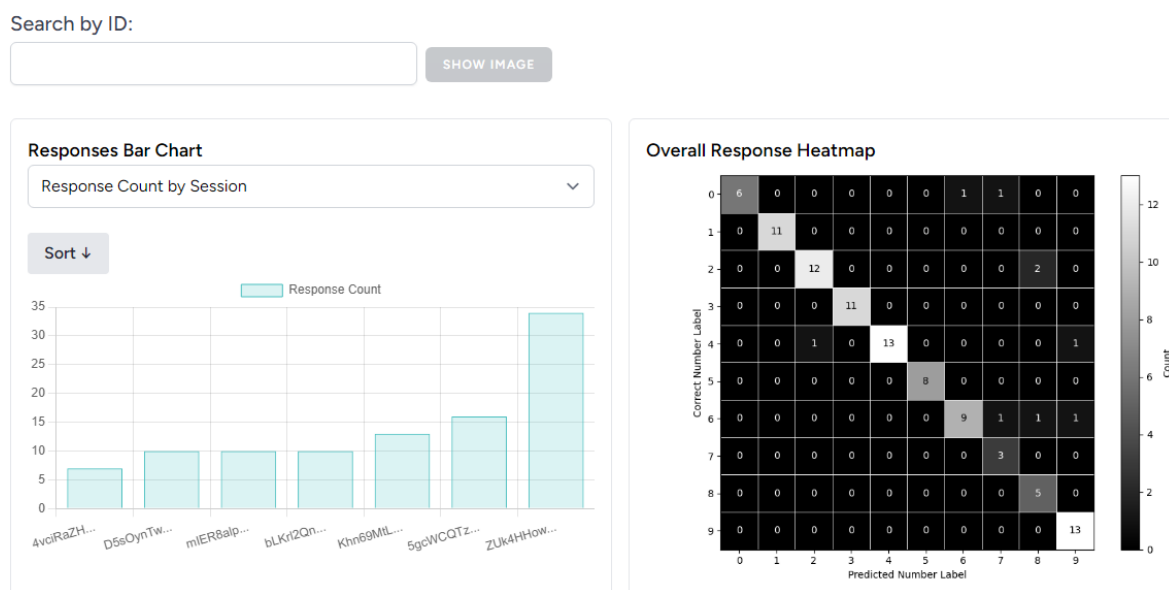
A **navbaron** tovább haladva a grafikonok és diagramok fülön két főnézetet különíttek el. Egyik a Responses (válaszok), másik az Image Frequencies (képi frekvenciák) nézete. Ezen nézetek között a navbar alatti legördülő menüpont lenyitásával lépkedhetünk, ahogy azt az alábbi ábra mutatja.



3.8. ábra. Nézetek közötti váltás a Graphs & Charts oldalon

3.3.1. Responses

A válaszokért felelős nézeten megtalálható egy kereső mező, amely az alatta lévő oszlopdiagram megjelenítését módosítja a keresett értékkel. A keresés ID alapján történik, amely lehet session ID, vagy image ID. A keresés mező mellett található egy „SHOW IMAGE” gomb, amely a keresett image ID-hoz tartozó képet mutatja meg egy újonnan felugró ablakban, ugyanazt a már korábban bemutatott Popup-ot használva, amelyet **ide kattintva meglekinthet**.



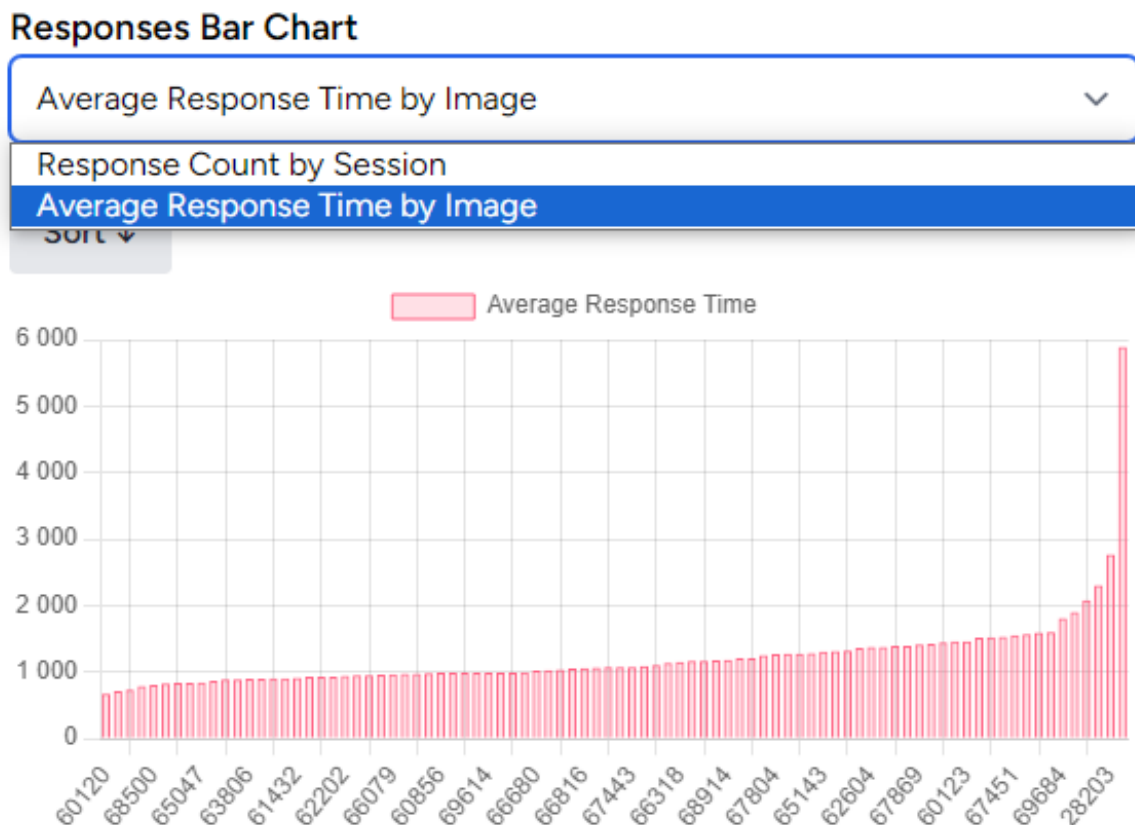
3.9. ábra. Responses grafikonok a Graphs & Charts nézeten

A válaszok oszlopdiagramja (Responses Bar Chart) alapértelmezetten a felhasználóhoz társított session ID-kénti válaszadások számát mutatja meg. Ez azért lehet fontos, mert ebből leolvasható, hogy egy adott felhasználó mennyi képet azonosított, illetve segítségével szűrhetjük a gyanús kitöltéseket, valamint a visszatérő kitöltőket. A diagram **,X'** tengelyén a **session ID**, míg az **,Y'** tengelyén a **válaszadások száma** szerepel.

Az oszlopdiagramhoz tartozik egy második nézet is, amely egy legördülő menüponton változtatható, ezt **ide kattintva** az ábrán láthatja. Ez a második nézet a

képenkénti átlagos válaszidő oszlopdiagramját jeleníti meg. Ebből juthatunk olyan következtetésekre, mint például melyek azok a képek amelyeket a leglassabban tudnak azonosítani, legtöbbet időznek el rajtuk, azaz bonyolultabb mint a többi. Természetesen ez inkább nagy mennyiségű megkapott válaszok után használható adat. A diagram **,X'** tengelyén az **image ID**, míg az **,Y'** tengelyén az **eltelt idő** szerepel milliszekundumban (ms) kifejezve.

Mindkét diagram nézetén lehetőség van a növekvő illetve csökkenő mód rendezésére, melyet a **Sort** gombbal valósul meg.



3.10. ábra. Responses nézetén belüli diagramváltás

Végül a válaszokhoz tartozik még egy hő térkép (heatmap) is (**ide kattintva megtekintheti**), amelyből leolvasható az adott számjegyre történt összesített azonosítások száma. Ebből pedig egyszerűen kinyerhető a félreazonosítások mértéke is. A hő térkép **,X'** tengelyén a **válaszként megadott számjegyek**, míg az **,Y'** tengelyén **0-9-ig a lehetséges számjegyek** szerepelnek.

