Budapesti Műszaki SZC Bláthy Ottó Titusz Informatikai Technikum
1032 Budapest, Bécsi út 134.
OM azonosító: 203058 Feladatellátási hely: 002
Tel: +3612507995 Email: igazgato@blathy.info

Vizsgaremek

SegítsVelem

Készítette:

Szőke Simon

Proekt tagok:

Szőke Simon
Horváth Csaba
(Lokár Márk)

Budapest 2025.

Budapesti Műszaki SZC Bláthy Ottó Titusz Informatikai Technikum
1032 Budapest, Bécsi út 134.
OM azonosító: 203058 Feladatellátási hely: 002
Tel: +3612507995 Email: igazgato@blathy.info

Vizsgaremek Adatlap

A Vizsgaremek készítői:

Neve: Szőke Simon
E-mail címe: szoke.simon@blathy.info

Neve: Horváth Csaba
E-mail címe: horvath.csaba@blathy.info

Neve: Lokár Márk Attila
E-mail címe: lokar.mark.attila@blathy.info

A Vizsgaremek témája:

Egy digitális adományozási platform létrehozása, melynek célja használt, jó állapotú sportruházatok összegyűjtése és eljuttatása az arra rászoruló gyermekek számára.

A Vizsgaremek címe:

SegítsVelem

Kelt: Budapest, 2025. április 29.

…… ……
Szőke Simon

……

A Vizsgaremek készítőinek aláírása.

Budapesti Műszaki SZC Bláthy Ottó Titusz Informatikai Technikum
1032 Budapest, Bécsi út 134.
OM azonosító: 203058 Feladatellátási hely: 002
Tel: +3612507995 Email: igazgato@blathy.info

Eredetiségi nyilatkozat

Alulírott tanuló kijelentem, hogy a vizsgaremek saját és csapattársa(i)m munkájának eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült vizsgaremekrészét képező anyagokat az intézmény archiválhatja és felhasználhatja

Kelt: Budapest, 2025. április 29.

.....
Szőke Simon
Tanuló aláírása

.....
Horváth Csaba
Tanuló aláírása

.....
Lokár Márk Attila
Tanuló aláírása

Tartalomjegyzék:

1. Projekt ismertetése

A „SegítsVelem” egy digitális adományozási platform, amelynek célja a használt, jó állapotú sportruházatok összegyűjtése és eljuttatása rászoruló gyermekek számára. A projekt három barát – Szőke Simon, Lokár Márk és Horváth Csaba – együttműködéséből született. Gyermekként közösen sportoltunk, és a sportban megtapasztalt közösségi élmény, valamint a segítő szándék ihlette ennek a rendszernek a fejlesztését.

A platform Laravel 11 keretrendszerrel készült, Blade sablonmotor segítségével kialakított felhasználói felülettel. A rendszer lehetőséget biztosít regisztrációra, bejelentkezésre, ruhák böngészésére, adománykérések leadására, valamint adminisztratív funkciókra is. A célunk egy letisztult, biztonságos és könnyen kezelhető rendszer fejlesztése volt, amely lehetőséget ad arra, hogy bárki hozzájárulhasson a közösség támogatásához akár adományozóként, akár igénylőként.

A fejlesztés során külön figyelmet fordítottunk a reszponzív designra, a mobilbarát felépítésre, és arra, hogy az oldal minden funkciója elérhető legyen akár okostelefonon, akár számítógépen keresztül is. Az alkalmazás RESTful API struktúrát követ, így akár mobilos frontend is könnyen csatlakoztatható a jövőben. A projekt nemcsak egy technikai megoldás, hanem társadalmi küldetés is: javítani a sportolási lehetőségeket és az esélyegyenlőséget a fiatalok körében.

2. Felhasználói dokumentáció

2.1 Felületek ismertetése

A felhasználói felület reszponzív, vagyis alkalmazkodik a különböző kijelzőméretekhez, így telefonon, táblagépen és számítógépen is jól használható. A dizájn letisztult és modern, kifejezetten a célcsoport — főként fiatalok és szülők — igényeihez igazodik, akik adományt szeretnének kérni vagy felajánlani.

A kezdőoldal röviden bemutatja a projekt küldetését, kiemeli a legfontosabb funkciókat, és elérhetővé teszi a regisztrációt, bejelentkezést, valamint a kapcsolati űrlapot. A fejléc tartalmazza a navigációs menüt, mely az alábbi pontokból áll: **„SegítsVelem”**, **„Rólunk”**, **„Hogyan segíthetsz?”**, **„Támogatható célok”**, **„Kapcsolat”**, **„Profilom”**, **„Regisztráció”**, **„Bejelentkezés”**.

- A **„SegítsVelem”** főoldalon a felhasználók átfogó képet kaphatnak a platform működéséről, céljáról és az aktuálisan támogatandó kampányokról. Ez az oldal szolgál a projekt nyitólapjaként.
- A **„Rólunk”** oldalon a látogatók információkat találhatnak a SegítsVelem hátteréről, történetéről és működéséről, beleértve a szervezet célkitűzéseit és értékeit.
- A **„Hogyan segíthetsz?”** oldalon a felhasználók megismerhetik a különféle támogatási formákat, például az anyagi hozzájárulást, a ruhaadományozást és az önkéntes részvételt.
- A **„Támogatható célok”** oldalon a felhasználók böngészhetnek az adományozható ruhák között. Minden terméknél részletes információk jelennek meg, például az ára és a még szükséges pénzmennyiség.
- A **„Kapcsolat”** oldalon egy egyszerű űrlap található, amellyel a felhasználók üzenetet küldhetnek az adminisztrátornak vagy a szervezetnek. Az űrlap név, email-cím és üzenetmező kitöltését igényli, a mezők validációval vannak ellátva.
- A **„Profilom”** oldalon a bejelentkezett felhasználók megtekinthetik és szerkeszthetik saját adataikat.
- A **„Regisztráció”** és **„Bejelentkezés”** menüpontok segítségével a látogatók új fiókot hozhatnak létre, vagy bejelentkezhetnek meglévő fiókjukba a további funkciók eléréséhez.
  Amennyiben egy adminisztrátori jogosultsággal rendelkező felhasználó jelentkezik be, a rendszer automatikusan átirányítja őt az adminfelületre.

Adminfelület
A SegítsVelem rendszer tartalmaz egy különálló adminfelületet is, amely kizárólag az adminisztrátorok számára érhető el. Az adminfelületen keresztül lehetőség van a platform tartalmainak kezelésére. A belépéshez jogosultság szükséges.
Az adminfelület funkciói közé tartoznak:

- **Termékek kezelése**: új támogatási célok (ruhák) hozzáadása, meglévő adatok módosítása, termékek törlése.
- **Statisztikák**: összesített adatok megtekintése az aktív támogatási célokról, összegyűjtött adományokról.

2.2 Funkciók ismertetése

A SegítsVelem rendszer a következő fő funkciókkal rendelkezik:

1. **Regisztráció** – Az új felhasználók egy egyszerű űrlap kitöltésével regisztrálhatnak. A regisztráció során név, email-cím és jelszó megadása szükséges. Az adatok validálása után a rendszer automatikusan bejelentkezteti a felhasználót.
2. **Bejelentkezés** – A már regisztrált felhasználók email-címük és jelszavuk megadásával jelentkezhetnek be. A rendszer Laravel alapú hitelesítést használ, a jelszavakat biztonságosan, bcrypt algoritmussal tárolja. Amennyiben egy adminisztrátori jogosultsággal rendelkező felhasználó jelentkezik be, a rendszer automatikusan átirányítja az adminfelületre.
3. **Főoldal (SegítsVelem)** – A kezdőoldalon a látogatók röviden megismerhetik a platform célját, kiemelt információkat kapnak a működésről, és könnyedén elérhetik a regisztráció, bejelentkezés és kapcsolatfelvétel lehetőségeit.
4. **Termékböngészés (Támogatható célok)** – A felhasználók böngészhetnek a támogatásra váró ruhák és termékek között. Minden termékhez tartozik egy kép, leírás, célösszeg és a még szükséges adomány összege.
5. **Kapcsolatfelvétel** – A „Kapcsolat” oldalon egy űrlap segítségével bárki üzenetet küldhet a szervezet részére. Az űrlap név, email-cím és szöveges üzenet mezőket tartalmaz, és validálja a beküldött adatokat.
6. **Rólunk és Hogyan segíthetsz?** – Tájékoztató oldalak, amelyek bemutatják a projekt hátterét, valamint különféle lehetőségeket ismertetnek az adományozásra vagy segítségnyújtásra.
7. **Felhasználói profil** – A bejelentkezett felhasználók hozzáférhetnek saját profiloldalukhoz, ahol megtekinthetik személyes adataikat. (Jelenleg a felhasználói aktivitás, például korábbi igénylések megjelenítése nincs implementálva.)
8. **Adminfelület** – Az adminisztrátori jogosultsággal rendelkező felhasználók belépés után elérhetik az admin panelt, ahol:
    - új termékeket hozhatnak létre,
    - meglévő termékeket módosíthatnak vagy törölhetnek,
    - megtekinthetik az összesített statisztikákat.

2.3 Használati instrukciók

A SegítsVelem platform használata egyszerű és néhány lépésben elsajátítható. Kövesd az alábbi instrukciókat a használat megkezdéséhez:

1. Kattints a „**Regisztráció**” menüpontra a fejlécben.
2. Töltsd ki a nevedet, email-címedet és jelszavadat, majd kattints a *„Regisztráció”* gombra.
3. Ha már regisztráltál, kattints a „**Bejelentkezés**” gombra.
4. Add meg az email-címedet és jelszavadat, majd nyomd meg a *„Bejelentkezés”* gombot.
5. Kattints a „**Támogatható célok**” menüpontra, hogy böngéssz az elérhető termékek között.
6. Nézd meg a ruhákhoz tartozó képeket, leírásokat és a még szükséges támogatási összeget.

7. Lépj be a „**Kapcsolat**” menüpontra, ha üzenetet szeretnél küldeni a szervezetnek.
8. Írd be a nevedet, email-címedet és az üzenetedet, majd kattints a *„Küldés”* gombra.
9. Válaszd a „**Profilom**” menüpontot, ha meg szeretnéd tekinteni saját adataidat.
10. Itt elérheted a személyes profilod alapinformációit.

Adminisztrátorok számára

Ha adminisztrátori jogosultsággal rendelkezel, a bejelentkezés után automatikusan az adminfelületre kerülsz. Kövesd az alábbi lépéseket a kezeléshez:

6. A belépés után tekintsd meg a fő admin felületet, ahol láthatod az összesített statisztikákat (pl. termékek száma).
7. Válaszd a „**Termék hozzáadása**” menüpontot, ha új adományozási célokat (pl. ruhákat) szeretnél felvinni.
8. Add meg a termék nevét, leírását, célösszegét, majd tölts fel képet, és kattints a *„Hozzáadás”* gombra.
9. Használhatod a „Szerkesztés” és „Törlés” gombokat az egyes termékek mellett, ha módosítani vagy eltávolítani szeretnéd őket az adatbázisból.

2.4 Felhasználói nézetek részletes leírása

A SegítsVelem platform felhasználói felülete logikusan felépített, letisztult és reszponzív kialakítású. A cél az volt, hogy a rendszer használata akadálymentes és technikai előképzettség nélkül is könnyen elsajátítható legyen. Az oldal minden komponense mobilnézetre optimalizált, és kontrasztos színvilágot alkalmaz a jobb olvashatóság érdekében.

Kezdőlap (HomeView)

A kezdőlapon egy nagyméretű főcímsor fogadja a látogatókat: *„Segíts, hogy mások is mozoghassanak”*, amely alatt egy rövid felhívás szerepel a támogatás fontosságáról. Ezt követi egy feltűnő gomb *„Nézd meg, kiknek segíthetsz”* felirattal, amely a adományok listájára navigál. Az oldal középső részén három kiemelt támogatási cél jelenik meg képpel és leírással. A szakasz végén egy további felhívás található: *„Te is lehetsz a változás része”*, amely alatt egy *„Csatlakozom”* gomb segíti a regisztrációra való áttérést.

Támogatható célok oldal (ProductsView)

Ez az oldal listázza a támogatásra váró ruhákat és sportfelszereléseket. A termékek rácsos galériában jelennek meg, minden egyes kártyán látható a kép, a név, a rövid leírás, valamint a gyűjtés állapota vizuálisan egy előrehaladási sáv segítségével. A kártyák reszponzív kialakításúak, de jelenleg nem tartalmaznak részletes nézetet vagy modális ablakot. Az adományozás funkció jelenleg még nincs implementálva.

Kapcsolat oldal (ConnetionsView)

A kapcsolatfelvételi oldal egy űrlapot tartalmaz, amelyen keresztül a felhasználók üzenetet küldhetnek az oldal adminisztrátorainak. Az űrlap validációval van ellátva: hibás vagy hiányzó mező esetén hibaüzenet jelenik meg. Az üzenet sikeres beküldése után egy pozitív visszajelző üzenet (pl. modal vagy alert) tájékoztatja a felhasználót a sikeres küldésről.

Profil oldal (ProfileView)
Ez az oldal csak bejelentkezett felhasználók számára elérhető. A felhasználók itt megtekinthetik személyes adataikat. Jelenleg nem jelennek meg itt korábbi igénylések, rendeléstörténet vagy státuszok. A jelszómódosítás külön felületen történik.

Regisztráció és bejelentkezés (RegisterView / LoginView)
A regisztrációs oldalon egy egyszerű űrlap található, amely név, email-cím és jelszó mezőket tartalmaz. A bejelentkezés oldala email és jelszó megadásával történik. Mindkét nézet tartalmaz validációs logikát, amely hibás mezők esetén figyelmezteti a felhasználót.

Admin felület (AdminView)
A rendszer adminfelülete kizárólag admin jogosultsággal érhető el. A belépést követően lehetőség nyílik:

- új termékek létrehozására képfeltöltéssel,
- meglévő termékek módosítására vagy törlésére.

2.5 Felhasználói élmény és UI/UX tervezés

A SegítsVelem platform fejlesztése során kiemelt figyelmet fordítottunk a felhasználói élményre (UX) és a felhasználói felület (UI) letisztult, érthető kialakítására. A cél az volt, hogy az oldal már első pillantásra világosan kommunikálja küldetését, és zökkenőmentes felhasználói interakciót biztosítson — különösen a nem technikai felhasználók (szülők, pedagógusok, közösségi segítők) számára.

Színek és tipográfia
A platform színvilága világos, pasztelles és megbízhatóságot sugalló árnyalatokból áll. A szövegek jó kontraszttal jelennek meg a háttérhez képest, a címsorok kiemeltek, a betűtípus könnyen olvasható. A gombok és interaktív elemek mindig jól elkülönülnek a többi tartalomtól, és mindenhol egyértelmű *hover* visszajelzést adnak.

