



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

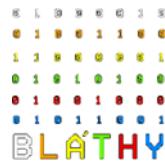
SegítsVelem

Készítette:
Szőke Simon

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@gmail.com

A Vizsgaremek téma:

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. szeptember 20.

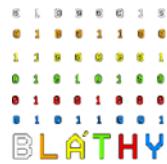
A handwritten signature in blue ink that reads "Szőke Simon". Below the signature is a dotted line for the name.

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. szeptember 20.


.....

Szőke Simon
Tanuló aláírása

Tartalomjegyzék:

1.	Projekt ismertetése	5
2.	Felhasználói dokumentáció	5
2.1	Felületek ismertetése.....	5
2.2	Adminfelület ismertetése	8
2.3	Funkciók ismertetése	8
2.4	Felhasználói használati utasítás	9
2.5	Felhasználói nézetek részletes leírása.....	10
2.6	Felhasználói élmény és UI/UX tervezés	14
2.7	Felhasználói esettanulmány	15
2.8	További felhasználói történetek.....	16
2.9	UI elemek és visszajelzések.....	17
2.10	Teljes felhasználói workflow részletesen	17
2.11	Rendszer működés szimuláció – teljes példafolyamat.....	18
2.12	Felhasználói interakciós folyamat.....	18
3.	Fejlesztői dokumentáció.....	19
3.1	fejlesztői tevékenységek	19
3.2	Telepítési dokumentáció	20
3.3	Adatbázis dokumentáció	21
3.3.1	Diagram	21
3.3.2	Mezők részletes ismertetése.....	22
3.3.3	Kapcsolatok indoklása ismertetése	23
3.4	Komponens/struktúra dokumentáció	23
3.4.1	Backend elemei.....	23
3.4.2	Főbb elemek részletes ismertetése	24
3.5	API dokumentáció	24
3.5.1	Felhasználói autentikációhoz kapcsolódó végpontok.....	24
3.5.2	API válaszformátumok példákkal.....	25
3.5.3	Rendelések (támogatások)	26
3.5.4	Kapcsolatfelvétel	26
3.5.5	Adminisztrációs végpontok	26
3.6	Teszt dokumentáció	27
3.7	Backend működés részletezése	27
3.7.1	E-mail küldés	28
3.7.2	E-mail küldési logika	28
3.7.3	Küldési folyamat (Kapcsolatfelvétel)	28
3.7.4	.env konfiguráció és e-mail beállítások	29

3.7.5 Jogosultság és hitelesítés	29
3.7.6 Hibakezelés	29
3.8 Admin dashboard működése.....	30
3.8.1 A dashboard fő funkciói	30
3.8.2 Termékek kezelése.....	31
3.9 Vue nézetek és mappaszervezés	31
3.10 Middleware és biztonság.....	32
3.10.1. Middleware csoportok	32
3.10.2. Laravel Sanctum autentikáció.....	33
3.10.3. CORS konfiguráció.....	33
3.10.4. Adatvalidáció	34
3.10.5. Admin jogosultság	34
3.10.6. Session és CSRF védelem.....	34
3.11 Alkalmazott technológiák és eszközök.....	35
3.12 Adatbázis séma elemzés	35
products tábla – Támogatási célok vagy kampányok	35
sessions tábla – Munkamenetek.....	36
sessions tábla – Munkamenetek.....	36
3.13 Admin funkciók workflow.....	37
3.14 Adatvédelem és biztonság.....	37
4. Részletes fejlesztői tevékenységek.....	37
4.1 Felhasználói felület és dizájn	37
4.2 Backend és adatmodell	38
4.3 Admin input mezők fejlesztése	39
4.4 Designelvek és UI irányelvek részletesen.....	40
4.5 API architektúra részletesen	40
4.6 Felhasználói élmény és vizuális egység megvalósítása	42
4.7 Backend optimalizálás, teljesítmény és tesztelhetőség	43
4.7.1. Kódstrukturálás és optimalizált controller használat	43
4.7.2. Adatbázis-lekérdezések hatékonysága.....	43
4.7.3. Middleware szűrés és jogosultság-ellenőrzés	44
4.7.4. Cache és session kezelés.....	44
4.7.5. Tesztelhetőség – gyakorlati megközelítés	45
4.8 Reszponzív tervezés és felhasználói visszajelzések kezelése	45
4.9 Adatstruktúrák optimalizálása és relációk finomítása	46
4.9.1. Relációk megvalósítása.....	46
4.9.2. Eager Loading és teljesítmény	47
4.9.3. Adattípusok optimalizálása.....	47
4.9.4. Migrációs fájlok struktúrája	47
4.9.5. Jövőbeli bővíthetőség	48
4.10 Felhasználói pszichológia és interaktív élmény tervezése.....	48