3.3.2. Image Frequencies

A Graphs & Charts másik nézete a már említett képi frekvenciákért felelős statisztikákat mutatja be diagramokkal. Itt is megtalálható egy kereső mező, amelyben kifejezetten image ID-ra kereshetünk rá, amely a keresett értékhez tartozó diagramokat rajzolja ki. Továbbá a „SHOW IMAGE” gomb, amely a keresett image ID-hoz tartozó képet mutatja meg egy újonnan felugró ablakban, ugyanúgy mint a Responses nézeten. **Ide kattintva ezt is megtekintheti.** Az alábbi ábrán a ,69247’ azonosítójú képre keresett eredménye látható.

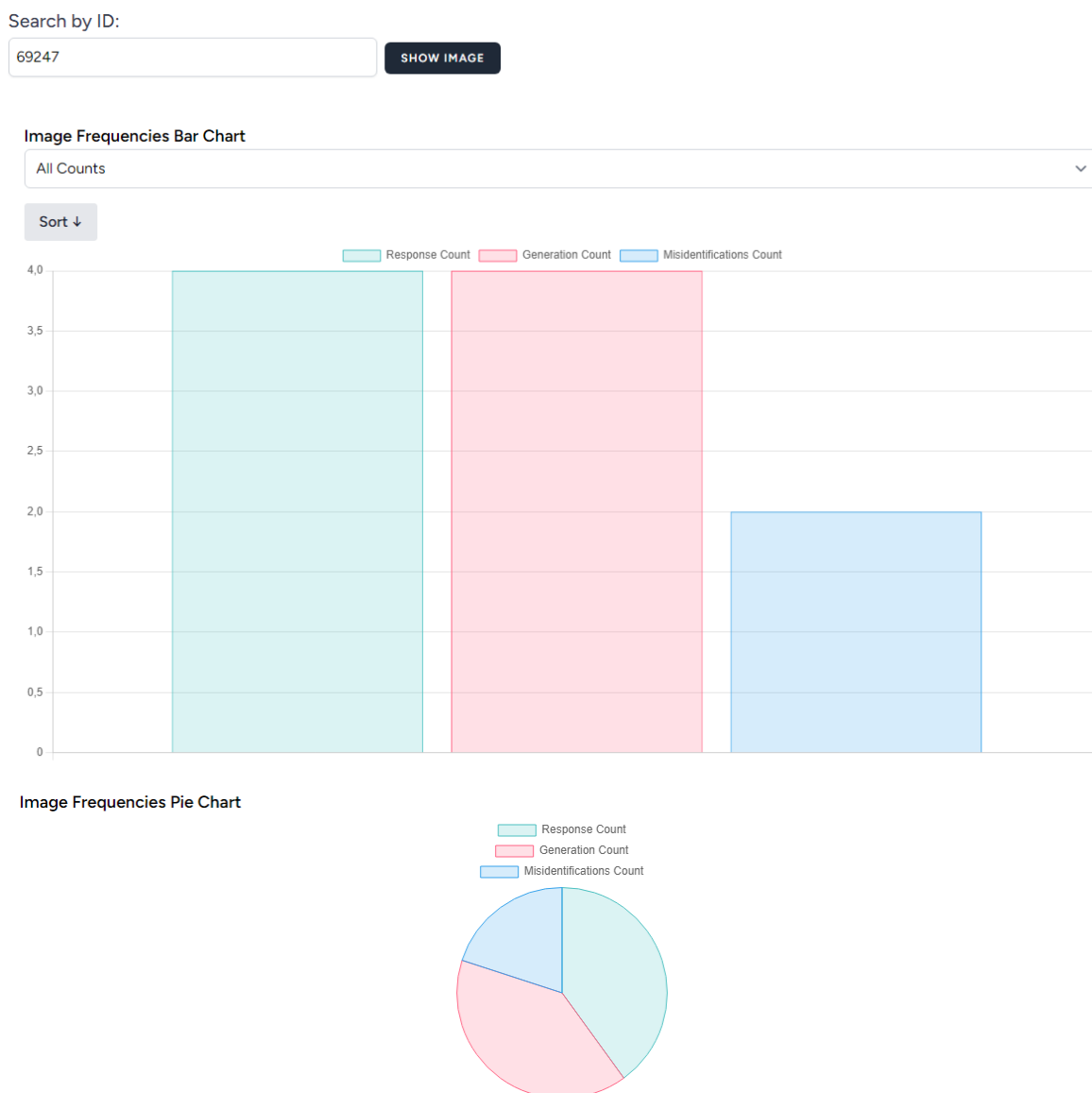


Image Frequencies Bar Chart

All Counts

Sort ↓

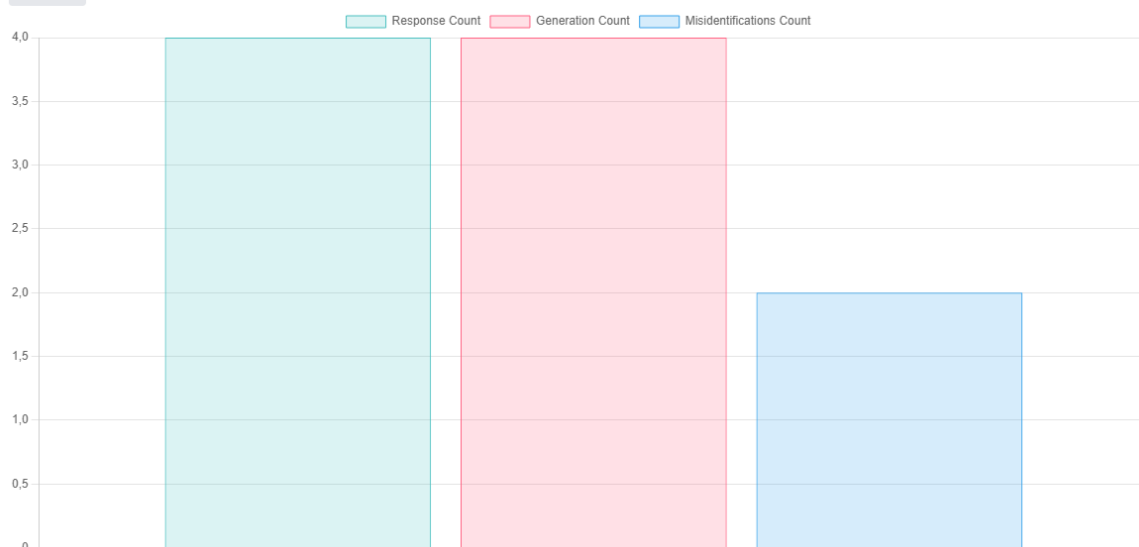
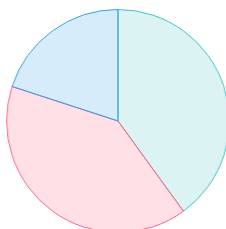


Image Frequencies Pie Chart

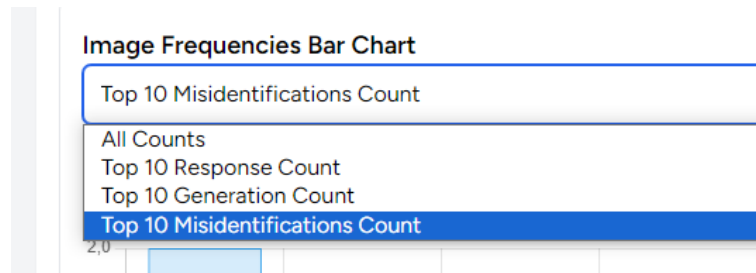
Response Count
Generation Count
Misidentifications Count



3.11. ábra. Image Frequencies nézeten oszlop- és kördiagram a ,69247’-es ID-hoz

Az oszlop és kördiagramon háromféle adatot jelenítünk meg egy időben, ezek a következők: **Response Count**, **Generation Count**, **Misidentification Count**, azaz rendre a válaszok száma, generálások száma és a félreazonosítások száma.

Az itt található oszlopdiagramhoz összesen négyféle nézet létezik. Alapértelmezetten az „**All Counts**” a már egyszer legalább legenerált image ID-hoz tartozó adatok tulajdonságát jeleníti meg. Továbbá három darab Top 10-es kategóriánkénti nézet, amely a legtöbb tíz olyan image ID-t jeleníti meg, amelyek az adott kategóriában a legmagasabb számlálókkal bírnak. A nézetek közötti váltást az oszlopdiagram felett található legördülő menüvel lehet megvalósítani, ezt az alábbi ábrán megtekintheti.



3.12. ábra. Image Frequencies nézeten belüli diagram váltás

Az oszlopdiagramok **,X'** tengelyén az **image ID**-k, míg az **,Y'** tengelyén az **adott kategória mennyiségei** szerepelnek.