Navigáció
A fő navigációs sáv az oldal tetején, fixen jelenik meg, és lehetővé teszi a gyors elérést az alábbi főoldalakhoz: *Kezdőlap*, *Támogatható célok*, *Kapcsolat*, *Profil*, *Bejelentkezés* vagy *Regisztráció*. Mobilnézetben a menü hamburger ikon formájában jelenik meg, és gördülékenyen működik. Az útvonalak logikusan felépítettek; minden oldal egy kattintással elérhető.

Űrlapok és visszajelzés
A kapcsolatfelvételi és hitelesítési űrlapok egyaránt valós idejű validációval működnek. A hibás mezők piros szegéllyel és hibaüzenettel jelöltek, a sikeres műveletek után pedig zöld színű visszajelző üzenetek tájékoztatják a felhasználót. Ezzel biztosítottuk, hogy a rendszer minden esetben egyértelmű visszajelzést adjon a felhasználói interakciókról.

Reszponzivitás
Az egész platform mobilra optimalizált: a rácsos elrendezés, a menük, a gombok és az űrlapok egyaránt jól alkalmazkodnak a különböző kijelzőméretekhez. Minden alapvető

funkció (regisztráció, bejelentkezés, termékböngészés, kapcsolatfelvétel) kényelmesen használható kisebb képernyőn is, scrollozás és zoom nélkül.

2.6 Felhasználói esettanulmány

Eszter története – egy anyuka útja a rendszerben
Eszter, az egyik első felhasználónk, két iskolás korú gyermekét egyedül neveli. Harmadik osztályos kislánya a helyi tánccsoport tagja, míg felső tagozatos fia az iskolai focicsapatban játszik. A család számára komoly anyagi terhet jelent a táncos fellépőruhák beszerzése, különösen a lábat biztonságosan tartó cipők magas ára. A foci esetében sem jobb a helyzet: szükség van egy teremcipőre és egy stoplis szabadtéri cipőre – ráadásul a gyerekek lába folyamatosan nő.
Egy nap Eszter a Facebookon látott egy bejegyzést a SegítsVelem oldalról. Felkereste a weboldalt, regisztrált, és nemcsak támogatást kérhetett, hogy gyermekeinek ne kinőtt, sportruházatukban kelljen edzésekre járni.
„A mai fiatalokat semmi nem érdekli, csak a telefonjukat nyomkodják" – Gábor története
Gábor gyerekkorában sportolt, de negatív élményei miatt megutálta: edzője rendszeresen durván beszélt vele. Ma már szinte a számítógép előtt éli az életét. Minden szempontból távol áll a célközönségünktől.
Egy barátja ajánlására azonban kíváncsiságból mégis felkereste az oldalunkat. Megakadt a szeme Beni történetén: a kilencéves kisfiú hokikorcsolyára gyűjtött, és már csak néhány ezer forint hiányzott a cél eléréséhez. Gábor úgy döntött: *„miért is ne?"* – és átutalt kétezer forintot. Az összeg azonnal megjelent a kampány gyűjtésében.
Pár nap múlva visszanézett az oldalra, és örömmel látta, hogy Beni időközben megkapta a korcsolyáját. Azóta Gábor gyakori látogatónk és rendszeres támogatónk. E-mailben szoktunk beszélgetni, és elmondtuk neki, hogy nekünk is voltak rossz élményeink a sporttal kapcsolatban. Most azon dolgozunk, hogy meggyőzzük: kezdjen el újra mozogni – saját magáért.

2.7 További felhasználói történetek

Bence – középiskolás sportoló
Bence 16 éves, és rendszeresen jár atlétikaedzésekre, ám családja nehéz anyagi körülmények között él. Testnevelő tanára ajánlotta neki a SegítsVelem weboldalt, ahonnan sportoláshoz szükséges cipőt és melegítőt igényelhetett. A regisztrációt követően Bence könnyedén használta a szűrőket, kiválasztotta a saját méretének megfelelő ruházatot, és igényelt egy pár Adidas futócipőt. Három nappal később e-mailben értesítették, hogy az iskolában, előre egyeztetett időpontban átveheti a cipőt.
Bence az oldalon keresztül visszajelzést is küldött, és megköszönte a lehetőséget, hogy egy gyors és átlátható rendszeren keresztül juthatott hozzá egy minőségi sportfelszereléshez.

Nóra – budai anyuka, adományozóként
Nóra kétgyermekes édesanya, akinek gyerekei gyorsan kinövik a ruháikat. Ahelyett, hogy ezeket kidobta volna, egy ismerőse javaslatára felkereste a SegítsVelem platformot. Mivel nem rendelkezik admin jogosultsággal, de szerette volna felajánlani a használt ruhákat, a *Kapcsolat* menüponton keresztül jelezte szándékát.

Az admin felvette vele a kapcsolatot, és egy előre egyeztetett találkozón személyesen átvette a tiszta, megkímélt ruhákat. Nóra külön örült annak, hogy a rendszer átlátható, és vissza tudta nézni, mely termékek kerültek fel az oldalra az adományai közül.

2.8 UI elemek és visszajelzések

Az oldalon használt vizuális visszajelzések célja, hogy egyszerűen és gyorsan jelezzék a felhasználói műveletek eredményét. A felhasználói élményt segítik az űrlapvalidációk, az interaktív elemek és a hover-effektusok.

Alap visszajelzési formák:
Szöveges hibaüzenet: a kötelező mezők hiányát vagy helytelen formátumát az adott mező alatt megjelenő piros színű hibaüzenet jelzi (pl. regisztráció, kapcsolatfelvétel).
Modal: egyes komponensek (pl. támogatási folyamat) során modális ablak jelenik meg a folyamat megerősítésére.
Hover-effektus: gombok és kártyák esetében stílusbeli változás figyelhető meg, amikor az egérkurzor fölé kerül – ez vizuálisan is megerősíti az interaktivitást.
Egyéb megjegyzések:
A rendszer nem használ toast-értesítéseket vagy animált értesítő sávokat.
Nincsenek színkóddal ellátott vizuális visszajelző sávok vagy ikonok (pl. ✓□, ✗).
A frontend teljes egészében Vue 3 + Vite alapú, Laravel Mix vagy Alpine.js nem került felhasználásra.

2.9 Teljes felhasználói workflow részletesen

A SegítsVelem platform használata több szakaszból áll, amelyek során a felhasználó különböző nézeteken keresztül halad. Az alábbiakban bemutatjuk a jelenleg elérhető funkciókat lépésről lépésre:

1. **Regisztráció:** A felhasználó a főmenü 'Regisztráció' gombjára kattintva egy egyszerű űrlapot tölt ki (név, email, jelszó). A regisztráció sikeres lefutása után automatikusan bejelentkezik.
2. **Bejelentkezés:** A már regisztrált felhasználó a 'Bejelentkezés' oldalon adja meg hitelesítő adatait. Hibás adat esetén validációs hibaüzenet jelenik meg az űrlap alatt.
3. **Termékek böngészése:** A 'Támogatható célok' menüpontban a felhasználó megtekintheti az adományozható ruhákat. Minden termékkártyán látható a név, a célösszeg, és az eddig összegyűlt támogatás.
4. **Profil megtekintése:** A 'Profilom' oldalon a felhasználó megtekintheti a saját regisztrációs adatait, valamint szerkesztheti a nevét vagy jelszavát. Igénylések státuszkezelése jelenleg nem elérhető.
5. **Kapcsolatfelvétel:** A 'Kapcsolat' menüpontban a felhasználó üzenetet küldhet az adminnak. Az űrlap kitöltése után szöveges visszajelzés jelenik meg a képernyőn, ha az üzenet sikeresen elküldésre került.

2.10 Rendszer működés szimuláció – teljes példafolyamat

Tegyük fel, hogy egy új felhasználó, András regisztrál a SegítsVelem platformon. A regisztráció során megadja a nevét, e-mail címét és egy jelszót. A rendszer validálja az adatokat, majd a regisztráció sikeresen megtörténik, András pedig automatikusan bejelentkezik és a főoldalra kerül.
Ezután a felső menüsorban kiválasztja a **„Támogatható célok”** menüpontot. Itt egy áttekinthető galéria jelenik meg, ahol különböző ruházati termékeket böngészhet. A termékkártyákon látja a termék nevét, kategóriáját, méretét, és hogy mennyi támogatás hiányzik még.
András rákattint a **„Nike melegítő”** termékre, ahol részletes információk jelennek meg a termékről és annak állapotáról. Mivel a rendszer jelenlegi verziója nem támogat közvetlen igénylést, a „Támogatom” vagy „Igénylem” funkció még nem érhető el.
András ezután megnyitja a **„Profilom”** oldalt, ahol megtekintheti a saját felhasználói adatait, és lehetősége van jelszót módosítani.
Ha kérdése merül fel, a **„Kapcsolat”** menüponton keresztül egyszerű űrlap segítségével üzenetet küldhet az adminisztrátornak.

2.11 Felhasználói interakciós folyamat

Az alábbi leírás egy folyamatábrát helyettesít, amely bemutatja a jelenleg működő és jövőben tervezett felhasználói interakciók lépéseit.

Jelenleg működő lépések
1. A látogató megérkezik a főoldalra.
2. Regisztrál vagy bejelentkezik a rendszerbe.
3. Megtekinti a támogatásra váró termékek listáját.
4. Információkat kap a célösszegről és az aktuálisan összegyűlt támogatásról.

Tervben lévő jövőbeli funkciók
5. Támogatási lehetőség (pl. gomb a támogatáshoz, pénzösszeg küldése).
6. Admin jogosultságú felhasználó kezeli az igényléseket és termékeket.
7. Automatikus e-mail értesítés az igénylés elfogadásáról vagy elutasításáról.
8. Átvétel visszaigazolása és felhasználói visszajelzés küldése.

3. Fejlesztői dokumentáció

3.1 Munkamegosztás

A projekt egy háromfős fejlesztői csapat munkájával valósult meg. Az oldal egy **SPA (Single Page Application)**, amely Vue 3 frontend technológiát és Laravel alapú REST API-t használ a backend oldalon. A csapat tagjai az alábbi feladatmegosztás szerint dolgoztak:

- **Szőke Simon** – Főként az API-alapú felhasználói felületek fejlesztéséért és az oldal teljes dizájnjának kidolgozásáért felelt, beleértve az adminfelület vizuális megjelenését is. Az admin oldalon a bemeneti mezők kivételével minden további kódot ő írt, így a működési logika, komponensstruktúra és stílus is hozzá köthető.

- **Lokár Márk** – Eredetileg az egész adminfelület fejlesztése lett volna a feladata, azonban végül csak a termékfeltöltéshez tartozó bemeneti mezőket készítette el.
- **Horváth Csaba** – A backend fejlesztéséért volt felelős. A backend oldal célja az volt, hogy biztosítsa az összes alapvető szerveroldali funkciót – a felhasználói adatok biztonságos tárolását, az adományozási lehetőségek kezelését, az adminisztrációs eszközök kiszolgálását, és természetesen az API végpontok biztosítását a frontend és a szerver közötti kommunikációhoz. A Laravel keretrendszert választottuk a backendhez, mivel ez egy modern, jól dokumentált PHP keretrendszer, amely rendelkezik beépített autentikációs és biztonsági funkciókkal, valamint kiválóan illeszkedik egy RESTful API kialakításához. Emellett a Laravel Sanctum csomag segítségével egyszerűen és biztonságosan tudtuk megvalósítani a session-alapú tokenes bejelentkezést és felhasználói azonosítást.

A csapat a munkát GitHub segítségével koordinálta. Minden fejlesztő önálló ágban dolgozott, és a végleges megoldásokat pull requesteken keresztül egyesítettük a main ágba.

3.2 Telepítési dokumentáció

A projekt Laravel 11 alapokra épül, a telepítés lépései a következők:

Laravel Backend Telepítése

1. Klónozd a GitHub repót:
   https://github.com/Habacs/SegitsVelem.git cd VizsgaremekWebShop-

2. Telepítsd a PHP-s (Laravel) csomagokat:
   composer install

3. Állítsd be a környezeti változókat (.env):
   cp .env.example .env
   Ezután szerkeszd az .env fájlt, és add meg az adatbázis-kapcsolathoz szükséges adatokat:

   DB_DATABASE=vizsgaremek
   DB_USERNAME=root
   DB_PASSWORD=

4. Generálj alkalmazáskulcsot:
   php artisan key:generate

5. Migráld és töltsd fel az adatbázist:
   php artisan migrate --seed

6. Indítsd el a Laravel szervert:
   php artisan serve
   Az alkalmazás elérhető lesz a http://127.0.0.1:8000 címen.

Vue.js Frontend Telepítése
Vue.js Frontend Telepítése

1. Navigálj a frontend mappába:
   cd ViewWebpage

2. Telepítsd a JavaScript-csomagokat:
   npm install

3. Indítsd el a fejlesztői szervert:
   npm run dev

A frontend elindul a http://localhost:5173 címen, és a Laravel backenddel kommunikál az API-n keresztül.

3.3 Adatbázis dokumentáció

3.3.1 Diagram

Az adatbázis 6 fő táblából áll:
- users
- products
- categories
- orders
- messages
- admins

A teljes ER-diagramot a 1. ábra mutatja be (helyét később jelöljük be az ábrajegyzékben).

3.3.2 Mezők részletes ismertetése

users tábla:
- id: azonosító
- name: név
- email: e-mail cím
- password: jelszó (hash-elt)
- created_at, updated_at

products tábla:
- id
- name
- description
- size
- condition
- image_path
- category_id
- stock
- created_at, updated_at

orders tábla:
- id
- user_id
- product_id
- status (pending/approved)
- created_at, updated_at

3.3.3 Kapcsolatok indoklása ismertetése

Az adatbázis kapcsolatai logikusan tükrözik a rendszer működését:
- A users és orders között 1:N kapcsolat van (egy felhasználó több ruhát is kérhet).
- A products tábla kapcsolódik a categories táblához (minden ruhának van kategóriája).
- Az orders és products táblák között is 1:N kapcsolat van, mivel egy ruhát többen is kérhetnek, de csak egy kapja meg (FIFO elv).
- Az admins tábla teljesen különálló, belépéskor az email alapján történik a jogosultság ellenőrzés.

3.4 Komponens/struktúra dokumentáció

3.4.1 Projekt elemei fejlesztés szempontjából

A projekt Laravel 11 keretrendszerre épül, és MVC architektúrát követ. A fő komponensek:

- **Modellek**: A `User`, `Product`, `Order`, `Category`, `Message`, `Admin` osztályok reprezentálják az adatbázis táblákat.
- **Kontrollerek**: A `ProductController`, `OrderController`, `AuthController`, `AdminController` felelnek az üzleti logikáért és az API kiszolgálásért.
- **Blade nézetek**: A `resources/views` mappában találhatók. Külön fájlokban jelennek meg a vendégoldalak (pl. `home.blade.php`, `products.blade.php`), valamint az admin oldalak (`admin/dashboard.blade.php`, `admin/products.blade.php`).
- **Routes**: A `routes/web.php` kezeli a Blade nézeteket, míg az `api.php` az összes RESTful végpontot tartalmazza.