4.11 Működésbiztonság és hibakezelés	49
4.11.1. Autentikáció és hozzáférés-kezelés	49
4.11.2. Adatvalidáció és hibakezelés	49
4.11.3. Központi hibakezelés és naplózás.....	50
4.11.4. CORS beállítások és HTTP védelem.....	50
4.11.5. SQL injekció elleni védelem.....	50
4.11.6. Session és cookie biztonság.....	50
4.11.7. E-mail küldés és továbbítás	51
4.11.8. Felhasználói jogosultságok szétválasztása.....	51
4.12 Komponensalapú frontend struktúra Vue 3 keretrendszerben.....	52
4.13 Teljes oldalkészítési folyamat és komponenslogika	54
4.13.1. Cél meghatározása	54
4.13.2. Szerkezeti felépítés Vue komponensben	54
4.13.3. Komponensek beillesztése.....	54
4.13.4. Validáció és adatkezelés	55
4.13.5. Reszponzivitás ellenőrzése	55
4.13.6. Tesztelés és iteráció	55
4.13.7. Komponenslogika és újrahasznosíthatóság	55
4.14 Adminfelület	56
4.15 Frontend és adminpanel dizájn	56
4.16 Backend API és adatkezelés	57
4.16.1. Fő erőforrások (jelenleg használt végpontok):	57
4.16.2. Adatkezelés Eloquent ORM-mel	58
4.16.3. Felhasználói adatok kezelése	58
4.16.4. Adminisztrációs funkciók	58
4.16.5. Üzenetküldés e-mailben.....	59
4.16.6. Biztonságos adatfeldolgozás	59
4.16.7. Tokenkezelés és authentikáció.....	59
4.17 Stíluskezelés és vizuális visszajelzések beépítése Vue 3 frontendben	60
4.18 Backend.....	61
4.19 Interaktív funkciók és felhasználói bevonás	62
5. Továbbfejlesztési lehetőségek.....	63
5.1 Lehetséges modulok és integrációk	64
5.2 Projekt értékelése fejlesztői szemmel	65
6. A rendszer társadalmi haszna és fenntarthatóság	66
7. Felhasználói visszajelzések és statisztikák hasznosítása.....	67
8. Összegzés	68
9. Források.....	69
10. Ábrajegyzék	69

1. Projekt ismertetése

A „SegítsVelem” egy digitális adományozási platform, amelynek célja a használt, jó állapotú sportruházatok összegyűjtése és eljuttatása rászoruló gyermekek számára.