A diagram nézeteken belül lehetőség van a növekvő illetve csökkenő mód rendezésére, melyet a **Sort** gombbal valósíthat meg.

3.4. Data Listing

A Data Listing nézet szolgál az adatok táblázatbeli megjelenítéséért. Akárcsak a Graphs & Charts nézeten itt is két különböző fő nézetet különítek el. Egyik a Responses (válaszok), másik az Image Frequencies (képi frekvenciák) táblázatbeli kilistázása. Ezen nézetek között a navbar alatti legördülő menüpont lenyitásával lépkedhetünk ugyanúgy mint a Graphs & Charts nézeten. Lásd **ide** kattintva.

Függetlenül attól, hogy éppen a Responses vagy az Image Frequencies nézeten vagyunk, az oldal felépítése megegyezik, a különbséget csak az adatok jelentik. Éppen ezért az alábbi szakaszban a nézeten megtalálható különböző funkciókat ismertetem.

3.4.1. Keresés

A kereséssel szűrés valósítható meg az adatokon, függetlenül az oszlopoktól, illetve, hogy hol helyezkedik el a keresett bemenet az adaton.

Export Settings

72

IMAGE_ID ↑	RESPONSE	SESSION_ID	TIME	HAND	MAJOR	CREATED_AT	
9915	3	bLKrl2QnFHOcjbSTK0xmX...	1572	Unknown hand	Unknown major	2024. 04. 16. 18:32:0...	
65119	2	D5sOynTw6GtV2aw8ACNM5...	1418	Unknown hand	Unknown major	2024. 04. 02. 3:31:32	

2 / 100

Show more

3.13. ábra. Data Listing nézeten keresés funkció

3.4.2. Rendezés

Az adattábla teljes fejléce kattintható, az adott oszlopnévre kattintva növekvő, valamint csökkenő módban rendezhetők az adatok.

Export Settings

72

IMAGE_ID ↑

RESPONSE

SESSION_ID

TIME

HAND

MAJOR

CREATED_AT

9915	3	bLKrl2QnFHOcjbSTK0xmX...	1572	Unknown hand	Unknown major	2024. 04. 16. 18:32:0...	
65119	2	D5sOynTw6GtV2aw8ACNM5...	1418	Unknown hand	Unknown major	2024. 04. 02. 3:31:32	

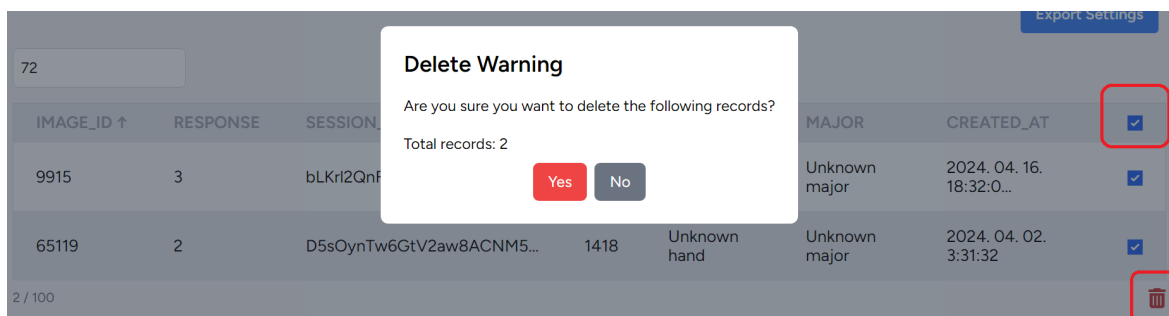
2 / 100

Show more

3.14. ábra. Data Listing nézeten rendezés funkció

3.4.3. Adattörlés

Lehetőség van az adatok törlésére, amelyet az adattábla legutolsó sora alatt található szemetes ikonra kattintva lehet megtenni. A törlendő adatokat az utolsó oszlopban található jelölőnégyzetek kiválasztásával lehet kiválasztani. Az összes adat kijelölésére a fejlécen található jelölőnégyzet ad lehetőséget. A szemetes ikonra kattintva egy felugró ablak jelenik meg, amely figyelmeztet a törlésről illetve törlendő adatok mennyiségéről.



3.15. ábra. Data Listing nézeten törlésről figyelmeztető felugró ablak

3.4.4. Adatok exportálása

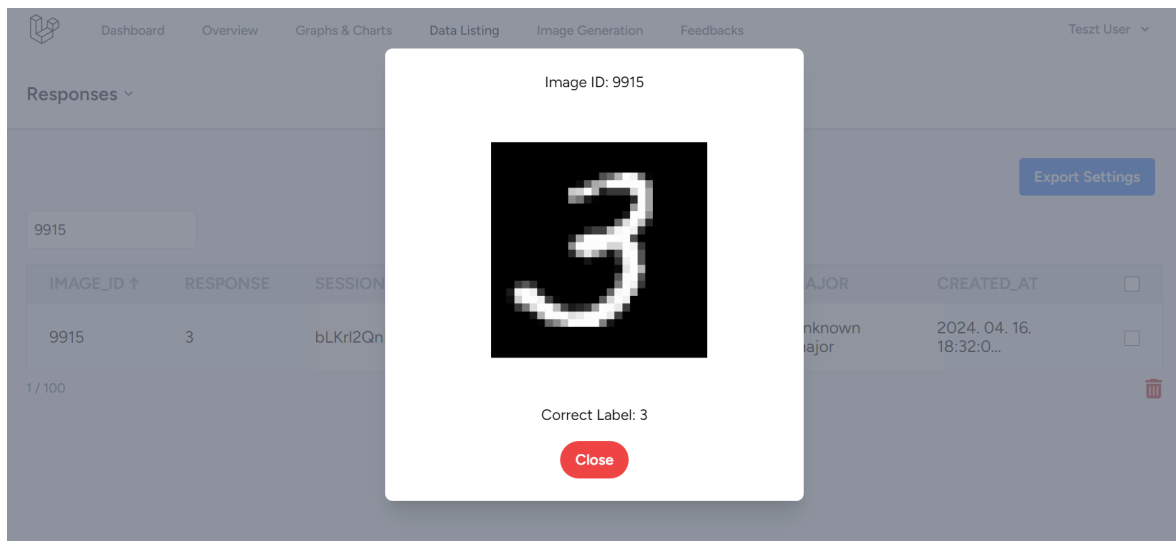
Az „Export Settings” gombra kattintva megjelennek a nézethez tartozó oszlopnevek, ahol a jelölőnégyzetek kiválasztásával konfigurálhatjuk az exportálni kívánt adatokat. Az exportálandó mezők kiválasztása után lehetőség van az adatok CSV fájlba való kimentésére.



3.16. ábra. Data Listing nézeten adatok exportálása funkció

3.4.5. Kattintható sorok

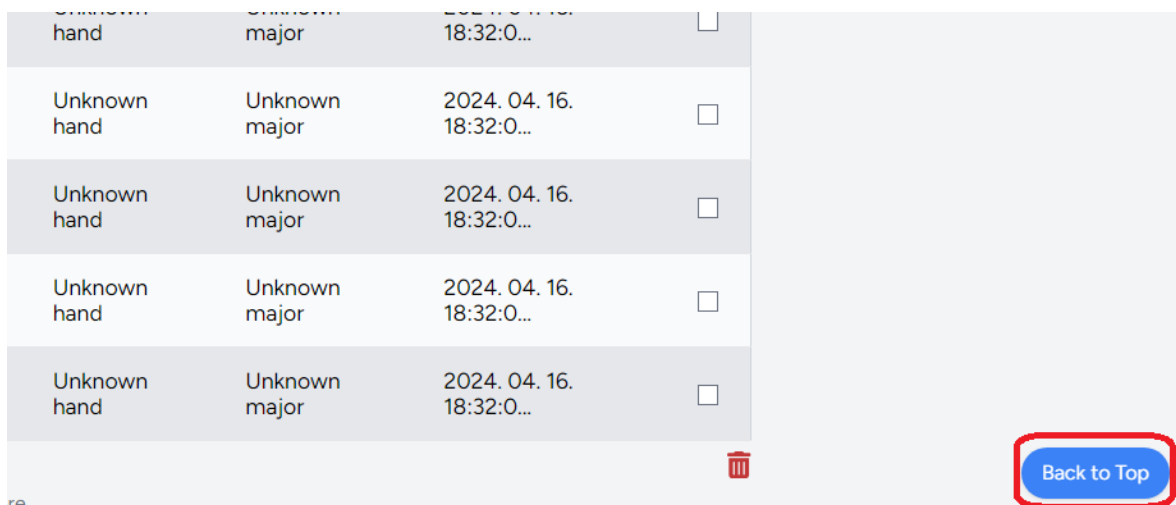
Lehetőség van a táblázaton belüli sorok kattintására, amely az adott sorhoz tartozó képet illetve az azt ábrázoló helyes számjegyet jeleníti meg egy felugró ablakban.