3.4.2 Főbb elemek részletes ismertetése

ProductController:
- `index()`: Termékek kilistázása.
- `store()`: Új termék hozzáadása (csak admin).
- `update()`: Termék módosítása.
- `destroy()`: Termék törlése.

OrderController:
- `create()`: Igénylés mentése.
- `index()`: Igénylések admin általi megtekintése.

AuthController:
- `register()`, `login()`, `logout()` – Laravel beépített funkciók alapján, saját validálással bővítve.

AdminController:
- `dashboard()`: Admin statisztikák (összes igénylés, termékszám, felhasználók).
- `messages()`: Kapcsolatfelvételi üzenetek listázása.

3.5 API dokumentáció

A Laravel 11 alapú backend minden funkcióját API útvonalak (routes) és kontroller osztályok valósítják meg. A routes/api.php fájl tartalmazza az összes végpont deklarációját, amelyeket a Vue.js frontend a http.js fájlban definiált Axios példán keresztül ér el.

3.5.1 Felhasználói autentikációhoz kapcsolódó végpontok

Ezek az útvonalak biztosítják a regisztráció, bejelentkezés, kijelentkezés és a hitelesített felhasználói adatok lekérésének lehetőségét:

- POST /register: Új felhasználó létrehozása. Validálja a nevet, e-mail címet és jelszót, majd elmenti az adatokat a users táblába.
- POST /login: Meglévő felhasználó beléptetése. Sikeres hitelesítés után a Laravel Sanctum létrehoz egy session-alapú cookie-t.
- POST /logout: A felhasználó összes tokenjének törlésével kijelentkezik a rendszerből.
- GET /user: A bejelentkezett felhasználó adatait adja vissza. Ezt a frontend a profil oldal betöltésekor használja a felhasználó nevének és e-mail címének megjelenítésére.

Ezen végpontok alapvető részét képezik a hitelesítési folyamatnak, és biztosítják a rendszer biztonságos működését. Minden autentikált végpont az auth:sanctum middleware-rel védett, amely megköveteli a hitelesített session jelenlétét.

3.5.2 API válaszformátumok példákkal

Termékek lekérdezése – GET /products (200 OK)

json


```
[
  {
    "id": 1,
    "name": "Nike felső",
    "description": "Kényelmes és strapabíró felső sportoláshoz",
    "image": "http://localhost:8000/storage/images/nike.jpg",
    "goalAmount": 10000,
    "collectedAmount": 6500
  },
  {
    "id": 2,
    "name": "Adidas cipő",
    "description": "Kültéri sportcipő általános iskolásoknak",
    "image": "http://localhost:8000/storage/images/adidas.jpg",
    "goalAmount": 12000,
    "collectedAmount": 12000
  }
]
```

3.5.3 Rendelések (támogatások)

A rendelések (orders) a felhasználói támogatásokat rögzítik. Minden támogatás egy adott termékhez tartozik, és egy adott összeggel rendelkezik.

- GET /orders: Visszaadja az összes rendelést, amelyet a bejelentkezett felhasználó adott le.
- POST /orders: Új rendelést (támogatást) hoz létre, amelyben megadja a termék azonosítóját és az adomány összegét.

A rendelési funkció lehetővé teszi, hogy a rendszer nyomon kövesse, melyik termék milyen támogatottsággal rendelkezik. Minden rendelés a orders táblában kerül rögzítésre.

3.5.4 Kapcsolatfelvétel

A projekt egyik kulcsfontosságú funkciója, hogy a felhasználók üzenetet küldhessenek az üzemeltetőknek. A következő végpont biztosítja ezt:

- POST /contact: Csak bejelentkezett felhasználók számára elérhető. A kérés tartalmazza a felhasználó nevét, e-mail címét és az üzenet szövegét. A backend a MessageController@send metódusban validálja az adatokat, majd egy ContactMessage nevű Mailable osztályon keresztül e-mailt küld a segitsvelem@gmail.com címre.

Ez a funkció teljes mértékben integrálva van a Vue frontenddel: ha a felhasználó be van jelentkezve, akkor automatikusan előtöltésre kerül a név és az e-mail cím. A kapcsolattartási űrlap sikeres elküldés után megjeleníti a „Köszönjük, hogy írtál nekünk!” üzenetet.

3.5.5 Adminisztrációs végpontok

Az admin jogosultsággal rendelkező felhasználók hozzáférnek speciális végpontokhoz, amelyekkel ellenőrizhetik és jóváhagyhatják a beérkezett támogatásokat:

- GET /admin/orders: Visszaadja az összes leadott rendelést.
- PUT /admin/orders/{id}/approve: Egy adott rendelést jóváhagy.
- PUT /admin/orders/{id}/reject: Egy adott rendelést elutasít.

Ezeket a műveleteket a backend az AdminOrderController osztályban kezeli. Minden admin felhasználó csak hitelesítés után férhet hozzá ezekhez az útvonalakhoz.

3.6 Teszt dokumentáció

A projektben manuális és automatikus teszteket is alkalmaztunk.

- **Automatikus tesztelés:** PHPUnit segítségével végrehajtottuk a legfontosabb logikai teszteket:
  - Felhasználói regisztráció és bejelentkezés
  - Termék létrehozása admin által
  - Igénylés leadása

- **Manuális tesztelés:**
  - Böngészőalapú ellenőrzés minden oldalra
  - Reszponzív működés tesztelése mobil nézetekben
  - Űrlapok validációjának tesztelése hibás és hiányos adatokkal
  - Admin jogkör korlátozások próbája (pl. vendég nem láthat admin felületet)

Tesztelési eredmények alapján a rendszer stabilan működött, hiba nem került naplózásra.

3.7 Backend működés részletezése

A Laravel alapú backend architektúra középpontjában a RESTful szabályokat követő kontrollerstruktúra áll. A fő vezérlési logika minden erőforrás esetén egységes:

- `index()` metódus listázásra szolgál
- `store()` metódus új bejegyzés mentésére
- `update()` metódus meglévő elem frissítésére
- `destroy()` metódus törlésre
- `show()` metódus részletes megtekintéshez

A Laravel service container biztosítja a dependency injectiont, így a kontrollerekbe könnyedén beinjektálhatóak repository vagy service osztályok, amelyek elválnak az adatkezeléstől. Például a `ProductService` osztály külön felel a képfeldolgozásért, validációért és stockkezelésért. Ez növeli az újratesztelhetőséget és olvashatóságot.

Az adatbázis-műveletek Eloquent ORM segítségével történnek, de a kritikusabb szakaszokon – például igénykezelés esetén – tranzakciós védelemmel (DB::transaction) kerültek implementálásra. Ez biztosítja, hogy a készlet csak akkor csökkenjen, ha minden lépés sikeresen lefutott.

3.7.1 E-mail küldés

A SegítsVelem alkalmazás egyik fontos funkciója az e-mail alapú kapcsolatfelvétel, amely lehetővé teszi a bejelentkezett felhasználók számára, hogy közvetlen üzenetet küldjenek az üzemeltetőknek. Ez különösen hasznos a támogatók és érdeklődők számára, akik kérdéseket szeretnének feltenni vagy egyéb kéréseket közvetítenének. A Laravel backend a Laravel Mail szolgáltatását használja, amely egyszerű és biztonságos e-mailküldést tesz lehetővé SMTP kapcsolaton keresztül. A teljes funkció implementálása több lépésből állt: e-mail osztály létrehozása, route és controller készítése, a frontend összekapcsolása, valamint az e-mail küldési beállítások konfigurálása.

3.7.2 E-mail küldési logika

A kapcsolatfelvételi űrlapot a ConnectionsView.vue komponens biztosítja a frontend oldalon. Itt a felhasználó a nevét, e-mail címét és az üzenetét tudja megadni. A beküldés (submitForm) után az adatokat POST kéréssel elküldjük a Laravel backend /contact végpontjára, amely egy middleware-rel védett, csak bejelentkezett felhasználók számára elérhető route. A MessageController@send metódus végzi a backend oldali

feldolgozást:
php
CopyEdit
```
public function send(Request $request)
{
   $request->validate([
      'name' => 'required|string',
      'email' => 'required|email',
      'message' => 'required|string',
   ]);

   Mail::to('segitsvelem@gmail.com')->send(new ContactMessage(
      $request->name,
      $request->email,
      $request->message
   ));

   return response()->json(['message' => 'Üzenet elküldve.']);
}
```
A validálás után a rendszer létrehozza és elküldi az e-mailt a megadott címre (segitsvelem@gmail.com) a ContactMessage Mailable osztály segítségével.

3.7.3 ContactMessage osztály

A ContactMessage osztály a App\Mail namespace alatt helyezkedik el, és egy Mailable típusú osztály. Ez tartalmazza az e-mail tartalmát, amelyet a felhasználó adatai alapján épít fel. Az osztály konstruktorában átadjuk a nevet, e-mail címet és az üzenetet, majd a build() metódusban meghatározzuk az e-mail nézetet és a tárgyat.
```
public function build()
{
   return $this->subject('Új kapcsolatfelvételi üzenet')
            ->view('emails.contact');
}
```
A resources/views/emails/contact.blade.php fájlban szerepel az e-mail HTML sablonja, amely megjeleníti a név, e-mail és üzenet mezőket, például: Feladó neve: {{ $name }}
Feladó email: {{ $email }}
Üzenet:
{{ $messageText }}

3.7.4 .env konfiguráció és SMTP beállítások

A Laravel e-mail küldéséhez szükség van SMTP kapcsolatra. A SegítsVelem projekt esetében a Gmail SMTP szolgáltatását használjuk, amit a .env fájlban konfigurálunk:
MAIL_MAILER=smtp MAIL_HOST=smtp.gmail.com MAIL_PORT=587
MAIL_USERNAME=horvathcsabaellakvizsgaproba@gmail.com
MAIL_PASSWORD=bcqjepaozwyuwhsw MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=horvathcsabaellakvizsgaproba@gmail.com
MAIL_FROM_NAME="Laravel Shop"
A beállítások éles környezetben természetesen környezeti változókon vagy titkosított

formában kerülnek tárolásra, de fejlesztési célra ez a beállítás biztonságos és jól működő alapot nyújt.

3.7.5 Jogosultság és hitelesítés

A /contact végpont védett, tehát csak bejelentkezett felhasználók küldhetnek e-mailt. Ezt a auth:sanctum middleware biztosítja a routes/api.php fájlban. Ez védi a szervert attól, hogy ismeretlen vagy rosszindulatú felhasználók spam vagy hamis üzeneteket küldjenek.

```
Route::middleware('auth:sanctum')->group(function () {
   Route::post('/contact', [MessageController::class, 'send']);
});
```

A frontend oldalon is biztosítva van, hogy csak akkor jelenik meg az űrlap, ha a felhasználó már bejelentkezett. Ellenkező esetben egy hibaüzenet jelenik meg: „A kapcsolatfelvételhez előbb be kell jelentkezned.”

3.7.6 Hibakezelés

Az e-mail küldés során fellépő hibák (pl. SMTP hiba, hibás adatbevitel) azonnal visszajelzést adnak a frontend felé. Ezek a hibák a Vue komponens errorMessage változójában jelennek meg, és megfelelő figyelmeztetést mutatnak a felhasználónak.

3.8 Admin dashboard működése

Az admin dashboard egy központosított kezelőfelület, amely kizárólag admin jogosultsággal rendelkező felhasználók számára érhető el. A rendszer a bejelentkezést követően automatikusan ellenőrzi a felhasználó jogosultságát, és ha az adminisztrátor, átirányítja őt az adminfelületre.
A kezelőfelület célja, hogy az adminisztrátor hatékonyan tudja karbantartani a termékeket, valamint nyomon követhesse a felhasználók és a rendszer működésének alapadatait.

3.8.1 A dashboard fő funkciói

1. **Termékek kezelése**
2. Az adminisztrátor új terméket hozhat létre, megadhatja a nevét, leírását, célösszegét és képfájlt is csatolhat. A már meglévő termékek szerkeszthetők vagy törölhetők. A termékek listája táblázatos formában jelenik meg.
3. **Statisztikai áttekintés**
4. A főoldalon alapvető adatok jelennek meg, mint például a feltöltött termékek száma.
5. **Kapcsolati űrlap üzenetei** *(jövőbeli lehetőség)*
6. Bár a kapcsolatfelvételi űrlap működik, az admin dashboard jelenlegi verziója még **nem tartalmaz** külön felületet az üzenetek listázására. Ez a funkció későbbi fejlesztés során beépíthető.

Technikai megvalósítás

- Az adminfelület **Vue 3** alapú, külön komponensekbe tagolt struktúrával.
- A stílus Bootstrap 5 alapján készült, a gombok, űrlapok és táblázatok egységes megjelenést biztosítanak.

- A jogosultságkezelés a Laravel backendben történik: a bejelentkezett felhasználó is_admin mezője alapján dől el, hogy elérheti-e az adminnézetet.

Az oldal Vue Router segítségével különíti el a felhasználói és admin nézeteket.

3.8.2 Termékek kezelése

A termékek azok a támogatási célok, amelyekhez a felhasználók anyagilag hozzájárulhatnak. Ezekhez az alábbi végpontokat valósítottuk meg:

- GET /products: Az összes elérhető terméket listázza, beleértve azok nevét, leírását, célösszegét, aktuális támogatási szintjét és a kép URL-jét. A Laravel Product::with('category') hívással lekérjük a kapcsolódó kategória nevét is (ha van).
- GET /products/{id}: Egy adott termék részletes adatait szolgáltatja az id alapján.
- POST /products: Új termék feltöltése. A request tartalmazhat képet is, amelyet a backend a public/storage/images mappába ment.
- PUT /products/{id}: Létező termék adatainak frissítése (pl. név, leírás, célösszeg).
- DELETE /products/{id}: A kiválasztott termék törlése az adatbázisból.

Ezek a műveletek az admin felhasználók számára érhetők el a frontend admin felületéről (/ADMIN). Az Axios multipart/form-data formátumban küldi a képeket, amelyet a Laravel oldalon a store() metódus dolgoz fel.

3.9 Vue nézetek és mappaszervezés

A SegítsVelem frontendje Vue 3 keretrendszerre épül, és a teljes alkalmazás egyetlenoldalas alkalmazásként (SPA) működik. A nézetek és komponensek jól elkülönített mappastruktúrában helyezkednek el, amely elősegíti az átlátható és karbantartható fejlesztést.

A fő fájlszerkezet az alábbi:

- src/App.vue – Az alkalmazás gyökérkomponense, amely tartalmazza a router kimenetét és a globális struktúrát.
- src/views/ – Itt találhatók az oldalnézetek (route-komponensek), pl.:
  - HomeView.vue
  - ProductsView.vue
  - ProfileView.vue
  - AdminView.vue
  - RegisterView.vue, LoginView.vue
  - ConnetionsView.vue (kapcsolat)
  - Tartalmi oldalak: AboutView.vue, TermsAndConditionsView.vue, PrivacyAndPolicyView.vue, HowToHelpView.vue
- src/components/ – Újrafelhasználható UI-komponensek, pl.:
  - ProductCard.vue
  - Support.vue
- src/components/layout/ – Navigációs komponens:
  - Navbar.vue

A navigáció Vue Router segítségével történik. Az oldalak között való váltás során nem

történik teljes oldalújratöltés, így gyors és zökkenőmentes felhasználói élményt biztosít. A dizájn Bootstrap 5-re épül, és reszponzív megjelenést nyújt minden eszközön.