3.17. ábra. Data Listing nézeten egy sor kiválasztása

3.4.6. Ugrás a tetejére

A felhasználói élmény javításának érdekében az oldal jobb alsó sarkában az adattábla mellett egy „Back to Top” gomb jelenik meg, amint a görgetősávon (scroll bar) elhagynánk a kezdő pozíciót. Azaz, ha netalántán már a sokadik sornál járunk az adatok vizsgálatánál, de mondjuk egy rendezés vagy exportálás miatt újra az oldal tetején szeretnénk találni magunkat, a „Back to Top” gombra kattintva automatikusan az oldal tetejére visz fel a görgetősáv felfelé gördülő animációjával.



3.18. ábra. Data Listing nézeten ugrás a tetejére gomb

3.5. Image Generation

A **navigációs sávon** haladva az Image Generation nézeten a kérdőívben megjelenő képek manipulálására szolgáló vezérlőpanel jelenik meg. Itt van lehetősége az adminisztrátornak az alapértelmezetten randomizált képek egyféle „súlyozott random” beállítására. Három opció közül választhat, amelyek két további csoportra bonthatóak. Ezen generálási opciók különböző gyorsasággal bírnak, ami azt jelenti, hogy a véletlen manipulálásakor változhat a kitöltőknek megjelenő képek betöltési ideje annak függvényében, hogy az adott beállítás több megszorítást vagy kevesebbet tartalmaz.

Mivel ez a beállítás közvetlenül a kérdőívre van kihatással, ezért ennek bővebb felhasználói dokumentációját a szakdolgozati társam készítette el, amelyről bővebben az Ő szakdolgozatában lehet olvasni.

The screenshot shows a control panel for image generation with three main sections: Random, Balancing, and Most Often Misidentified. Each section has a description, a speed indicator (dots), and a 'Save' button. The 'Random' section is highlighted in blue. The 'Balancing' section has a checkbox for 'From Train dataset' and a radio button for 'From Test dataset'. The 'Most Often Misidentified' section has a speed indicator with 5 dots. A legend at the bottom right indicates 'More ● = Faster'.

Random	Balancing	Most Often Misidentified
Selects completely randomly. Image weighting is ignored.	Selects randomly from the least generated images based on the weights of previously generated images. Users won't receive the same image twice within an hour.	Selects randomly from the images that were most often misidentified based on their weights. Users won't receive the same image twice within an hour.
Speed: ●●●	Speed: ●●	Speed: ●●●●●
	<input checked="" type="checkbox"/> From Train dataset <input type="checkbox"/> From Test dataset	
<button>Save</button>		

More ● = Faster

3.19. ábra. Image Generation nézeten a képek manipulálására szolgáló vezérlőpanel

3.6. Feedbacks

Végül az utolsó oldal a visszajelzések kilistázásért felel. A kérdőívet kitöltve a felhasználóknak lehetőségük van visszajelzést adni, amelyet az adminisztrátor itt tud megtekinteni. A visszajelzés tartalmazza továbbá a küldő session id-ját illetve az elküldés pontos dátumát. Alapértelmezetten oldalankénti tíz különböző visszajelzés jelenik meg, ezenfelül lapozással lehet lépkedni köztük.

Feedbacks

Nagyon tetszik az oldal

2024-04-01 20:33:51

ZUk4HHow5pjRpYarRkec7Oh6qcUUK3nEHI6V4sQg

Szuper!

2024-04-01 20:34:57

5Ziuq9eUbuF8e23ey197IVciXWq6PZ2D5LHF7CPX

Jó lett

2024-04-01 20:42:17

a9DP4AQSNOEgXRZhtPcUdyf76ljctKpyXp4TOVKH

3.20. ábra. Feedbacks nézetén az oldalról kapott visszajelzések listája

4. fejezet

Fejlesztői dokumentáció

4.1. Inicializálás

A webalkalmazás Laravel keretrendszerben készült melyhez hozzátartozott egy kezdeti csomag integrálása. A csomag bemutatása előtt fontosnak tartom megemlíteni, hogy egy Laravel projekt elkészítéséhez szükség van a „Composer” nevű php csomagkezelőre, amely a projekthez szükséges csomagokat és függőségeket kezeli. Továbbá egy másik követelménye a „Node.js”, egy JavaScript futtatókörnyezet, amely lehetővé teszi a JavaScript eszközök és könyvtárak használatát a Laravel frontend fejlesztésében.

Ahogy azt említettem Laravel projekt létrehozása után beemeltem egy már előre elkészített úgynevezett „Laravel Breeze” csomagot, amely egy egyszerű, de hatékony autentikációs rendszer a keretrendszerhez. Ennek segítségével a regisztrációhoz és bejelentkezéshez szükséges backend logikához illetve frontend megvalósításhoz már volt egy kiindulási alapom. A frontendet React könyvtárral oldottam meg, mely ezen „Breeze” csomaggal rendkívül kompatibilis. Továbbá az inicializáláshoz tartozott az Inertia.js telepítése, amelyet a dinamikus oldalfrissítés megvalósításának céljából telepítettem.

4.2. Adatbázis táblák

Az inicializálás után az adatbázis táblák létrehozása a Laravel migrációkon keresztül történt. A Laravel migrációi lehetővé teszik a táblák és az adatbázis-struktúra egyszerű és hatékony kezelését, valamint az adatbázis frissítését. A migrációk segítségével könnyen létrehozhatóak, módosíthatóak vagy törölhetőek az adatbázis táblák.

Közvetlenül a statisztikák kimutatására használt adatbázis táblák a következők:

- responses
- image_frequencies
- misidentifications

- `number_frequencies`

Mindegyiknek megvan a maga jelentősége. A statisztikai oldalon túlnyomó részt ezeken a táblákon történnek a műveletek. Éppen ezért fejlesztői szemmel nézve a következőekben azokat a megoldásokat ismertetem, amelyek ezen táblákon érdekes vagy említésre méltó módon lettek implementálva.

Először is az egyértelműség érdekében röviden ismertetem ezeket a táblákat és legfontosabb mezőit.

4.2.1. responses

Feladata a válaszok mentése. Négy mezőjét emelném ki.

- **`image_id`**: az adott kép azonosítója `mnist_images` táblából.
- **`guest_response`**: a kitöltő válasza a képre.
- **`session_id`**: a kitöltők azonosítója, habár a kitöltések regisztráció nélkül történnek, de az azonosításra szükségünk van.
- **`response_time`**: a válaszadás időtartama (milliszekundumban kifejezve).

4.2.2. images_frequencies

Feladata a kérdőívben legenerált azonosításra váró képek számát, illetve a válaszok számát tárolja el.

- **`image_id`**: az adott kép azonosítója az `mnist_images` táblából.
- **`generation_count`**: a kép generálásának száma, alapértelmezett értéke 0.
- **`response_count`**: a képhez kapcsolódó azonosítások száma, alapértelmezett értéke 0.

4.2.3. misidentifications

Feladata, hogy számolja a képenkénti félreazonosításokat.

- **`image_id`**: az adott kép azonosítója az `mnist_images` táblából.
- **`correct_label`**: a kép helyes címkéje (ha úgy tetszik a „helyes” válasz), melyet az `mnist_images` táblából szerzünk meg.
- **`count`**: a félreazonosítások száma, alapértelmezett értéke 1, ugyanis csak akkor történik ide adatmentés ha megtörtént az adott képhez az első félreazonosítás.

4.2.4. number_frequencies

Feladata, hogy számon tartsa a leggenerált képek közül, hogy az adott számjegy hányszor szerepelt összesen.

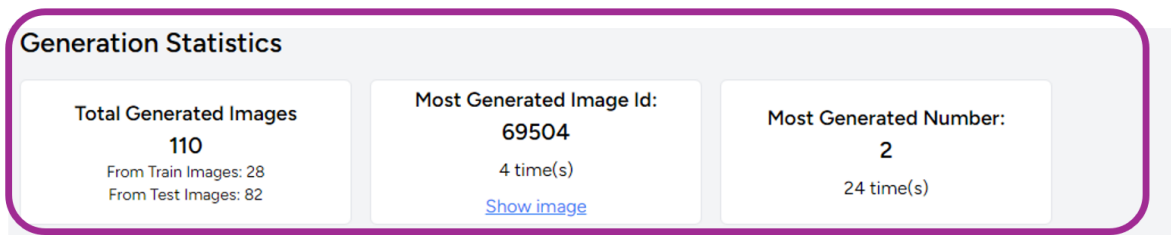
- **label**: az adott számjegy értéke, azaz összesen 10 lehetséges rekord (0-9-ig).
- **count**: az adott számjegy előfordulásának száma.

4.3. Összefoglaló nézet lekérdezései

A kiemelt és összefoglaló „**Overview**” nézet számára számos SQL lekérdezést hajt végre a backend, hogy adatokat gyűjtsön. Ezeket az SQL lekérdezéseket egy külön **OverviewController**-ben gyűjtöttem össze. A Laravel Controllerek a kód és az adatbázis lekérdezések kezelésének központi helyei. Feladataik közé tartozik az adatok előkészítése, a logika végrehajtása és az eredmények továbbítása a különböző nézetekhez az adatok megjelenítése érdekében.

Ezen különböző lekérdezéseket többnyire olyan műveletekkel valósítottam meg, mint szűrés, rendezés, összegzés. Ezekre a Laravelben beépített függvényeket használtam: **value()**, **whereBetween()**, **orderByDesc()**, **sum()**.

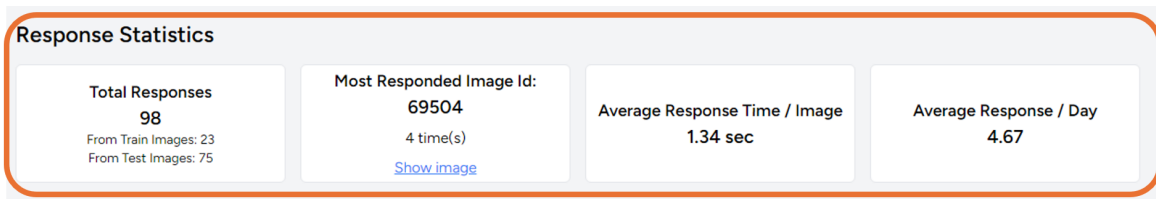
Az SQL lekérdezések mennyiségének bemutatása érdekében az alábbi képeken szemléltetem a különböző függvényhívásokat. A kód tisztán tartásának érdekében igyekeztem a kódismétlést elkerülve újrafelhasználható funkciókba szervezni a lehetséges lekérdezéseket.



```
$totalGeneratedImages = $this->getTotalGeneratedImages();  
$trainImagesCount = $this->getImagesCount(0, 59999);  
$testImagesCount = $this->getImagesCount(60000, 69999);  
$mostGeneratedNumber = $this->getMostGeneratedNumber();  
$mostGeneratedNumberCount = $this->getMostGeneratedNumberCount();  
$mostGeneratedImageId = $this->getMostGeneratedImageId();  
$mostGeneratedImageCount = $this->getImageCount($mostGeneratedImageId);  
$mostGeneratedImageData = $this->getImageData($mostGeneratedImageId);
```

Generation Statistics

4.1. ábra. Generation Statistics SQL lekérdezéseit megvalósító függvényhívások



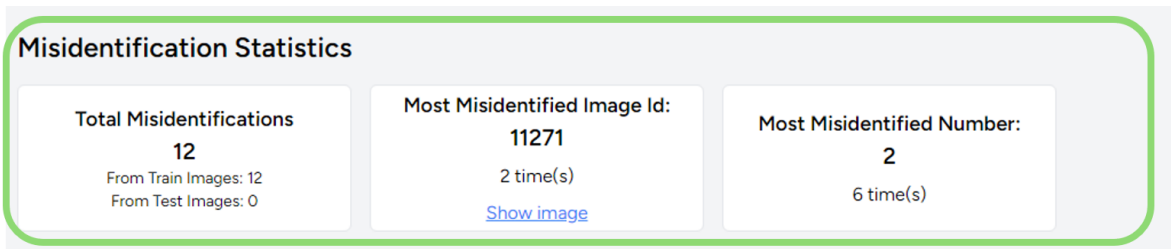
```

$totalResponses = $this->getTotalResponses();
$responsesFromTrain = $this->getResponsesCount(0, 59999);
$responsesFromTest = $this->getResponsesCount(60000, 69999);
$mostRespondedImageId = $this->getMostRespondedImageId();
$mostRespondedImageCount = $this->getImageCount($mostRespondedImageId);
$mostRespondedImageData = $this->getImageData($mostRespondedImageId);
$saverageResponseTime = $this->getAverageResponseTime();
$saverageResponsePerDay = $this->getAverageResponsePerDay();

```

Responses Statistics

4.2. ábra. Responses Statistics SQL lekérdezéseit megvalósító függvényhívások



```

$totalMisidentifications = $this->getTotalMisidentifications();
$misidentificationsFromTrain = $this->getMisidentificationsCount(0, 59999);
$misidentificationsFromTest = $this->getMisidentificationsCount(60000, 69999);
$mostMisidentifiedImageId = $this->getMostMisidentifiedImageId();
$mostMisidentifiedImageCount = $this->getImageCount($mostMisidentifiedImageId);
$mostMisidentifiedImageData = $this->getImageData($mostMisidentifiedImageId);
$mostMisidentifiedNumberCount = $this->getMostMisidentifiedNumberCount();
$mostMisidentifiedNumber = $this->getMostMisidentifiedNumber();

```

Misidentifications Statistics

4.3. ábra. Misidentifications Statistics SQL lekérdezéseit megvalósító függvényhívások

A lekérdezések közül az átlagos napi válaszok számát visszaadó függvényt (*getAverageResponsePerDay*) gondoltam részletesebben bemutatni, ugyanis ezt találtam érdekesebbnek a többitől, illetve ha már statisztikai feldolgozásról van szó, itt jelen van az átlagszámítás.

4.1. kód. *getAverageResponsePerDay()* function

```

1 private function getAverageResponsePerDay()
2 {
3     $firstDate = ImageFrequency::orderBy('created_at', 'asc')
4         ->value('created_at'); // Első előfordulási dátum
5     $lastDate = now(); // Aktuális dátum
6
7     $totalDays = $lastDate->diffInDays($firstDate) + 1; //
8         -> Az eltelt napok száma a két dátum között
9
10    $totalResponses = ImageFrequency::sum('response_count');
11    -> // Összesen a válaszok száma

```

```

9      // Átlag számítás kerekítéssel az adott feltétel alapján
10     if ($totalDays > 0) {
11         return round($totalResponses / $totalDays , 2);
12     } else {
13         return 0.00;
14     }
15 }

```

A kód részletes leírása: A kód harmadik sorában látható **\$firstDate** változóba mentem el az **image_frequencies** táblából lekérdezett legrégebben létrehozott kép dátumát, a **created_at** mező szerinti növekvő sorrendbe való rendezése után a legelső értéket kimentve. Majd a kód 4. sorában elmentem a **\$lastDate** nevű változóba az aktuális dátumot. A **\$totalDays** változóban kiszámolom az aktuális dátum és az **image_frequencies** táblából kiolvasott legrégebbi dátum közötti eltelt napok számát a **diffInDays()** függvénnyel. Továbbá +1 hozzáadásra kerül annak érdekében, hogy a kezdeti nap is figyelembe legyen véve ne csak a két dátum közötti különbség. Ezek után lekérem a **\$totalResponses** változóba a beérkezett válaszok számát. A szükséges adatok előkészítése után egy egyszerű vizsgálattal megnézem, hogy telt-e el már legalább 1 nap (ha úgy tetszik a **\$totalDays** értéke nagyobb-e mint 0), ha igen akkor történik az átlagos napi válaszadások kiszámítása úgy, hogy elosztom a válaszok számát az eltelt napok számával, majd az értéket 2 tizedesjegyre kerekítem. Abban az esetben, ha az **else** ágba futunk, *(ami csak akkor történhet meg, ha az **image_frequencies** tábla üres, mert ha legalább egy rekord is van a táblában, akkor már a **\$totalDays**-hez adott +1 érték miatt nem ide fut be)* ebben az esetben a 0.00 érték fog megjelenni az átlagos napi válaszok fölénél az Overview nézetben.