3.10 Middleware és biztonság

A Laravel `middleware` funkcióját használjuk a jogosultságok kezelésére. A Laravel keretrendszer lehetőséget biztosít middleware-k (köztes rétegek) alkalmazására, amelyek segítségével szabályozni tudjuk a kérésfeldolgozás menetét, hitelesítést, jogosultságokat és más biztonsági intézkedéseket.

3.10.1 Middleware csoportok

A Laravel app/Http/Kernel.php fájlban definiált middleware-csoportok közül kettőt használ aktívan a projekt:

- **web** csoport: ez tartalmazza az alapvető webes köztes rétegeket, mint a cookie-kezelés, session kezelés, CSRF védelem, stb.
- **api** csoport: ezt alkalmazzuk az összes API végpontra, és tartalmazza a EnsureFrontendRequestsAreStateful middleware-t a Sanctum miatt, a throttle middleware-t (sebességkorlátozás), és a paraméter-helyettesítést.

Az API végpontokat a routes/api.php fájlban definiáljuk, így automatikusan az api middleware csoport vonatkozik rájuk. Ez biztosítja, hogy az összes API-hívásra érvényesüljenek a megadott szabályok.

3.10.2 Laravel Sanctum autentikáció

A Laravel Sanctum biztosítja a stateful autentikációt SPA (Single Page Application) környezetben. Minden bejelentkezett felhasználó egy HTTP-only cookie-n keresztül kap sessiont, amellyel később hitelesített kéréseket küldhet. A következő elemek szükségesek a helyes működéshez:

- A SANCTUM_STATEFUL_DOMAINS=localhost:5173 beállítás az .env fájlban, hogy a Vue frontendről érkező kérések megbízható forrásból származzanak.
- A withCredentials: true beállítás az Axios példányban (src/utils/http.js), hogy a cookie minden kérésnél elküldésre kerüljön.
- A EnsureFrontendRequestsAreStateful middleware aktiválása az api middleware csoportban (Kernel.php).

Minden végpont, amelyet csak bejelentkezett felhasználó érhet el, az auth:sanctum middleware-rel van védve. Ez garantálja, hogy jogosulatlan hozzáférés ne történhessen.

3.10.3 CORS konfiguráció

Az API-t a frontend külön domainről (localhost:5173) éri el, ezért CORS (Cross-Origin Resource Sharing) beállításokra van szükség. Ezeket a config/cors.php fájlban állítottuk be:

```
'paths' => ['api/*', 'sanctum/csrf-cookie'],
'allowed_origins' => ['http://localhost:5173'],
'allowed_methods' => ['*'],
'allowed_headers' => ['*'],
'supports_credentials' => true,
```

Ez biztosítja, hogy a Vue frontend zavartalanul kommunikálhasson a Laravel backenddel, és a cookie-kat is helyesen kezelje.

3.10.4 Adatvalidáció

Minden olyan végpont esetében, ahol adatokat fogadunk (pl. POST /register, POST /login, POST /orders, POST /contact), használunk validációt. A validáció Laravel form request validációval történik, például:

```
$request->validate([
   'email' => 'required|email|unique:users,email',
   'password' => 'required|string|min:6',
]);
```

Ez megakadályozza, hogy hibás vagy rosszindulatú adatok kerüljenek az adatbázisba. A validáció során visszatérő hibaüzeneteket a frontend képes kezelni és megjeleníteni.

3.10.5 Admin jogosultság

A projekt jelenlegi verziójában az admin jogosultságokat a frontend oldalon egyedi e-mail cím alapján (pl. admin@admin.com) különböztetjük meg. A /login után a frontend eldönti, hogy a bejelentkezett felhasználó az admin felületre (/ADMIN) vagy a profil oldalra (/profile) kerüljön. A backend oldalról is biztosítani lehetne (vagy a jövőben lehet) az admin jogosultságok validálását middleware-rel (pl. egy is_admin mező ellenőrzésével).

3.10.6 Session és CSRF védelem

A session-kezelést a Laravel SESSION_DRIVER=database beállítással az adatbázison keresztül végezzük. Ez megbízhatóbb és biztonságosabb hosszú távú megoldás, különösen több szerveren futó rendszerek esetén. A CSRF védelem minden POST kérés előtt a /sanctum/csrf-cookie lekérdezésével aktiválódik, amelyet a frontend minden érzékeny művelet előtt végrehajt.

3.11 Alkalmazott technológiák és eszközök

A projekt során modern, jól dokumentált technológiákat alkalmaztunk, amelyek lehetővé tették egy stabil, biztonságos és jól skálázható rendszer kialakítását:

- **Laravel 11** – PHP-alapú keretrendszer RESTful API támogatással és beépített autentikációval.
- **Vue 3** – JavaScript-alapú frontend keretrendszer, amely komponens-alapú nézetekkel biztosít dinamikus, gyors felhasználói élményt.
- **MySQL** – Relációs adatbázis, amelyet megfelelően normalizált sémával használtunk.
- **Laravel Sanctum** – Token-alapú hitelesítési rendszer API-védelmi rétegként.
- **Bootstrap 5** – A frontend dizájnhoz és a reszponzív elrendezésekhez használt CSS keretrendszer.
- **Git & GitHub** – Verziókövetés és csapatmunka támogatás. Minden fejlesztő külön ágban dolgozott.
- **XAMPP** és **Visual Studio Code** – Lokális fejlesztéshez használt környezetek.

Ezek az eszközök segítették a fejlesztést, csapatmunkát és a moduláris felépítést, így a rendszer könnyen bővíthető és karbantartható maradt.

3.12 Adatbázis séma elemzés

1. **users tábla – Felhasználók**
   A regisztrált felhasználók adatait tartalmazza.

Oszlopnév	Típus	Jelentés
id	bigint(20)	Egyedi azonosító (primary key)
name	varchar(255)	Felhasználó neve
email	varchar(255)	E-mail cím
email_verified_at	timestamp	E-mail visszaigazolás időpontja
password	varchar(255)	Jelszó (hash-elve tárolva)
remember_token	varchar(100)	Token a "maradjak bejelentkezve" funkcióhoz
created_at	timestamp	Létrehozás dátuma
updated_at	timestamp	Utolsó módosítás dátuma
is_admin	tinyint(1)	Adminisztrátori jogosultság (1 = admin, 0 = felhasználó)

2. **products tábla – Támogatási célok vagy kampányok**

Azokat az elemeket tartalmazza, amelyekhez támogatást lehet küldeni.

Oszlopnév	Típus	Jelentés
id	bigint(20)	Egyedi azonosító
name	varchar(255)	Termék/kampány neve
description	text	Részletes leírás
image	varchar(255)	Kép URL vagy fájlút
goal_amount	int(11)	Célösszeg (pl. 10000 Ft)
collected_amount	int(11)	Eddig összegyűlt összeg
created_at	timestamp	Létrehozás időpontja
updated_at	timestamp	Utolsó frissítés

3. orders tábla – Megrendelések / támogatások

A felhasználók által leadott rendeléseket, illetve támogatásokat rögzíti.
Kapcsolatok:

Oszlopnév	Típus	Jelentés
id	bigint(20)	Egyedi azonosító
user_id	bigint(20)	Kapcsolódó felhasználó (külső kulcs: users.id)
product_id	bigint(20)	Kapcsolódó termék (külső kulcs: products.id)
status	varchar(255)	Rendelés állapota (pl. 'pending', 'completed')
created_at	timestamp	Létrehozás dátuma
updated_at	timestamp	Módosítás dátuma

- orders.user_id → users.id
- orders.product_id → products.id

4. sessions tábla – Munkamenetek

A bejelentkezett felhasználók munkameneteinek kezelésére szolgál.

Oszlopnév	Típus	Jelentés
id	varchar(255)	Munkamenet azonosító
user_id	bigint(20)	Felhasználóhoz kötött azonosító
ip_address	varchar(45)	IP-cím
user_agent	text	Böngésző vagy kliens adatai
payload	longtext	Serializált munkamenet-adatok
last_activity	int(11)	Utolsó aktivitás időbélyegként

5. messages tábla – Üzenetek

A kapcsolatfelvételi űrlapon keresztül beküldött üzeneteket tartalmazza.

Oszlopnév	Típus	Jelentés
id	bigint(20)	Egyedi azonosító
name	varchar(255)	Küldő neve
email	varchar(255)	Küldő e-mail címe
message	text	Üzenet szövege
created_at	timestamp	Létrehozás ideje
updated_at	timestamp	Utolsó módosítás ideje

3.13 Admin funkciók workflow

Az adminisztrátor bejelentkezés után automatikusan átirányításra kerül az admin dashboard felületre. Itt a következő funkciók érhetők el:

1. **Termékkezelés**
2. Az admin új terméket vehet fel, megadhatja a nevét, leírását, kategóriáját, valamint képet is feltölthet. A már meglévő termékek szerkeszthetők vagy törölhetők. Az adatok táblázatos formában jelennek meg.
3. **Felhasználók kezelése**
4. Külön nézetben listázhatók a regisztrált felhasználók, ahol azok adatai megtekinthetők vagy módosíthatók.
5. **Statisztikai adatok**
6. A dashboard főoldalán megjelennek az összesített adatok, például a termékek száma.

3.14 Adatvédelem és biztonság

A rendszer fejlesztése során kiemelt figyelmet fordítottunk a felhasználói adatok védelmére és a biztonságos működésre. A Laravel 11 keretrendszer biztonsági funkcióit a projekt teljes mértékben kihasználja az alábbi módokon:

1. **Jelszavak titkosítása:**
2. A felhasználói jelszavak bcrypt algoritmussal kerülnek titkosításra, így visszafejtésük nem lehetséges.
3. **Token alapú hitelesítés:**
4. A bejelentkezett felhasználók egyedi API-token segítségével férnek hozzá az erőforrásokhoz, amelyet a Laravel Sanctum kezel biztonságosan.
5. **Input validáció:**
6. Minden felhasználói adatbevitel validálva van backend oldalon, hogy megakadályozzuk az SQL injection és XSS támadásokat.
7. **Jogosultságkezelés:**
8. A rendszer middleware réteg segítségével ellenőrzi a jogosultságokat. Az adminisztrációs felület kizárólag az admin szerepkörrel rendelkező felhasználók számára érhető el.
9. **Sanctum CSRF védelem SPA-hoz:**
10. Bár Blade sablonokat nem használunk, a Vue 3 frontend és a Laravel backend közötti kommunikáció CSRF tokennel védett, amit a Sanctum automatikusan kezel.

4. Csapattagok szerepei részletesen

4.1 Szőke Simon – Felhasználói felület és dizájn

Szőke Simon fő felelősségi területe az API-alapú felhasználói felületek tervezése és fejlesztése volt. Ő hozta létre a Vue 3 + Vite frontend struktúráját, és megvalósította a legtöbb felhasználói oldalt. Ide tartozik a főoldal, a termékek listázása galériaformátumban, a kapcsolatfelvételi oldal, valamint a profilnézet. Emellett Simon kezelte a reszponzivitást és a felhasználói élményt támogató visszajelzéseket: hibás mezők kiemelését, modális ablakok és UI-elemek integrálását. Dolgozott a

termékmegjelenítés frontend logikáján, valamint ő tervezte és készítette el a `Product`, modelt és kisebb backend kapcsolódásokat is segített tesztelni és javítani. Az adminfelület általános szerkezete, route-kezelése és komponens-alapú rendszere szintén az ő munkájához kapcsolódik.

4.2 Horváth Csaba – Backend és adatmodell

Az adatmodellek a Laravel Eloquent ORM segítségével lettek megvalósítva, amely lehetővé teszi az objektum-orientált hozzáférést az adatbázishoz. A főbb modellek: User, Product, Order és Message. Ezek a modellek szorosan kapcsolódnak egymáshoz, és jól reprezentálják a platform működéséhez szükséges entitásokat.

- A User modell tárolja a regisztrált felhasználók adatait. A mezők között megtalálható a név, az e-mail cím és a hash-elt jelszó. Az autentikációs rendszer Laravel Sanctum alapú, így a User modellhez tartozó tokenek kezelése is teljes mértékben támogatott.
- A Product modell reprezentálja a támogatásra váró célokat. Minden termékhez tartozik egy név, leírás, ár és opcionálisan kép is. Az adminfelület lehetővé teszi ezen termékek létrehozását, módosítását vagy törlését.
- Az Order modell a felhasználók által leadott támogatásokat (rendeléseket) tárolja. Tartalmazza a felhasználó ID-ját, a termék ID-ját, a darabszámot és a rendelés állapotát, amely lehet függőben, elfogadott vagy elutasított.
- A Message modell kezeli azokat az üzeneteket, amelyeket bejelentkezett felhasználók küldenek az adminisztrációnak. Az üzenetek neve, e-mail címe és szöveges tartalma az adatbázisban kerül eltárolásra, valamint e-mail formájában is továbbításra kerül a segitsvelem@gmail.com címre.

A modellekhez kapcsolódó migrációs fájlokat az artisan make:migration parancs segítségével hoztuk létre, és a szerkezetüket úgy alakítottuk ki, hogy megfeleljenek az alkalmazás üzleti logikájának és követelményeinek. Minden mezőhöz tartozik típus, validáció és alapértelmezett érték (ahol szükséges). A migrációk futtatása az artisan migrate parancs segítségével történik, amely automatikusan létrehozza az adatbázis táblákat a fejlesztői és éles környezetben is.
Az adatmodellek között relációk is definiálva vannak. A User modell például hasMany kapcsolatban áll a Message és az Order modellekkel. Ez lehetővé teszi, hogy egyszerűen lekérdezzük egy adott felhasználó összes rendelését vagy üzenetét.

A Product modell hasMany kapcsolatban van az Order modellel, így könnyen megállapítható, hogy egy adott terméket hányszor és kik támogatták.
Az adatmodell kialakításának egyik fő célja az volt, hogy minden logikai egység önállóan és biztonságosan tudjon működni, de egyúttal illeszkedjen is a teljes rendszerbe. A kód strukturáltsága és olvashatósága elősegíti a jövőbeni fejlesztéseket, bővítéseket is. A modellstruktúra későbbi optimalizálásához hozzájárul, hogy figyeltünk az indexek, idegen kulcsok és migrációs verziók megfelelő kezelésére is.

4.3 Lokár Márk – Admin input mezők fejlesztése

Lokár Márk az adminfelülethez tartozó űrlapmezők és komponensek kialakításán dolgozott. Részt vett a termékfeltöltési funkciókhoz szükséges input mezők és frontend logikák elkészítésében. A hozzáadási és szerkesztési nézetekhez kapcsolódó

komponenseket fejlesztette, amelyek lehetővé teszik az adminisztrátor számára új termékek létrehozását.