4.3.1. Components

A webfejlesztés során szinte elképzelhetetlennek tartom, hogy ne íránk ismétlődő kódot, viszont a kódismétlés nem előnyös, éppen ezért igyekeztem újrafelhasználható komponenseket létrehozni a fejlesztés során. Nem beszélve arról, hogy a komponensekbe való kód kiszervezése egyike a rugalmasabb kódírás betartásának. Tanulva a programozási technológiák egyik alapelvéből miszerint: „A program kódja állandóan változik!”.[18]

A Laravel Breeze telepítésével alapvető React komponensek jönnek létre, ilyenek például a **DropDown**, **CheckBox**, **Modal**, **InputError**, stb. Ezeket a különféle oldalnézetek megírása során újrafelhasználtam. Továbbá írtam saját komponenseket a különféle nézetű diagramokra, táblázatokra. Egyik ilyen általam megírt a **DataTable**, amelynek meghívását az alábbi kódon szemléltetem.

4.2. kód. DataTable komponens meghívása

```
1 <DataTable data={tableData} columns={columns} deleteRoute={  
  ↪ deleteRoute} onDataUpdate={handleDataUpdate}/>
```

Jól látható, hogy négy darab paramétert vár, melyek az adattábla rekordjai, az oszlop nevek, egy törlési útvonal, illetve a dinamikus adatfrissítést végző függvény. A **DataTable** komponensen belül a megkapott adatokat feldolgozó logika megírása után ennek a komponensnek a felhasználásával hoztam létre a Responses és az Image Frequencies táblázatot is. További komponenseket is létrehoztam például a különböző felugró ablakokat (Pop-ups), amelyek egy úgynevezett **Modal** komponensből lettek újrahasznosítva.

4.4. Graphs & Charts fejlesztői szemmel

Ebben a szakaszban a grafikonokhoz és diagramokhoz használt chart.js telepítéséről és használatáról írok röviden, illetve a hőtérkép megalkotásáról osztom meg a tapasztalataimat. Ezeket éreztem fejlesztői szemmel nézve érdekesebbnek és az átlagnál nehezebben megvalósíthatónak.

4.4.1. Oszlopdiaagram létrehozása

A webalkalmazásban használt oszlop és kördiagramhoz a **chart.js**-t használtam. Ennek a projektbe való telepítése az **npm** csomagkezelő segítségével történt a következő módon: „*npm install chart.js*”. Továbbá az általam használt React könyvtár több előre elkészített diagram komponenset kínál a chart.js-hez melyet az: „*npm install react-chartjs-2*” paranccsal van lehetőség telepíteni.

A react-chartjs-2-t használva az egyik ilyen előre elkészített komponens a **Bar** nevű oszlopdiaagram. A JavaScript kódban az alábbi importokat kell megtenni.

4.3. kód. importok

```
1 import { Bar } from 'react-chartjs-2';  
2 import 'chart.js/auto';
```

Az importok után lehetőség van a **Bar** komponens használatához, ahogy ezt az alábbi kód mutatja.

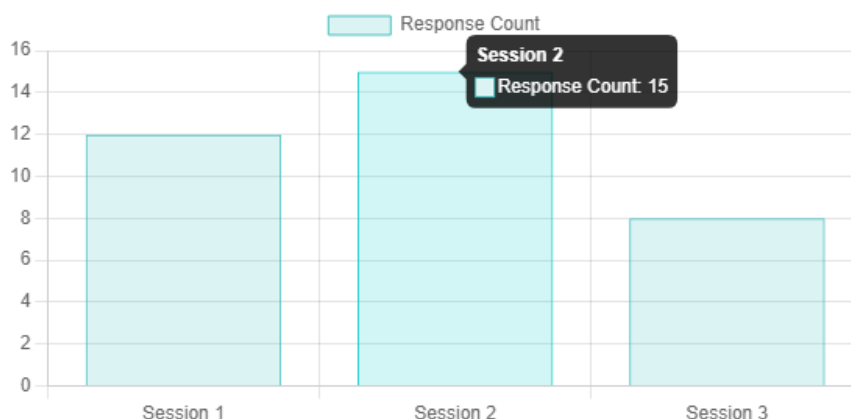
4.4. kód. BarChart.jsx

```

1 <Bar
2   data={{
3     labels: ['Session_1', 'Session_2', 'Session_3'], // X
      ↪ tengely értékei
4     datasets: [{
5       label: 'Response_Count', // diagram címe
6       data: [12, 15, 8], // Y tengely értékei
7       backgroundColor: 'rgba(75,192,192,0.2)', // háttérszín
8       borderColor: 'rgba(75,192,192,1)', // szegélyszín
9       borderWidth: 1, // szegély vastagsága
10    }],
11  }}
12 />

```

Az egyszerűség bemutatása kedvéért a harmadik sorban található **labels** és a hatodik sorban látható **data** tömböket lecseréltem példa adatokra. Valójában ezen tömbök a **responses** táblából kinyert adatokkal töltődnek fel. A példa oszlopdiagram az alábbi ábrán megtekinthető.



4.4. ábra. Példa oszlopdiagram

A webalkalmazásban hat különböző nézetű oszlopdiagram van elkészítve hasonlóan a fentebbi kódhoz. Továbbá a kördiagram megvalósítását is a React könyvtár komponensei közül megtalálható **Pie** használatával hoztam létre.

4.4.2. Hőtérkép megvalósítása

A hőtérképet egy rendkívül jó és kompakt szemléltetési módszernek találom az azonosítások és félreazonosítások kimutatására. Az általam elképzelt hőtérkép több lépésben működik, melyet az alábbi felsorolásban ismertetek.

1. **Adatlekérdezés és előkészítés:** backenden egy `calculateLabelCounts()` nevű függvényben elkészíték egy asszociatív tömböt, amely kulcs-érték párosításból áll össze. Az első szint kulcsai 0-tól 9-ig terjedő számok. Majd minden kulcshoz egy újabb tömb tartozik, amely tartalmazza a `guest_response` mező előfordulási számait. Ez a tömb 0-tól 9-ig terjed. Ha úgy tetszik eltároljuk a 0-tól 9-ig számjegyre történt összes azonosítást. Szemléltetésként az alábbi kódban láthat egy példát.

4.5. kód. A hőtérképhez előkészített asszociatív tömb példája

```
1  [
2      0 => [5, 0, 0, 2, 3, 1, 0, 0, 0, 0],
3      1 => [0, 3, 1, 4, 0, 0, 2, 0, 0, 0],
4      2 => [1, 2, 7, 0, 0, 0, 0, 0, 0, 0],
5      3 => [0, 0, 0, 5, 2, 0, 0, 1, 0, 0],
6      4 => [0, 1, 0, 2, 8, 0, 0, 0, 0, 1],
7      5 => [0, 0, 0, 0, 0, 4, 0, 0, 0, 0],
8      6 => [0, 0, 0, 0, 0, 0, 3, 0, 0, 0],
9      7 => [0, 0, 0, 0, 0, 0, 0, 6, 1, 0],
10     8 => [0, 0, 0, 0, 0, 0, 0, 0, 10, 0],
11     9 => [0, 0, 0, 0, 0, 0, 0, 0, 0, 9]
12 ]
```

2. **Adattovábbítás a Python szkriptnek:** a következő lépésben továbbra is a backenden maradva egy újabb függvény a `generateHeatmap()` először meghívja a már elkészített `calculateLabelCounts()` függvényt, amelynek az eredményét JSON formátumra alakítja, majd átadja egy `mnist_heatmap.py` nevű python script-nek amelyet a webalkalmazáson belül a Laravel `/storage/scripts` nevű mappájában hoztam létre.
3. **Adatfeldolgozás, heatmap generálás, visszaküldés:** a python script feldolgozza a JSON formátumban megkapott adatokat, egy 10x10-es mátrixot készít belőle, majd a **Matplotlib** könyvtár segítségével elkészíti a hőtérképet. Az elkészített hőtérképből egy PNG képet készít, amelyet base64 formátumra konvertál, hogy visszatudja küldeni az őt meghívó `generateHeatmap()` függvénynek kiemeneként.
4. **API útvonal:** a `generateHeatmap()` függvényhívásra elkészítettem egy új API útvonalat: „`/statistics/heatmap`”
5. **Adatlekérdezés és képmegjelenítés:** a frontenden található `fetchHeatmapImage()` függvény lekéri a backendtől az adatokat egy AJAX kéréssel: `axios.get()`. A lekérés eredményéből a kinyert base64 formátumú képet megjeleníti a frontend felületen, **lásd ide kattintva**.

4.5. Adatok törlése

Az adatok törlésére a „Data Listing” nézetén, azaz a táblázatos megjelenítésnél van lehetőség. Mint azt a **Data Listing szakaszban** is említettem két fő nézet szerepel a táblázatos kilistázásnál, egyik a Responses másik az Image Frequencies nézet.

Gondolhatnánk, hogy miért van szükség törlésre hiszen mégis csak statisztikai adatokról van szó. Egyetértek abban, hogy ezáltal veszíthet a végső hitelességén, de voltak érvek amik mégis a törlés funkció bevezetése mellett szóltak. Egyik, hogy mi van akkor ha hiába a felmérés elején található CAPTCHA, valakinek sikerül azt megugorva botolni a válaszokat. A másik, ha valakinek már látványosan több félreazonosítása van, mint amennyi az ésszerűség határán belül van, mondhatni szándékosan rongálja a mérési eredményeket. Éppen ezért úgy gondoltam, hogy az adminisztrátori nézetén legyen lehetőség az adatok törlésére.

A törlés soft delete módon van megoldva, azaz amikor egy rekordot nem véglegesen törölünk az adatbázisból, hanem egyszerűen csak inaktívnak jelöljük azt. Ezt úgy érjük el, hogy egy új „nullable”¹ mezőt adunk az adott táblához például „deleted_at” néven és a törlés időpontját, pontos dátumát tároljuk el benne. Abban az esetben, ha még nincs törölve a rekord egyszerűen „NULL” érték szerepel benne. Ez a fajta megoldás lehetőséget biztosít arra, hogy bármikor újra aktívvá tegyük a törölt adatokat.

A Laravel természetesen támogatja az adatok törlésének ezen módját. Az adatbázis táblák létrehozásakor a táblához tartozó migrációs fájlban szerepelnie kell egy **softDeletes()** mezőnek, ez már automatikusan nullable lesz, lásd az alábbi kód 8. sorában.

4.6. kód. deleted_at oszlop hozzáadása a responses migrációs fájlban

```
1 return new class extends Migration
2 {
3     public function up()
4     {
5         Schema::create('responses', function (Blueprint $table)
6             ↪ {
7             $table->id();
8             ... // A többi oszlop
9             $table->softDeletes();
10        });
11    }
12    public function down(): void
13    {
14        Schema::dropIfExists('responses');
15    }
16};
```

¹ nullable - képes NULL értéket befogadni az adott oszlop

Majd az adott táblához tartozó Model osztályban importálni kell a `SoftDeletes` trait-et², illetve a „use” kulcsszóval használni is azt, **lásd ide kattintva a kód 7. és 12. soraiban**. Érdeemes, adott esetben szükséges ellátni a `protected $dates = ['deleted_at'];` sorral ugyanis ez felel majd a dátumformátum beállításáért, **lásd a kód 16. sorában**. Innentől kezdve a Controllerben elvégzett `delete()` metódus nem ténylegesen törli az adatot, hanem beállítja a „deleted_at” oszlopot az aktuális időponttal. Az adatok visszaállítására egyszerűen a `restore()` metódussal van lehetőségünk.

4.7. kód. soft delete beállítása a responses Model osztályban

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Database\Eloquent\SoftDeletes;
8
9 class Response extends Model
10 {
11     use HasFactory;
12     use SoftDeletes;
13
14     protected $fillable = [ 'image_id', 'guest_response', '
        ↳ session_id', 'response_time' ];
15
16     protected $dates = [ 'deleted_at' ];
17
18 }
```

4.6. Telepítés

A webalkalmazás telepítéséhez[21] található egy útmutató, amiről ezen a linken: **<https://github.com/vamosakos/mnist-validate-by-human>** egy github repository-ban olvashat.

² trait - kód újrafelhasználásának mechanizmusa az olyan egyszeres öröklődési nyelvekben, mint a PHP[20]

5. fejezet

Tesztelés

5.1. Cypress tesztelés

A webalkalmazás teszteléséhez egy automatizált Cypress tesztet készítettem el. A Cypress egy modern tesztelési keretrendszer, amely lehetővé teszi a fejlesztők számára a webalkalmazások funkcionális tesztelését. Az alkalmazásba való cypress keretrendszer integrálása egyszerűen az `npm install cypress -save-dev` paranccsal megtehető. Majd az indításhoz az `npx cypress open` parancsot kell futtatni.

A Cypress tesztelésem során a hangsúlyt főként a „Graphs & Charts” nézeten lévő funkcióinak ellenőrzésére fektettem. Beleértve a keresés, rendezés, felugró ablakok, elemek kattintását, valamint az elvárt eredmények ellenőrzését.

5.1. kód. Graphs & Charts nézet tesztelése Cypressel

```
1 describe('Graphs & Charts Test', () => {
2   it('A Graphs & Charts oldalon teszteli a rendezést, keresést
   ↪ és a különböző diagram nézeteket', () => {
3     // Bejelentkezés
4     cy.visit('http://127.0.0.1:8000/login');
5     cy.get('#email').type('test1@test.com');
6     cy.get('#password').type('12345678');
7     cy.contains('button', 'Login').click();
8
9     // 'Graphs & Charts' oldalra navigálás
10    cy.get('a[href="http://127.0.0.1:8000/statistics/
       ↪ responses-charts"]').should('be.visible').click();
11    cy.wait(2000);
12
13    // 'Sort' gombra kattintás kétszer a csökkenő és növekvő
       ↪ sorrend ellenőrzéséhez
14    cy.get('button').contains('Sort').should('be.visible').
       ↪ click();
15    cy.wait(2000);
```

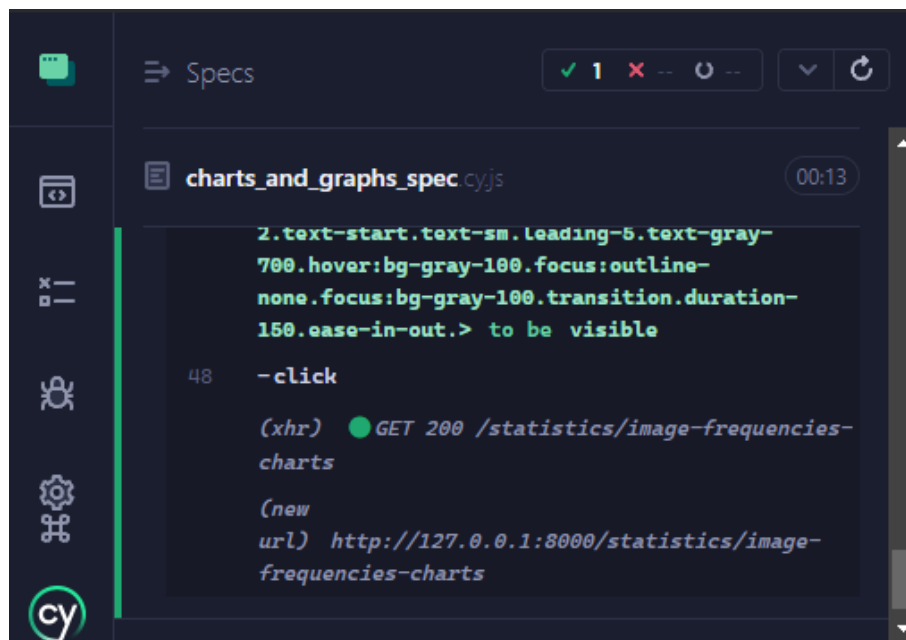
```