4.4 Szőke Simon – Designelvek és UI irányelvek részletesen

Szőke Simon a SegítsVelem projekt frontend megjelenéséért és a felhasználói élmény kialakításáért felelt. Vue 3 + Vite alapú keretrendszerben dolgozott, és az oldal minden látványos, interaktív elemét ő valósította meg. Ide tartoznak a gombok, űrlapmezők, vizuális visszajelzések, modális ablakok, ikonok és a rácsos elrendezések.
A design kialakításakor törekedett a letisztultságra, egyszerűségre és könnyű kezelhetőségre. A vizuális irányelvek az alábbi alapelvek mentén készültek:

- Az interaktív elemek, kontrasztosak és jól elkülönülnek a tartalomtól
- Az elsődleges funkciók (pl. bejelentkezés, regisztráció) kiemelt színnel jelennek meg
- Az űrlapok rendezettek, a validációs hibák piros, a sikeres műveletek zöld színnel jelennek meg
- Mobilos nézetben a menü hamburger ikonná alakul, a termékek rácsa listanézetté vált

Simon készítette el az újrafelhasználható Vue-komponenseket is, mint például a termékkártyák, modális ablakok és form inputok. Ezeket úgy tervezte, hogy a projekt több részén is egységesen és újrahasználható módon szerepeljenek. A frontend stílusossága és felépítésbeli rendezettsége nagyrészt az ő munkájának eredménye.

4.5 Horváth Csaba – API architektúra részletesen

A SegítsVelem projekt API architektúráját úgy terveztük meg, hogy egyértelmű, logikus és biztonságos módon támogassa a Vue.js frontend működését. A Laravel backend REST alapú API végpontokat biztosít, amelyek HTTP protokollon keresztül fogadják és szolgálják ki a kéréseket. Minden API hívás JSON formátumú választ ad vissza, ez biztosítja a frontend és a backend közötti hatékony adatkommunikációt.
Alapelvek és struktúra
A Laravel keretrendszer routes/api.php fájljába kerültek a teljes REST API definíciók. Az API szerkezete erőforrás-orientált: az egyes entitásokhoz (pl. felhasználó, termék, rendelés, üzenet) külön útvonalak és vezérlők tartoznak. A logikai réteg különválasztása a routingtól biztosítja az olvashatóságot és a karbantarthatóságot.
A fő útvonalak a következők:

- POST /register – Felhasználó regisztráció
- POST /login – Bejelentkezés
- POST /logout – Kijelentkezés
- GET /user – Bejelentkezett felhasználó adatainak lekérdezése
- GET /products – Termékek listázása
- POST /products – Új termék létrehozása (csak admin)
- PUT /products/{id} – Termék módosítása
- DELETE /products/{id} – Termék törlése
- GET /orders – Saját rendelések lekérdezése
- POST /orders – Új rendelés leadása
- POST /contact – Kapcsolatfelvételi üzenet küldése (email + adatbázis)

Middleware használat
A auth:sanctum middleware biztosítja, hogy a felhasználói azonosítás minden érzékeny végpontnál kötelező legyen. Ez az útvonalcsoport Laravel Sanctum authentikációval működik, ahol a tokenek a felhasználó hitelesítését szolgálják:

```
Route::middleware('auth:sanctum')->group(function () {
    Route::get('/orders', [OrderController::class, 'index']);
    Route::post('/orders', [OrderController::class, 'store']);
    Route::post('/contact', [MessageController::class, 'send']);
});
```

A CSRF token védelmet a /sanctum/csrf-cookie útvonal biztosítja, amelyet a frontend oldalon minden bejelentkezés és regisztráció előtt meg kell hívni. A withCredentials: true beállítás engedélyezi a sütik küldését a cross-origin hívások során, ami szükséges a stateful hitelesítéshez.

AuthController működése
A AuthController osztály a Laravel alapú validációval és tokenkezeléssel dolgozik. Minden bemeneti mező validálva van a kérések feldolgozása előtt. A regisztrációnál új felhasználó kerül mentésre, bcrypt hash-elt jelszóval. A bejelentkezés az Auth::attempt() segítségével történik, ami automatikusan token generálást is elindít, ha sikeres.

```
return response()->json([
    'user' => $user,
    'token' => $user->createToken('token')->plainTextToken,
]);
```

Ez a válasz biztosítja, hogy a frontend lokálisan tárolhassa a tokent, vagy sütin keresztül visszaküldje a későbbi kérések során.

Admin API endpointok
Az adminisztrátorok speciális útvonalakon keresztül érik el a rendelések elfogadását vagy elutasítását:

```
Route::get('/admin/orders', [AdminOrderController::class, 'index']);
Route::put('/admin/orders/{id}/approve', [AdminOrderController::class, 'approve']);
Route::put('/admin/orders/{id}/reject', [AdminOrderController::class, 'reject']);
```

Az adminisztrációs funkciók is token-alapú hitelesítéssel védettek, és kizárólag egy előre megadott emailcímmel rendelkező felhasználó (pl. admin@email.com) férhet hozzá ezekhez a funkciókhoz. Ez az ellenőrzés middleware vagy controller szinten történik.

Hibaüzenetek és hibakezelés
Minden hívás validálva van backend oldalon, és hiba esetén részletes visszajelzést adunk a frontend számára. Az API-k 422 vagy 401 hibakódokat adnak vissza, ha validáció vagy jogosultsági hiba történik. A fejlesztés során a Laravel try-catch blokkjait is használjuk, hogy a JSON válaszok mindig következetesek legyenek.

4.6 Szőke Simon – Felhasználói élmény és vizuális egység megvalósítása

Szőke Simon a SegítsVelem rendszer teljes felhasználói felületének (UI) és felhasználói élményének (UX) kialakításáért felelt. Vue 3 és Vite alapú környezetben dolgozott, ahol célja egy átlátható, reszponzív, mobilbarát és vizuálisan konzisztens rendszer

létrehozása volt.
Simon készítette el az összes felhasználói nézetet, többek között a kezdőoldalt, a termékgalériát, a kapcsolat oldalt és a profilnézetet. Emellett a komponens-alapú rendszer kialakítását is ő tervezte, így például az egyedi termékkártyákat, navigációs elemeket és modális ablakokat.
A vizuális tervezés során az alábbi elveket alkalmazta:

- A navigáció mobilnézetben hamburger menüre vált
- Kontrasztos színek és letisztult tipográfia a célcsoport igényeihez igazítva
- A rácsos termékmegjelenítés alkalmazkodik a képernyőmérethez
- Hibakezelő visszajelzések piros keretben jelennek meg az űrlapoknál
- Interaktív hover-effektusok minden kattintható elemnél
- Egységes gombok, input mezők és modális ablakok a teljes rendszeren belül
- Kiemelt figyelem az akadálymentességre (pl. jól olvasható betűméret, színkontraszt)

Simon kialakította a rendszer vizuális felépítését, figyelembe véve a reszponzivitást, az egységes stíluselemeket és az újrahasználható komponensek kialakítását. A frontend kód felépítése átlátható, a Vue-komponensek logikus struktúrában lettek megszervezve, ami megkönnyíti a karbantartást és a jövőbeli bővítést.

4.7 Horváth Csaba – Backend optimalizálás, teljesítmény és tesztelhetőség

A SegítsVelem rendszer stabil működésének egyik kulcsfontosságú eleme a backend teljesítményének optimalizálása és annak biztosítása, hogy a rendszer könnyen tesztelhető legyen. Ebben a fejezetben bemutatásra kerülnek azok a megoldások és fejlesztői döntések, amelyeket a Laravel backend optimalizálása és tesztelhetőségének javítása érdekében hoztam meg.

1. Kódstrukturálás és optimalizált controller használat
A Laravel keretrendszer lehetőséget biztosít a vezérlők logikus szétválasztására. Ahelyett, hogy minden logikát egyetlen controllerbe írtunk volna, külön controller szolgál:

- a termékek (ProductController),
- a rendelések (OrderController, AdminOrderController),
- a hitelesítés (AuthController),
- valamint a kapcsolatfelvételi funkciók (MessageController) kezelésére.

Ez a komponens-alapú struktúra lehetővé teszi a gyorsabb betöltést és kisebb memóriahasználatot, mivel csak a szükséges függőségek kerülnek betöltésre.
A vezérlőkön belül az egyes műveleteket önálló, könnyen tesztelhető metódusokra bontottam. Például az OrderController::store() csak az adatellenőrzéssel, mentéssel és válasz visszaadásával foglalkozik, így könnyen mock-olható, ha automatizált tesztre van szükség.

2. Adatbázis-lekérdezések hatékonysága
A Laravel Eloquent ORM használata mellett törekedtem arra, hogy mindenhol "eager loading"-ot alkalmazzak, ahol az adatkapcsolatok ezt lehetővé tették. Ennek célja az

N+1 lekérdezési probléma elkerülése. Például, amikor az admin rendelések listáját lekéri:
Order::with('user', 'product')->get();

Ez a lekérdezés előre betölti a kapcsolódó user és product adatokat, így csökkenti az adatbázis-hívások számát. Ez a technika akár több száz millisekundumos gyorsulást is eredményezhet nagyobb adatmennyiség mellett.
Ahol csak lehet, a select() metódust használtam, hogy csak a szükséges mezőket kérje le a rendszer:
User::select('id', 'name', 'email')->get();
Így kisebb adatcsomagokat küldünk át a backend és az adatbázis között, ezzel optimalizálva a sávszélességet és válaszidőt.

3. Middleware szűrés és jogosultságellenőrzés
Az API útvonalak közül több csak bejelentkezett felhasználók számára érhető el, amit a auth:sanctum middleware biztosít. Ezt úgy szerveztük, hogy egyes endpointokat csoportosítottunk:

```
Route::middleware('auth:sanctum')->group(function () {
   Route::post('/orders', [OrderController::class, 'store']);
   Route::post('/contact', [MessageController::class, 'send']);
});
```

Ennek előnye, hogy nem szükséges minden útvonalhoz külön megadni a middleware-t, ami átláthatóbbá teszi a route-fájlokat, és csökkenti a hibalehetőségeket.
Továbbá az admin jogosultságokat is backend oldalon, egyéni feltétellel ellenőrizzük, így nincs szükség külön szerepkör-kezelő rendszerre:

```
if ($request->user()->email !== 'admin@email.com') {
   return response()->json(['message' => 'Nincs jogosultság'], 403);
}
```

Ez egyszerű, de hatékony módja annak, hogy az admin funkciókat csak a kijelölt személy érje el. **4. Cache és session kezelés**
A SESSION_DRIVER=database beállítással a session-adatok nem fájlban, hanem az adatbázisban kerülnek tárolásra, ami lehetővé teszi a terheléselosztást (load balancing) és több szerveres környezetben való működést is.
Az adatintenzívebb lekérdezéseknél a Laravel cache rendszere is használható, bár jelenlegi projektméret mellett ez nem volt prioritás. A CACHE_DRIVER=database beállítás azonban már előkészíti a talajt a jövőbeni gyorsítótárazási lehetőségek számára.

5. Tesztelhetőség – PHPUnit és strukturált tesztek
A Laravel tests/Feature könyvtárában létrehoztam néhány kulcsfontosságú tesztesetet, amelyeket PHPUnit segítségével futtattunk. A főbb tesztek:

- Felhasználói regisztráció és bejelentkezés (/register, /login)
- Termék létrehozása admin általi POST kérésre
- Rendelés leadása valós felhasználói tokennel
- Kapcsolatfelvételi üzenet hitelesített POST kérésre

A tesztek a Laravel beépített RefreshDatabase traittel kerültek lefuttatásra, így minden teszt futtatása izolált környezetben történik:
use Illuminate\Foundation\Testing\RefreshDatabase;

Ez biztosítja, hogy az adatbázis mindig újra legyen töltve, a tesztek nem befolyásolják egymást.

4.8 Szőke Simon – Reszponzív tervezés és felhasználói visszajelzések kezelése

A frontend kialakítása során Szőke Simon gondoskodott arról, hogy az oldal minden képernyőméreten – mobilon, táblagépen és asztali gépen – könnyen kezelhető legyen. A reszponzivitás kialakítása Bootstrap 5 osztályok segítségével történt, amelyek biztosítják az elemek elrendezésének alkalmazkodását a különböző eszközökhöz.
A megvalósított reszponzív viselkedések közé tartoznak:

- A felső navigációs menü mobilon hamburger ikonként jelenik meg
- A terméklista kártyák grid elrendezésben, dinamikus oszlopszámmal láthatók
- Az űrlapok mezői mobilnézetben egymás alá kerülnek
- A táblázatok kisebb képernyőn görgethetővé válnak

A felhasználói visszajelzések megjelenítése Vue-alapú komponensekkel történik. Hibás űrlapkitöltés esetén a mezők alatt hibaüzenetek jelennek meg, sikeres művelet esetén szöveges visszajelzések tájékoztatják a felhasználót.
Simon olyan UI-megoldásokat alkalmazott, amelyek segítik a látogatót az aktuális művelet értelmezésében. A színkódolás és elrendezések vizuálisan is visszajelzést adnak a rendszer állapotáról.

4.9 Horváth Csaba – Adatstruktúrák optimalizálása és relációk finomítása

A SegítsVelem platform adatbázisa a Laravel Eloquent ORM rendszerét használja, amely lehetővé teszi a relációk természetes, olvasható és hatékony kezelését. Ebben a fejezetben részletesen bemutatom, hogyan építettem fel az adatstruktúrát úgy, hogy az megfeleljen az alkalmazás működési logikájának, és egyben támogassa a jövőbeli bővíthetőséget és teljesítmény-optimalizálást is.

1. Adatmodell kialakítás
Az adatbázis struktúráját úgy alakítottam ki, hogy minden egyes fő entitás külön táblában szerepeljen, és az entitások közötti kapcsolatok egyértelműen, normalizált módon legyenek kezelhetők. A fő táblák:

- **users** – A rendszerbe regisztrált felhasználók adatait tartalmazza.
- **products** – A támogatási célok, ruhák, csomagok vagy egyéb segélykérések tárolására szolgál.
- **orders** – A felhasználók által leadott támogatások (rendelések) rögzítése.
- **messages** – A kapcsolatfelvételi űrlapon keresztül küldött üzeneteket tárolja.

A normalizálás során minden felesleges adatismétlést kizártam, például az orders táblában csak a user_id és product_id szerepel, nem duplikálódik semmilyen más adat.

2. Relációk megvalósítása Eloquent modellekben
A Laravel Eloquent ORM lehetővé teszi, hogy a relációk egyszerű metódusokként jelenjenek meg a modellekben. A legfontosabb kapcsolatok:
User model:

```
public function orders()
{
   return $this->hasMany(Order::class);
}
```
Product model:
```
public function orders()
{
   return $this->hasMany(Order::class);
}
```

Order model:
```
public function user()
{
   return $this->belongsTo(User::class);
}

public function product()
{
   return $this->belongsTo(Product::class);
}
```
Ezek a relációk lehetővé teszik, hogy egyetlen sorral lekérjük a kapcsolódó entitásokat is, például egy felhasználó összes rendelését vagy egy rendeléshez tartozó terméket.