16     cy.get('button').contains('Sort').should('be.visible').
        ↪ click();
17     cy.wait(2000);
18
19
20     // 'Display Option' legördülő menüből 'Average Response
        ↪ Time by Image' oszlopdiagram nézet kiválasztása
21     cy.get('#displayOption').select('Average□Response□Time□
        ↪ by□Image');
22
23     // 'Search' mezőbe '70000' beírása és 'Show Image'
        ↪ gombra kattintás
24     cy.get('#searchInput').type('70000');
25     cy.contains('button', 'Show□Image').click();
26
27     // Hibaüzenet ellenőrzése
28     cy.contains('Please□enter□a□valid□image□ID□(maximum□
        ↪ value□is□69.999)').should('be.visible');
29     cy.wait(2000);
30
31     // 'Search' mezőbe '100' beírása és 'Show Image' gombra
        ↪ kattintás
32     cy.get('#searchInput').clear().type('100');
33     cy.contains('button', 'Show□Image').click();
34     cy.wait(2000);
35
36     // 'Image ID: 100' és 'Correct Label:' szövegek
        ↪ ellenőrzése, amely azt jelzi, hogy a kép
        ↪ megjelenítése sikeres volt
37     cy.contains('Image□ID:□100').should('be.visible');
38     cy.contains('Correct□Label:').should('be.visible');
39
40     // A képet megjelenítő Pop-up bezárása
41     cy.contains('button', 'Close').click();
42
43     // Átnavigálás az 'Image Frequencies' nézetre
44     cy.get('.flex.items-center').contains('Responses').click
        ↪ ();
45     cy.get('a[href="http://127.0.0.1:8000/statistics/image-
        ↪ frequencies-charts"]').should('be.visible').click
        ↪ ();
46     });
47     });

```

- A tesztelés egy bejelentkezéssel indul, majd a navigációs sávon haladva a „Graphs & Charts” nézetre navigál át (4-11. sor).

- Az oldalra érve a „Sort” (Rendezés) gomb megléte esetén kattintás, majd egy rövid két másodperces várakozás után ismét kattintás a csökkenő és növekvő sorrend ellenőrzéséhez (13-17. sor).
- A diagram nézetek közötti váltáshoz megkeresi a legördülő menüt, ahol az „Average Response Time by Image” nézetet választja ki (21. sor).
- A kereső mező teszteléséhez először egy hibás, nem létező kép azonosítóra keres rá, amelyre a hibaüzenet meglétét várja. Majd egy helyes azonosítóra keresve a kép megjelenítés gombot választja (24-34. sor).
- A felugró ablakon ellenőrzi, hogy helyes képet kapott-e meg, majd a „Close” gomb kattintására bezárja azt (37-41. sor).
- Végül a Responses és Image Frequencies nézetek közötti váltást a navigációs sáv alatt található legördülő menü kiválasztásával az Image Frequencies nézet diagramjainak útvonalára ugrik (44-45. sor).



5.1. ábra. Cypress teszt lefutás után

A „Graphs & Charts” nézeten lévő tesztelés kimeríti a főbb funkciók tesztelését. Ugyanis a keresés, rendezés, felugró ablakok ugyanazzal a módszerrel, komponenssel lettek implementálva.

6. fejezet

Továbbfejlesztési ötletek

6.1. Személyre szabhatóság

A felhasználói élmény javításának érdekében a diagramok kicsinyítése és nagyítása funkciók megléte segítené az adott esetben zsúfolt diagramok olvasását. Ezzel együtt a különböző grafikonok „grid-en” belüli áthelyezése például „drag & drop” funkcióval bevezetve úgy gondolom, hogy tovább fokozná az élményt.

Szinek tekintetében is látok különböző fejlesztési lehetőségeket. Jelenleg az oldal stílusa a világos letisztult vonalat követi, viszont manapság a sötét mód egy igen népszerű választás a felhasználók körében.

6.2. Lekérdezések optimalizálása

Habár nagy mennyiségű adatokkal nem volt alkalmam tesztelni az alkalmazást, így elképzelhető teljesítmény romlás. Ezért úgy gondolom, hogy a különböző SQL lekérdezésekben van helye az optimalizálásnak.

6.3. Terheléses tesztek

Az előző szakaszból kifolyólag fontos ismerni az alkalmazás határait, szélsőségeit. Érdemes lenne nagy mennyiségű adatokkal tesztelni az alkalmazás működőképességét.

Összegzés

Összességében elégedett vagyok a munkámmal. Visszagondolva a fejlesztésre élveztem a vele eltöltött időt. Biztos vagyok benne, hogy az egyéni tudásom fejlesztésében nagy szerepe volt a dolgozat elkészítésének. Megtapasztaltam milyen az, ha egy nagy adat rengetegből kell dolgozni majd azt hatékonyan elemezni és feldolgozni. Elmondható, hogy a problémamegoldó képességem is fejlődött, mivel meg kellett oldani az adatbázis tervezésével, adatfeldolgozással és megjelenítéssel kapcsolatos kihívásokat.

A dolgozat témáját tekintve úgy gondolom, hogy nem egy gyakori esetet fed le. Mégis csak egy gépi tanulásra szánt adathalmaznak, mondhatni speciálisabb megközelítésről van szó, annak ellenére, hogy a nap végén egy felmérésből kimutatott statisztikai összefoglalást kapunk.

A webalkalmazás felületén igyekeztem a fontosabb és hasznosabb információkat összegyűjteni a felmérésből bejövő adatokból. Ehhez hasonló módon jártam el a szakdolgozati dokumentáció írása közben is, ahol a számomra releváns tapasztalataimat foglaltam össze. Bízom benne, hogy ez sikerült, ahogy abban is, hogy a webalkalmazás segíti majd a témában érdekeltek kutatását.

Végezetül, hálás vagyok mindenkinek, aki támogatott ebben a folyamatban. Külön köszönet a konzulensemnek és szakdolgozati társamnak, valamint a támogató szüleimnek és az egyetemi éveim során megismert tanárainknak.

Irodalomjegyzék

- [1] LEXIQ: *MNIST adatbázis*
URL: <https://lexiq.hu/mnist-adatbazis>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [2] ORACLE: *What is database?*
URL: <https://www.oracle.com/database/what-is-database/>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [3] SULINET: *A statisztika fogalma*
URL: <https://tudasbazis.sulinet.hu/hu/szakkepzes/kereskedelem-es-marketing/kereskedelmi-es-marketing-modulok/a-statisztika-fogalma-feladatai-a-statisztika-kapcsolata-mas-tudomanyokkal/a-statisztika-fogalma>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [4] LINKEDIN: *History of Statistics*
URL: <https://www.linkedin.com/pulse/history-statistics-dr-subhabaha-pal>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [5] LINKEDIN: *Statistical reports*
URL: <https://www.linkedin.com/advice/1/what-some-best-practices-tips-writing-clear>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [6] IBM: *What is data visualization?*
URL: <https://www.ibm.com/topics/data-visualization>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [7] PLANTUML: *Use Case Diagram*
URL: <https://plantuml.com/use-case-diagram>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14

- [8] DBDIAGRAM: *dbdiagram*
URL: <https://dbdiagram.io/home>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [9] MYSQL: *mysql*
URL: <https://www.mysql.com>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [10] W3SCHOOLS: *What is HTML?*
URL: https://www.w3schools.com/html/html_intro.asp
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [11] W3SCHOOLS: *What is CSS?*
URL: https://www.w3schools.com/css/css_intro.asp
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [12] PHP.NET: *What is PHP?*
URL: <https://www.php.net/manual/en/intro-what-is.php>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [13] JSX: *Writing Markup with JSX?*
URL: <https://react.dev/learn/writing-markup-with-jsx>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [14] BLOG.HUBSPOT: *Tailwind CSS*
URL: <https://blog.hubspot.com/website/what-is-tailwind-css>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [15] CHART.JS: *Chart.js docs*
URL: <https://www.chartjs.org/docs/latest/>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [16] INERTIA.JS: *Inertia.js*
URL: <https://inertiajs.com/>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [17] ATlassian: *Microservices vs. monolithic architecture*
URL: <https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [18] DR. KUSPER GÁBOR ÉS DR. RADVÁNYI TIBOR: *Jegyzet a projekt labor című tárgyhöz*
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14

- [19] LARAVEL: *Meet Laravel*
URL: <https://laravel.com/docs/11.x>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [20] PHP.NET: *Traits*
URL: <https://www.php.net/manual/en/language.oop5.traits.php>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14
- [21] GITHUB: *github repository*
URL: <https://github.com/vamosakos/mnist-validate-by-human>
UTOLSÓ MEGTEKINTÉS DÁTUMA: 2024.04.14

NYILATKOZAT

Alulírott *Vámos Ákos János* büntetőjogi felelősségem tudatában kijelentem, hogy az általam benyújtott, *Webalkalmazás fejlesztése az MNIST adatbázis humán validációjához* című szakdolgozat (diplomamunka) önálló szellemi termékem. Amennyiben mások munkáját felhasználtam, azokra megfelelően hivatkozom, beleértve a nyomtatott és az internetes forrásokat is.

Tudomásul veszem, hogy a szakdolgozat elektronikus példánya a védés után az Eszterházy Károly Katolikus Egyetem könyvtárába kerül elhelyezésre, ahol a könyvtár olvasói hozzájuthatnak.

Kelt: Eger, 2024. április 14.

.....*Vámos Ákos János*.....

aláírás