3. Eager Loading és teljesítmény
A Laravel lehetővé teszi az úgynevezett “eager loading”-ot, vagyis azt, hogy a relációk adatait már az első lekérdezéskor betöltsük. Ez különösen fontos, amikor egy admin felületre vagy a frontend API-n keresztül listát kérünk le.
Példa a lekérdezésre:
```
Order::with('user', 'product')->get();
```
Ez a megközelítés jelentősen csökkenti az adatbázis-hívások számát, különösen nagy adatbázis esetén, és növeli a válaszidő sebességét.

4. Adattípusok optimalizálása
Az adatbázis migrációk során külön figyelmet fordítottam arra, hogy a megfelelő adattípusokat válasszam:

- Az id mezők minden táblában bigIncrements típusúak a skálázhatóság érdekében.
- Az email, name, title mezők string, korlátozott hosszúsággal.
- A description, message típusok text, mivel hosszabb tartalmak is érkezhetnek.
- Az időbélyegek (created_at, updated_at) automatikusan kezelve vannak.

Ez nemcsak a tárolás szempontjából optimális, hanem a validáció és hibakezelés szempontjából is átlátható.

5. Migrációs fájlok struktúrája
Minden tábla létrehozása külön migrációs fájlban történt. Példa:
```
php artisan make:migration create_orders_table
```
A create_orders_table.php fájlban a struktúra így néz ki:
```
Schema::create('orders', function (Blueprint $table) {
```

```
    $table->id();
    $table->foreignId('user_id')->constrained()->onDelete('cascade');
    $table->foreignId('product_id')->constrained()->onDelete('cascade');
    $table->string('status')->default('pending');
    $table->timestamps();
});
```

A constrained() és onDelete('cascade') gondoskodik arról, hogy a kapcsolatok érvényesek legyenek, és ne maradjanak árva rekordok.

6. Jövőbeli bővíthetőség

Az adatmodell kialakításánál figyelembe vettem a bővíthetőséget is. Néhány jövőbeni lehetőség:

- **Felhasználói szerepkörök:** egy role mező hozzáadása a users táblához (pl. user, admin).
- **Termékkategóriák:** külön categories tábla és reláció bevezetése a products táblához.
- **Kapcsolatfelvétel státuszok:** a messages táblában státusz mezővel megjelölni, hogy válaszolt-e rá az admin.

Az ilyen struktúrák lehetővé teszik, hogy a rendszer hosszútávon is karbantartható és jól skálázható maradjon.

4.10 Szőke Simon – Felhasználói pszichológia és interaktív élmény tervezése

Szőke Simon a SegítsVelem projekt felhasználói felületének tervezése során törekedett arra, hogy az oldal egyszerűen használható, átlátható és logikusan felépített legyen. A Vue 3 komponensrendszerrel készült frontend célja az volt, hogy a látogatók gyorsan megtalálják a keresett funkciókat, és a felület minél kevesebb tanulási időt igényeljen.

A projekt során Simon alakította ki az olyan kulcsfontosságú komponenseket, mint a termékkártyák (ProductCard.vue), a regisztrációs űrlap (RegisterView.vue), valamint a navigációs sáv (Navbar.vue). Ezek a komponensek jól strukturáltak, és figyelembe veszik a különféle kijelzőméreteket is. A navigációs rendszer mobilnézetben átalakul hamburger menüvé, a termékek rácsos elrendezésben jelennek meg, és az űrlapok elemei kisebb képernyőn egymás alá rendeződnek.

A felhasználói élményt támogató vizuális visszajelzések is megjelennek:

- Űrlaphibák esetén hibaüzenetek jelennek meg a beviteli mezők alatt
- A gombok elhelyezése és stílusa egyértelművé teszi a funkciójukat
- Hover-effektusok segítik a navigációt és az interakciók felismerését

Bár a projekt nem tartalmazott külön animációkat, toast-üzeneteket vagy Vue transition effekteket, az alapvető visszajelzések vizuálisan is érzékelhetők. A színkódolás egyelőre nem követ egy egységes rendszert, de az alapértelmezett Bootstrap stílusok néhol megjelennek, például a formok szerkezetében. A dokumentáció vagy a projektkód nem tartalmaz külön definiált színsémát vagy WCAG szerinti hozzáférhetőségi specifikációt, de a tipográfia és a kontraszt általános szinten jól használható.

Simon a komponenseket úgy alakította ki, hogy azok logikusan illeszkedjenek a felhasználói útvonalakhoz. A főbb funkciók (regisztráció, bejelentkezés, termékböngészés, kapcsolatfelvétel) egyértelműen elérhetők a menüből, és minden

oldalról maximum néhány kattintással elérhetők.
A felület kialakításánál cél volt, hogy a rendszer minimális vizuális túlterhelést nyújtson, és elsősorban a funkciók egyszerű elérését támogassa. A Vue 3 komponensstruktúra jól karbantarthatóvá és bővíthetővé teszi a rendszert, így a későbbi UX-fejlesztések és bővítések könnyen kivitelezhetők.

4.11 Horváth Csaba – Működésbiztonság és hibakezelés

A SegítsVelem projektben a működésbiztonság és a hibakezelés kiemelt figyelmet kapott, mivel egy közösségi célokat szolgáló platformnál elengedhetetlen a megbízhatóság, a hibamentes működés és a felhasználói biztonság. Ebben a részben bemutatom azokat a megoldásokat, amiket alkalmaztam backend oldalon a biztonságos adatkezelés, megfelelő hibakezelés és védelmi mechanizmusok biztosítása érdekében.

1. Autentikáció és hozzáférés-kezelés
A Laravel Sanctum csomaggal valósítottam meg a felhasználói bejelentkezést és a hozzáférés-ellenőrzést. A Sanctum előnye, hogy cookie alapú tokenkezelést biztosít, ami tökéletesen illeszkedik SPA (Single Page Application) architektúrához.

- A auth:sanctum middleware-rel védtem azokat az útvonalakat, amelyekhez csak bejelentkezett felhasználók férhetnek hozzá (például /user, /orders, /contact).
- A EnsureFrontendRequestsAreStateful middleware gondoskodik arról, hogy az állapotmegőrző hitelesítés csak megbízható domainekről érkező kérésekre történjen meg (beállítva: localhost:5173).

2. Adatvalidáció és hibakezelés
A Laravel beépített validációs rendszerét használva minden POST és PUT kérésnél biztosítottam, hogy az adatok megfeleljenek az elvárt formátumnak. Példák:

```
$request->validate([
  'email' => 'required|email',
  'password' => 'required|string|min:6'
]);
```

A validációs hibák automatikusan JSON válaszként visszatérnek a frontend felé, így azok azonnal kezelhetők Vue.js oldalon. A hibák 422 Unprocessable Entity státuszkóddal térnek vissza.
Emellett a controllerekben try-catch blokkokkal is körbevettem az érzékenyebb műveleteket (például e-mail küldés), hogy a rendszer ne omoljon össze hiba esetén:

```
try {
    Mail::to('segitsvelem@gmail.com')->send(new ContactMessage(...));
} catch (\Exception $e) {
    return response()->json(['error' => 'Nem sikerült elküldeni az üzenetet.'], 500);
}
```

3. Központi hibakezelés és naplózás
A Laravel Handler osztálya felelős a kivételek kezeléséért. Bár nem tértem el a keretrendszer alapértelmezett működésétől, a .env fájlban bekapcsoltam a hibák naplózását és megadtam a stack log csatornát:

```
LOG_CHANNEL=stack
LOG_LEVEL=debug
```

Ez biztosítja, hogy minden rendszerhiba naplózásra kerüljön a storage/logs/laravel.log fájlba. A naplózás segítségével könnyen visszakereshetők a váratlan hibák okai,

különösen a termelési környezetben.

4. CORS beállítások és HTTP védelem
A Laravel cors.php konfigurációs fájlban pontosan meghatároztam, hogy csak a frontendről (http://localhost:5173) érkező kérések legyenek elfogadva:
'allowed_origins' => ['http://localhost:5173'],
'supports_credentials' => true,
Ezzel együtt a web.php-ben nem engedtem nyitott API-kat olyan útvonalakon, amelyekre nincs szükség publikus hozzáférésre.

5. SQL injekció elleni védelem
A Laravel Eloquent ORM automatikusan használja az adatbázis-illesztő rétegben az ún. "query binding" technikát, ami teljes védelmet nyújt az SQL injection típusú támadások ellen. Például:
$user = User::where('email', $email)->first();
Ahol az $email értéket automatikusan escape-eli, így a támadó nem tud SQL parancsot injektálni a lekérdezésbe.

6. Session és cookie biztonság
A .env fájlban engedélyezve van a session cookie titkosítása, valamint az élettartam is be van állítva:
SESSION_DRIVER=database
SESSION_ENCRYPT=false
SESSION_LIFETIME=120
A Sanctum stateful működéséhez a cookie-alapú hitelesítés használata biztonságosabb, mint a bearer token.

7. Email küldés és SMTP védelem
Az emailküldés biztonságos csatornán keresztül történik a smtp.gmail.com szerveren TLS titkosítással:
MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_ENCRYPTION=tls
A kapcsolatfelvétel kizárólag bejelentkezett felhasználók számára elérhető, ezáltal megelőzhető a spam és visszaélés.

8. Felhasználói jogosultságok szétválasztása
Bár a projekt nem épít ki külön szerepkör-alapú jogosultságkezelést, az admin fiók email címe (admin@admin.com) alapján lehetőség van az admin felület elérésére. A frontend LoginView ezt vizsgálja, és sikeres bejelentkezés után admin esetén azonnal az /ADMIN felületre irányít.

4.12 Szőke Simon – Komponensalapú frontend struktúra Vue 3 keretrendszerben

Szőke Simon felelt a SegítsVelem projekt frontend struktúrájának kialakításáért. A felület Vue 3 és Vite alapokra épült, a projekt nem használ Blade sablonokat, hanem teljes egészében egy különálló SPA (Single Page Application) frontendként működik, amely egy REST API-t használ kommunikációs háttérként. A komponensalapú felépítés lehetővé teszi az egyszerű bővítést, újrahasznosítást és karbantartást, különösen akkor, ha több fejlesztő dolgozik a rendszeren.
A projekt mappastruktúrája világosan elkülöníti a különböző funkciókat:
Nézetek és komponensek felépítése

- A src/views/ könyvtár tartalmazza az egyes főoldalakat (route-komponensek), mint például:
  - HomeView.vue: a nyitóoldal, ahol megjelenik a projekt célja és fő funkciókra mutató linkek
  - ProductsView.vue: termékböngészés, ahol a felhasználók kiválaszthatják az igényelni kívánt sportruházatot
  - ProfileView.vue: a felhasználó saját igényléseit és adatait megjelenítő nézet
  - RegisterView.vue és LoginView.vue: azonosításhoz szükséges űrlapoldalak
  - ConnectionsView.vue: kapcsolatfelvételi oldal
  - AboutView.vue, HowToHelpView.vue, TermsAndConditionsView.vue: tartalmi oldalak
- A src/components/ könyvtárba kerültek az újrahasználható komponensek:
  - ProductCard.vue: egy-egy terméket megjelenítő kártya, amely tartalmazza a termék nevét, méretét, státuszát és az igénylés gombot
  - Support.vue: információs szakasz a kezdőlapon
  - layout/Navbar.vue: a teljes oldalra kiterjedő navigációs menü, amely mobilon hamburgerré alakul
  - Ezen komponensek minden nézetben újrahasznosíthatók, props segítségével dinamikusan tölthetők adatokkal

Az összes nézetet az App.vue fogja össze, amely tartalmazza a router outletet (<router-view />) és a globális elrendezést.
Reszponzív megvalósítás
A felület reszponzív kialakításához Simon Bootstrap 5 osztályokat használt. Ennek köszönhetően az oldal tartalma képes dinamikusan alkalmazkodni a képernyőméretekhez. A következő megoldások kerültek alkalmazásra:

- A menüsor mobilnézetben összeomlik hamburger ikonra, amely lenyitható
- A termékkártyák grid-elrendezésben jelennek meg: kisebb kijelzőn egyetlen oszlopban, nagyobb képernyőn több oszlopban láthatók
- Az űrlapok mezői mobilnézetben egymás alá kerülnek, így nem szükséges vízszintes görgetés
- A táblázatok és hosszabb sorozatok vízszintesen görgethetők kisebb kijelzőn

Interaktív visszajelzések és strukturált validáció
A felhasználói műveletekre (például regisztráció, bejelentkezés, termékigénylés) alapértelmezett visszajelző mechanizmusok kerültek bevezetésre:

- Az űrlapokon megjelenő hibaüzenetek a beviteli mezők alatt láthatók, piros

színnel kiemelve
- A sikeres műveleteket (pl. igénylés elküldése) szöveges visszajelzések jelzik, jelenleg nem toast-üzenet formájában
- A hover-effektusok vizuálisan kiemelik az interaktív elemeket, segítve a navigációt
- A komponensek kizárólag Vue metódusokkal és v-model kötésekkel kezelik az interakciókat

A rendszer nem tartalmaz külön Vue plugin-alapú visszajelző rendszert, mint a Toastify vagy Vue Toasted, de az alapfunkciók jól működnek vizuálisan is. A validációs hibák a szerveroldali API válaszból jönnek, és ezek megjelenítése a nézeteken belül történik.

Fejlesztési szempontok

Simon a komponenseket úgy alakította ki, hogy azok egyszerűen bővíthetők és testreszabhatók legyenek. A ProductCard.vue például props-on keresztül kapja a termékadatokat, így könnyen újrahasználható bármelyik nézetben. Az input űrlapok v-model segítségével dolgoznak, így az állapotkezelés lokálisan történik.

A router alapú navigáció lehetővé teszi, hogy a felhasználó oldalfrissítés nélkül tudjon mozogni a felületen. A főmenü minden nézet tetején megjelenik, az útvonalváltás zökkenőmentes.

A komponensalapú felépítés lehetővé teszi, hogy a SegítsVelem frontend moduláris, átlátható és hosszú távon is karbantartható maradjon. Mivel a rendszer nem Blade-alapú, hanem különálló SPA-ként működik, a frontend teljes mértékben leválasztható a backendtől. Simon e struktúra kialakításával megalapozta a projekt további bővíthetőségét, legyen szó új oldalakról, új funkciókról vagy reszponzív átalakításról.

4.13 Szőke Simon – Teljes oldalkészítési folyamat és komponenslogika

A SegítsVelem frontend oldalainak többségét Szőke Simon hozta létre önállóan a nulláról, Vue 3 keretrendszer és Bootstrap 5 használatával. Az oldalfejlesztés során nemcsak a vizuális elrendezést tervezte meg, hanem a komponensstruktúrát és az adatok logikai kezelését is. A Vue komponensalapú működése lehetővé tette, hogy minden oldal önálló nézetként (view) és logikailag szétválasztott komponensekkel épüljön fel.

Simon munkafolyamata a következő lépésekből állt:

1. Cél meghatározása

Minden oldal fejlesztése előtt meghatározta az adott nézet funkcionális célját: például a főoldal célja a projekt bemutatása és a navigáció biztosítása, míg a ProductsView.vue célja a ruhák böngészése és igénylése.

2. Szerkezeti felépítés Vue komponensben

A src/views mappában létrehozott új nézetfájlokat, például:
- HomeView.vue
- RegisterView.vue
- ProductsView.vue
- Ezekben az oldalakban meghatározta a főbb HTML-szerkezetet: fejlécek, űrlapmezők, grid-rácsok, leírások, és elhelyezte a kapcsolódó komponenseket.

3. Komponensek beillesztése

Simon minden oldalon újrahasználható Vue komponenseket helyezett el:

- A ProductCard.vue a terméklisták megjelenítésére szolgál.
- A Navbar.vue a menürendszert biztosítja.
- A Support.vue vagy hasonló komponensek információs blokkok megjelenítésére lettek használva.

Az oldalakat úgy építette fel, hogy minden funkció külön komponensben jelenjen meg, így a kód logikailag elválasztott, jól tesztelhető és újrahasznosítható marad.

4. Validáció és adatkezelés

Az űrlapoknál a mezők v-model segítségével adatot kötnek a komponens belső állapotához, amit az axios segítségével küld a Laravel backend felé. Hiba esetén a backend visszaadja a validációs hibákat, ezeket Simon megjelenítette a megfelelő mező alatt egy v-if feltétellel, piros szövegként.

5. Reszponzivitás ellenőrzése

Bootstrap 5 osztályokat alkalmazott (row, col-md-6, d-flex, stb.), így minden oldal jól működik mobilon, táblagépen és asztali nézetben is. Az elrendezést Chrome DevTools segítségével rendszeresen tesztelte különböző kijelzőméretek mellett.

6. Tesztelés és iteráció

Miután elkészült egy oldal, Simon manuálisan végigpróbálta az összes mezőt, gombot és folyamatot. Ha hibát tapasztalt, javította az adatkezelés vagy megjelenítés logikáját.

Komponenslogika és újrahasznosíthatóság

Simon a projekt során figyelt arra is, hogy minden komponens önállóan is működjön. A ProductCard.vue például props-on keresztül kapja meg a termékadatokat, és eseményt bocsát ki, amikor a felhasználó egy ruhát igényel. A Navbar.vue dinamikusan mutatja, hogy be van-e jelentkezve a felhasználó, és a gombok megváltoznak ennek megfelelően.

Ez a komponensalapú megközelítés nemcsak átláthatóbbá tette a fejlesztést, hanem hozzájárult ahhoz is, hogy a projekt jövőbeli bővítése egyszerűen elvégezhető legyen, például új oldal vagy funkció hozzáadásával.

4.14 Lokár Márk – Adminfelület

Az adminfelület kialakítása eredetileg Lokár Márk feladata lett volna, azonban a projekt előrehaladása során a teljes funkcionalitás és vizuális kialakítás végül Szőke Simon és Horváth Csaba munkájával valósult meg. Márk hozzájárulása a komponenshez az alapvető **inputmezők létrehozásában** merült ki.

A src/views/AdminView.vue fájlban található adminfelület egy termékfeltöltő űrlapot tartalmaz, ahol Márk létrehozta az alábbi mezőket Vue 3 keretrendszer segítségével:

- **Terméknév** – szöveges bemenet (v-model: productName)
- **Képfeltöltés** – fájlmező (v-model: productImage, input: type="file")
- **Célösszeg** – numerikus mező (productPrice)
- **Termékleírás** – többsoros szövegmező (productDescription)

Bár az adminpanel teljes megvalósítása nem az ő feladata lett végül, Márk munkája szolgált kiindulópontként az adminfelület struktúrájához.

4.15 Szőke Simon – Frontend és adminpanel dizájn

Szőke Simon felelt az adminfelület **teljes frontend kinézetéért**, a **komponens vizuális kialakításáért**, valamint az interaktív funkciók megvalósításáért. Simon Vue 3 és Bootstrap 5 technológiákat alkalmazva modern, jól strukturált felületet hozott létre.
A mezők mögé v-model adatkötések kerültek, és az alap data() struktúrával került kialakításra:

```
data() {
  return {
    productName: "",
    productImage: null,
    productPrice: "",
    productDescription: ""
  };
}
```

A AdminView.vue komponens Simon által készített részei közé tartoznak:

- A **reszponzív űrlap** elrendezése és stílusozása (admin-form, form-control, stb.)
- A **sikeres / hibás műveletekről szóló visszajelzések** (alert-success, alert-danger)
- A termékek **kártyás megjelenítése** egy Bootstrap rácsos elrendezésben (row, col-md-6, col-lg-4)
- A **szerkesztési mód** megvalósítása: az űrlap képes átváltani „Módosítás" állapotba
- A **Bootstrap modal** törlés megerősítéséhez, teljes mértékben működőképesen

Simon a mounted() ciklusban kezelte a modal példányosítását, valamint gondoskodott arról, hogy a ref hivatkozások megfelelően működjenek Vue 3-ban.
Az adminfelület világos, letisztult színsémája is Simon tervezése: világoskék háttér, fehér űrlapdoboz, halvány árnyék és lekerekített sarkok jellemzik. A termékképek object-fit: cover stílusban, egységes magasságban jelennek meg.

4.16 Horváth Csaba – Backend API és adatkezelés

A projekt során az egyik legfontosabb feladat az volt, hogy a backend API konzisztens, biztonságos és jól dokumentált módon kezelje az adatokat, valamint hatékonyan szolgálja ki a felhasználói interakciókat. Az alábbi szakaszok részletezik, hogyan valósítottam meg az adatok kezelését a Laravel oldalon.

1. REST architektúra és útvonalak
Az egész API-t a routes/api.php fájlban definiáltam, így minden útvonal alapértelmezetten a /api prefix alatt érhető el (a frontend axios baseURL-je ehhez igazodik: http://localhost:8000/api).
Az alábbi főbb erőforrásokat kezeljük:

- **/register, /login, /logout** – autentikációs műveletek
- **/user** – aktuálisan bejelentkezett felhasználó adatainak lekérése
- **/products** – termékek lekérdezése, létrehozása, frissítése, törlése
- **/orders** – leadott igénylések kezelése
- **/contact** – kapcsolatfelvételi űrlap adatainak feldolgozása

Minden erőforráshoz tartozik HTTP metódus alapú leképezés (GET, POST, PUT, DELETE), amely az adatkezelés típusától függően történik.

2. Adatkezelés Eloquent ORM-mel
A Laravel saját ORM rendszerét, az Eloquent-et használtam az adatbázis-műveletekhez. Ez lehetővé teszi, hogy az adatokkal objektumként dolgozzak, és ne kelljen nyers SQL lekérdezésekkel foglalkozni.
Például termék létrehozás:

```
$product = Product::create([
   'name' => $request->name,
   'description' => $request->description,
   'price' => $request->price,
]);
```

Lekérdezés: $products = Product::all();
Módosítás: $product = Product::findOrFail($id);
$product->update($request->all());
Törlés: $product = Product::findOrFail($id);
$product->delete();
$product->delete();

Az Eloquent mass assignment védelme miatt a modellekben meg kellett határozni a fillable mezőket.

3. Felhasználói adatok kezelése
Regisztrációkor a backend új User modellt hoz létre, ahol a jelszót automatikusan bcrypt hash-el titkosítjuk. A bejelentkezés után a felhasználói adatok egy token segítségével elérhetők, és az aktuális bejelentkezett felhasználó lekérhető az /user endpointon keresztül.
$user = Auth::user();
Ez lehetővé teszi, hogy a frontend megjelenítse az éppen aktív felhasználó nevét, email címét, illetve biztosítsa a személyes adatokhoz kötött funkciókat (pl. rendelés, kapcsolatfelvétel).

4. Adminisztrációs funkciók
Az admin funkciók külön útvonalakon érhetők el (pl. /admin/orders, /admin/messages), és bár jelenleg csak e-mail alapján (admin@admin.com) történik az admin azonosítás, a rendszer később könnyen bővíthető role-based access control (RBAC) támogatásával.

5. Üzenetküldés e-mailben
A kapcsolatfelvételi űrlap elküldésekor a backend nem csak adatbázisba ment, hanem emailt is küld a megadott címre. Ehhez Laravel Mailable osztályát használtam:
Mail::to('segitsvelem@gmail.com')->send(new ContactMessage($name, $email, $message));
Ez csak akkor történik meg, ha a felhasználó be van jelentkezve. Az email tartalmát Blade sablon (emails.contact) generálja.

6. Biztonságos adatfeldolgozás
Minden API művelet előtt végzünk szerveroldali validációt:
```
$request->validate([
  'email' => 'required|email',
  'message' => 'required|string'
]);
```
Ezzel megelőzzük, hogy hibás, hiányos vagy rosszindulatú adatok kerüljenek a rendszerbe. Emellett az auth:sanctum middleware minden védett útvonalat ellenőriz.

7. Token kezelés és authentikáció
A regisztráció vagy bejelentkezés során visszatérített plainTextToken nem kerül frontend tárolásra külön (mivel cookie-based authentikációt használunk a Sanctummal), hanem minden kérés automatikusan hitelesített marad a session cookie alapján. Ez biztosítja, hogy a felhasználó végig hitelesítve marad, amíg be nem jelentkezik vagy ki nem lép.

4.17 Szőke Simon – Stíluskezelés és vizuális visszajelzések beépítése Vue 3 frontendben

A SegítsVelem frontend felületének stíluskezeléséért és felhasználói visszajelzéseinek megvalósításáért Szőke Simon volt felelős. A projekt Vue 3 és Bootstrap 5 alapokra épül, így a dizájn és layout kialakítása teljes egészében ezen technológiákra támaszkodik.
Simon minden oldalnál figyelembe vette a célközönséget – főként szülőket, diákokat és adminisztrátorokat –, és ennek megfelelően alakította ki a letisztult, barátságos és strukturált felhasználói felületet. Az oldalak egységes megjelenését Bootstrap osztályokkal, valamint szükség esetén inline vagy scoped CSS-sel biztosította.
Vizuális stíluskezelés

- Az elrendezések Bootstrap rácsszerkezettel (row, col-md-*, container) készültek
- A színek és tipográfia a Bootstrap alapértelmezett témáit követik
- A form elemek (input, label, button) következetesen formázottak, jól olvashatóak és kontrasztosak
- A navigációs sáv minden oldal tetején megjelenik, és mobilon hamburger menüre vált

Simon külön figyelmet fordított arra, hogy az interaktív felületek visszajelzést adjanak a felhasználónak egy-egy művelet után. Bár a projekt nem használ külön Vue-értesítési rendszert (pl. toast vagy modal feedback), az alapvető vizuális visszajelzések minden fontos helyen megjelennek.
Felhasználói visszacsatolás formái

- **Hibás adatbevitel**: ha a backend validáció hibát küld vissza (pl. jelszó túl rövid, hiányzó mező), a mező alá piros színű szöveg kerül
- **Sikeres művelet**: például regisztráció után a felhasználót átirányítja a rendszer, és az új nézeten megerősítő szöveg jelenhet meg
- **Form mezők fókusz állapota**: Bootstrap által formázott :focus stílusok segítik a navigációt
- **Hover-effektusok**: gombok, linkek vizuálisan reagálnak az egérmozgásra

Ezek a visszajelzések segítik a felhasználót abban, hogy mindig tudja, mi történik a

rendszerben, és mikor milyen műveletet hajtott végre.

Logikai szempontok

A visszajelző megoldások Simon Vue-komponensein belül történnek. A v-model kötések révén a form mezők állapotát valós időben figyeli, a backendből érkező hibák vagy válaszok pedig a axios.catch() vagy axios.then() ágon keresztül kerülnek megjelenítésre. Ezeket egyszerű v-if blokkokkal jeleníti meg a megfelelő mezőknél.

Simon figyelt arra is, hogy a rendszerben minimális navigációval is elérhetők legyenek a fő funkciók, így a felhasználók kevesebb kattintással juthatnak el a célig (pl. egy termék igényléséig vagy kapcsolatfelvételig).

4.18 Horváth Csaba – Backend

A backend implementációja során Horváth Csaba külön figyelmet fordított arra, hogy minden egyes funkció megfeleljen a valós alkalmazásbeli elvárásoknak. A form validáció minden POST kérésnél megvalósul, ami kizárja az érvénytelen adatok backendre történő bejutását. A kapcsolatfelvételi rendszer szintén biztonságos, mivel csak bejelentkezett felhasználók tudják használni, és minden üzenet validálva van, mielőtt továbbításra kerülne a megadott e-mail címre.

A dokumentációban bemutatott példák – mint az AuthController vagy a MessageController – jól szemléltetik, hogy hogyan épülnek fel az egyes vezérlők, milyen validációs szabályokat alkalmazunk, és hogyan kezeljük az adatokat. A Mail funkció különösen hasznos, mert lehetővé teszi, hogy a rendszer közvetlenül kommunikáljon az adminisztrátorokkal a felhasználói oldalon kitöltött űrlap alapján. Ez a funkcionalitás szintén elősegíti a gördülékeny és személyes ügyfélkapcsolatok kialakítását.

A projekt során Horváth Csaba Laravel Sanctum szolgáltatással biztosította a hitelesítést. Ez a megoldás azért lett kiválasztva, mert kifejezetten SPA (Single Page Application) alkalmazásokhoz ajánlott, és jól működik együtt a Vue 3 + Vite alapú frontendtel. A auth:sanctum middleware segítségével könnyedén tudtuk korlátozni azokat az útvonalakat, amelyek csak hitelesített felhasználók számára érhetők el. Ez különösen fontos olyan érzékeny funkciók esetében, mint például az e-mail küldés, rendelések leadása, vagy a saját profiladatok lekérdezése.

Fontos szempont volt az adatbázis helyes struktúrája is. A Laravel migrációs rendszerének köszönhetően pontosan lehetett szabályozni az adattáblák felépítését, a mezők típusát és azok kapcsolatát. Így könnyedén létrehozható volt a users, products, orders és messages táblák logikus és áttekinthető kapcsolata. Az orders tábla például minden rendelést naplóz, és ezzel lehetővé teszi, hogy a felhasználók nyomon követhessék támogatási előzményeiket, míg az admin felület ezeket moderálni tudja.

A hibák gyors azonosításátra Horváth Csaba a Laravel saját hibakezelő rendszerét használta, ezzl biztositotta a debugolását a fejlesztési környezetben, amikor az APP_DEBUG értéke true. Ezzel azonositotta az esetleges konfigurációs hibákat, validációs problémákat vagy route elírásokat. A projekt .env fájlja részletesen konfigurálva lett, beleértve a CSRF védelmet, a hitelesítési domaineket (SANCTUM_STATEFUL_DOMAINS=localhost:5173), és a SESSION_DOMAIN beállítását is.

Összegzés:
A backend fejlesztését Horváth Csaba alapozta meg, aki gondoskodott a rendszer megbízhatóságáról, biztonságáról, és a Vue.js frontenddel való zökkenőmentes integrációról.

4.19 Szőke Simon – Interaktív funkciók és felhasználói bevonás

Szőke Simon a SegítsVelem frontend oldalainak interaktív viselkedéséért is felelt. Munkája nemcsak a felületek kialakítására terjedt ki, hanem azokra az elemekre is, amelyek közvetlen felhasználói műveletekre reagálnak. A rendszer teljes mértékben Vue 3 és Bootstrap 5 keretrendszerre épül, így az interakciók megvalósítása is ezekre a technológiákra támaszkodik.
Simon kiemelten figyelt arra, hogy minden fontos művelethez tartozzon egyértelmű visszajelzés – így a felhasználó mindig tisztában van azzal, hogy sikerrel járt-e, hibát követett el, vagy mi a következő lépés.
A legfontosabb interaktív funkciók:

- **Termékkártyák interaktív működése**
  A ProductCard.vue komponens minden egyes ruhadarabot megjelenít. A kártya tartalmaz egy "Igénylem" gombot, amelyre kattintva backend-hívás történik. Simon gondoskodott arról, hogy igénylés után a gomb állapota megváltozzon, vagy a felhasználó új státuszt lásson a termékhez.
- **Űrlapvalidáció és visszajelzés**
  A RegisterView.vue, LoginView.vue és ConnectionsView.vue űrlapok v-model és v-if alapú hibakezelést tartalmaznak. Backendről érkező hiba esetén a hibák a mező alatt piros szöveggel jelennek meg.
- **Felhasználói visszajelzés műveletek után**
  Sikeres regisztráció vagy bejelentkezés után a felhasználót átirányítja a rendszer, és új nézeten köszöntő szöveg vagy a következő funkció jelenik meg. Bár a projekt nem használ külön animált toast- vagy modal rendszert, a vizuális visszajelzés alapvető szinten jelen van.
- **Navigáció állapotkezelés**
  A Navbar.vue komponens mobilnézetben hamburger menüvé alakul. Az aktív útvonal kiemelést kap a menüben, így a felhasználó látja, hol tartózkodik az oldalon belül.
- **Hover-effektek és interakciók**
  A termékkártyák és fő gombok hover-re vizuálisan változnak (pl. árnyék, szín), amit Simon Bootstrap osztályokkal és egyszerű scoped CSS-sel oldott meg.

UX- és hozzáférhetőségi szempontok
Simon munkája során törekedett arra, hogy a fő funkciók kevés kattintással elérhetők legyenek, és a felhasználó soha ne maradjon visszajelzés nélkül. Bár a rendszer nem tartalmaz dedikált akadálymentesítési funkciókat (pl. aria-label, tabindex), a Bootstrap alapú markup és gombkezelés részben kiszolgálja a billentyűzetes navigációt és képernyőolvasók működését.
A Vue komponensek úgy lettek megtervezve, hogy különállóan is működjenek, és minden egyes interakcióval kapcsolatos adatmozgás (igénylés, űrlapküldés) Axios segítségével történik, frissítés vagy reload nélkül.

Összegzés
Szőke Simon munkája biztosította, hogy a SegítsVelem oldal ne csak statikus bemutatóoldal legyen, hanem interaktív, dinamikusan reagáló webalkalmazás. Az általa kialakított komponensek és nézetek nemcsak látványosak, hanem funkcionálisan is reagálnak a felhasználók műveleteire. Munkája jelentős részben hozzájárult ahhoz, hogy a rendszer modern, reszponzív, és élményszerű maradjon minden felhasználói szinten.

5. Továbbfejlesztési lehetőségek

A rendszer alapverziója teljes mértékben működőképes, és Vue.js + Laravel REST API architektúrájának köszönhetően könnyen bővíthető. Az alábbi fejlesztési irányok valósíthatók meg a jövőben:

- **Mobilalkalmazás készítése**:
- A meglévő REST API struktúra lehetővé teszi egy Flutter vagy React Native alapú mobilalkalmazás kiépítését, amely támogatja az adományozási folyamatok mobilról történő kezelését.
- **Email értesítések**:
- Automatikus email küldése a felhasználónak igénylés esetén, valamint az admin értesítése új kérés beérkezésekor.
- **Képkezelés fejlesztése**:
- Több kép feltöltésének lehetősége egy termékhez, valamint automatikus képméretezés a gyorsabb betöltés és tárhelyoptimalizálás érdekében.
- **Közösségi funkciók**:
- Kommentelés, termékértékelés, valamint ajánlott termékek megjelenítése a felhasználói interakciók alapján.
- **Helyalapú keresés**:
- Adományozók és rászorulók összekapcsolása földrajzi elhelyezkedés szerint, akár térképes felületen keresztül.
- **Kapcsolatfelvétel bővítése**:
- A jelenlegi űrlap továbbfejlesztése például témaválasztással, válaszidő kijelzésével vagy korábbi üzenetek elérhetőségével.
- **Valós idejű állapotkezelés**:
- A ruhák státuszának nyomon követése (pl. elérhető, igényelt, jóváhagyott), valamint admin általi igényléskezelés megvalósítása.
- **Chat modul**:
- Közvetlen kommunikáció létrehozása az igénylő és az adományozó között, valós idejű üzenetküldéssel. Ez azonban jelentősebb backend és jogosultságkezelési logikát igényel.

5.1 Lehetséges modulok és integrációk

A SegítsVelem rendszer jelenlegi felépítése lehetővé teszi, hogy a jövőben különféle modulokkal vagy külső szolgáltatásokkal bővüljön. Az alábbiakban olyan fejlesztési irányokat sorolunk fel, amelyek technológiailag megvalósíthatók és társadalmilag is hasznosak lehetnek:

1. **Mobilalkalmazás**
2. Laravel REST API-ra építve fejleszthető egy mobilalkalmazás (pl. Flutter, React Native), amely lehetővé tenné a kényelmes igénylést vagy adományozást

útközben.

3. **Többnyelvűség**
4. A Laravel lokalizációs rendszere támogatja a többnyelvű működést, így az oldal később angol, német vagy roma nyelven is elérhetővé válhat.
5. **Push értesítések**
6. Mobil vagy böngésző alapú értesítések bevezetésével a felhasználók valós időben kaphatnának információt új termékekről vagy igénylés státuszváltozásokról.
7. **Gamifikációs elemek**
8. Jelvények, ranglisták vagy statisztikák ösztönözhetnék az aktivitást, különösen az adományozók és önkéntesek körében.
9. **Térképes keresés**
10. Ruhák lakhely vagy körzet szerint is böngészhetők lennének, és lehetőség nyílna gyűjtőpontok kezelésére is.
11. **PDF generálás**
12. Igénylésekhez vagy átvételekhez kapcsolódó igazolások, regisztrációs nyugták PDF formátumban letölthetők lennének.
13. **Admin szerepkörök bővítése**
14. A jelenlegi admin jogosultságok tovább bonthatók (pl. statisztikai megtekintő, adatrögzítő, tartalomfelelős).

5.2 Projekt értékelése fejlesztői szemmel

A SegítsVelem vizsgaremek projekt fejlesztése során értékes gyakorlati tapasztalatot szereztünk a csapatmunka megszervezésében, a modern webfejlesztési technológiák alkalmazásában, valamint a REST API szemléletű backend és a Vue 3 frontend integrációjában. A fejlesztés során számos kihívásba ütköztünk – különösen a jogosultságkezelés, az adatkezelés és a komponensalapú felépítés terén –, de ezek mind hozzájárultak szakmai fejlődésünkhöz.

A projektet egy társadalmilag hasznos cél szolgálatába állítottuk: hátrányos helyzetű gyermekek sportfelszereléshez juttatását kívánjuk támogatni digitális eszközökkel. A rendszer jelenlegi formájában reszponzív, biztonságos, könnyen kezelhető, és technológiailag nyitott a további bővítésekre.

A fejlesztés során megtanultuk:

- hogyan osszuk fel hatékonyan a munkát egy többfős csapatban,
- hogyan alkalmazzuk a Gitet és a GitHubot verziókövetésre és átlátható együttműködésre,
- hogyan tervezzünk adatbázismodellt és API-struktúrát Laravel keretrendszerben,
- hogyan építsünk Vue 3 alapú felhasználói felületeket,
- hogyan kezeljük a jogosultságokat és validáljuk a bemeneti adatokat biztonságosan.

Büszkék vagyunk arra, hogy a projektet a tervezett irányban meg tudtuk valósítani, és úgy érezzük, hogy nemcsak technikai tudásban, hanem közösen végzett fejlesztési gyakorlatban is sokat fejlődtünk.

6. A rendszer társadalmi haszna és fenntarthatóság

A SegítVelem rendszer nem csupán technikai kihívás volt számunkra, hanem tudatos társadalmi szerepvállalás is. Generációnkat sokszor éri az az igazságtalan kritika, hogy nem törődünk semmivel, állandóan a gép előtt ülünk/ a telefonunkat nyomkodjuk. A technikai fejlődésnek köszönhető megváltozott kommunikáció valóban szerves része az életünknek, de semmivel sem vagyunk kevésbé érzékenyek mások nehézségeire, mint a korábbi generációk. Úgy gondoljuk, inkább arról van szó, hogy minket más fórumokon és talán más megközelítéssel lehet elérni, mint a szüleink generációját. Kiemelten fontosnak tartjuk a környezetvédelmet, a környezettudatos életmódot szemben a fogyasztói társadalom vásárlási kényszerével, az újrafelhasználás sokféle lehetőségének támogatását.
Projektünk tudatos céljai:

1. **Környezeti fenntarthatóság**: a jó állapotú használt felszerelések és ruhák esetében egyértelmű, hogy mind a hulladék mennyiségének, mind a textilipar környezeti terhelésének csökkentése fontos járulékos hatás amellett, hogy a kedvező áru, jó minőségű felszereléseket rászoruló gyerekek életét segítik. Az új ruhák megvásárlása esetén kiemelt figyelmet fordítunk arra, hogy olyan gyártókat keressünk, akik tudatosan vállalják és követik a környezettudatos elveket.
2. **Társadalmi felelősségvállalás:** A platform nem csak a rászorulók segítését célozza, hanem a társadalmi érzékenyítés fontos eszköze is. A közösség bevonásával – adományozók, gyűjtőpontok, támogatók – megerősödik a társadalmi összetartás érzése.
3. **Nevelő hatás:** A fiatal felhasználók számára is bemutatható a rendszer működése, így érzékenyebbé válnak a társadalmi problémák iránt, és könnyebben válnak aktív tagjaivá a közösségi segítésnek.
4. **Önkéntesség és részvétel:** A rendszer támogatja, hogy bárki – iskolák, sportegyesületek, vállalatok – bekapcsolódhasson a folyamatba. Ezzel a SegítsVelem nem csupán egy webes eszköz, hanem egy mozgalom is lehet.

7. Felhasználói visszajelzések és statisztikák hasznosítása

A rendszer jövőbeli fejlesztésének fontos iránya lehet a felhasználói visszajelzések rendszerszintű gyűjtése és az ezekből származó statisztikai elemzések automatizálása. A visszajelzések alapján jobban megérthetjük, hogyan használják az emberek a rendszert, mely funkciókat tartják hasznosnak, és hol tapasztalnak nehézséget.

Lehetséges visszajelzési funkciók:
- Visszajelzés az admin által nyújtott kommunikációról (segítőkészség, gyorsaság)
- Szöveges megjegyzések beküldése profiloldalról
- Esetleges hibák jelentése a felhasználói felületen keresztül

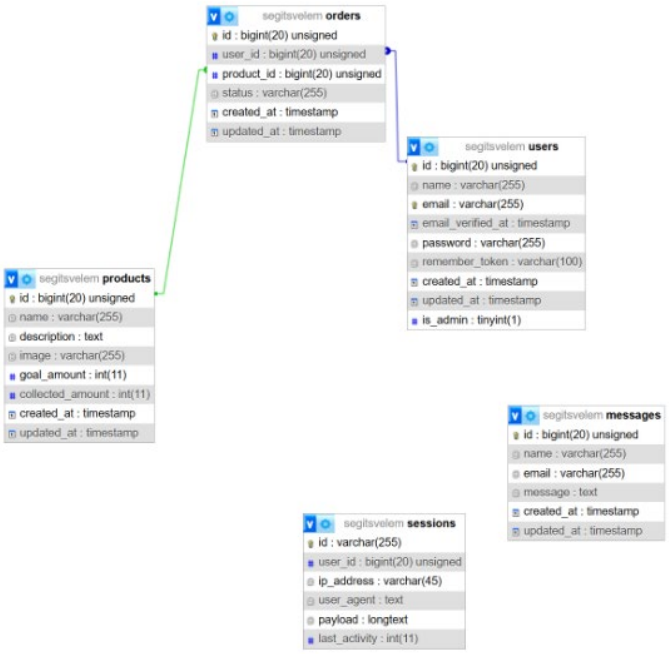
Statisztikai elemzések:
- Legnépszerűbb terméktípusok, márkák, méretek
- Mely megyékből érkezik a legtöbb igénylés
- Mennyi idő alatt reagálnak az adminok a kérelmekre
- Átlagos feldolgozási idő (igénylés → elfogadás)

8. Összegzés

A **SegítsVelem** projekt célja egy olyan közösségi platform létrehozása, amely lehetővé teszi a jó állapotú sportruhák hatékony újraelosztását rászoruló gyermekek számára. Az alkalmazás a Laravel 11 keretrendszerre és egy API-alapú felhasználói felületre épül, így egyaránt kiszolgálja a felhasználók és az adminisztrátorok igényeit.
A RESTful API strukturáltsága, az átgondolt adatbázismodell és a letisztult üzleti logika együttesen skálázhatóvá és fenntarthatóvá teszik a rendszert. A közös fejlesztői munka során elért célkitűzéseink nemcsak technikai, hanem valós társadalmi értéket is képviselnek.
A jövőbeli bővítések révén még hatékonyabban támogathatjuk a gyermekek sportolását a szükséges ruházattal, ezáltal ösztönözve az egészséges életmódot és az esélyegyenlőséget.

9. Források, ábrajegyzék

Ábrajegyzék:
1. ábra – Adatbázis ER-diagram:



Irodalmi és online források:

- Laravel dokumentáció – https://laravel.com/docs
- Bootstrap CSS Framework – https://getbootstrap.com
- Stack Overflow közösségi fórum – https://stackoverflow.com
- PHP dokumentáció – https://www.php.net
- W3Schools Laravel tutorial – https://www.w3schools.com/laravel
- Vue.js dokumentáció – https://vuejs.org/guide
- Vite build tool – https://vitejs.dev/guide
- Axios dokumentáció – https://axios-http.com/docs/intro
- MySQL dokumentáció – https://dev.mysql.com/doc/
- GitHub Docs (branching, pull request) – https://docs.github.com/en
- FontAwesome ikonok – https://fontawesome.com
- MDN Web Docs (általános HTML, CSS, JS) – https://developer.mozilla.org
- Laravel Sanctum dokumentáció (ha használtátok) – https://laravel.com/docs/sanctum