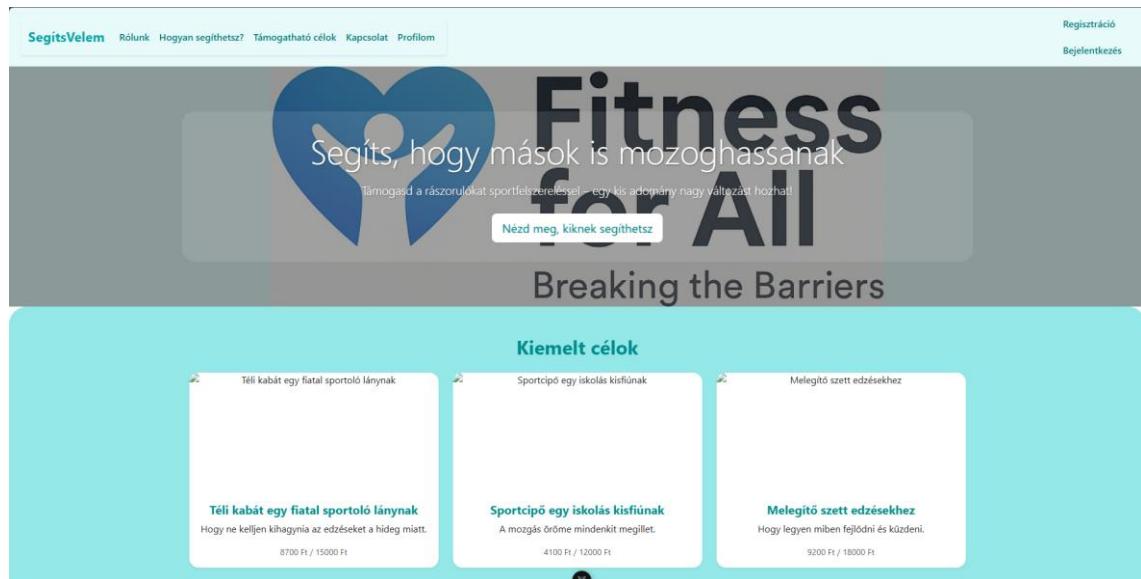
A platform frontendből, backendból és a hozzá kapcsolódó adatbázisból áll. A frontend külön Vue 3 + Vite alapú SPA (single page application), a Laravel pedig REST API-t szolgáltat. A rendszer lehetőséget biztosít a felhasználók regisztrációjára, bejelentkezésére, a ruhák böngészésére, valamint egyéb adminisztratív funkciókra is. A céлом ehhez egy letisztult, biztonságos és könnyen kezelhető rendszer fejlesztése volt, amely lehetőséget ad arra, hogy bárki hozzájárulhasson a közösség támogatásához, akár igénylőként.

A fejlesztés során külön figyelmet fordítottam 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.



1. ábra – kezdőoldal PC



2. ábra – kezdőoldal mobil

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**”.



3. ábra – fejléc

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.

2.2 Adminfelület ismertetése

A SegítsVelem rendszer tartalmaz egy különálló adminfelületet is, amely kizárolag az adminisztrátorok számára érhető el. Az adminfelületen keresztül lehetőség van a platform tartalmainak kezelésére. A belépéshoz jogosultság szükséges.

Az adminfelület funkciói közé tartoznak:

Termékek kezelése: új támogatási célok (ruhák) hozzáadása, meglévő adatok módosítása vagy törlése.

Statisztikák: összesített adatok megtekintése az aktív támogatási célokról, összegyűjtött adományokról.

2.3 Funkciók ismertetése

A SegítsVelem rendszer a következő fő funkciókkal rendelkezik:

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.

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.

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.

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.

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.

Rólunk és Hogyan segíthetsz? – Tájékoztató oldalak, amelyek bemutatják a projekt hátterét, valamint különféle lehetőségeket ismertetnek az adományozásra vagy segítségnyújtásra.

Felhasználói profil – A bejelentkezett felhasználók hozzáférhetnek saját profiloldalukhoz, ahol megtekinthetik személyes adataikat. (Jelenleg a felhasználói aktivitás, például korábbi igénylések megjelenítése nincs implementálva.)

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.

2.4 Felhasználói használati utasítás

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

Kattints a „**Regisztráció**” menüpontra a fejlécben.

Töltsd ki a nevedet, email-címedet és jelszavadat, majd kattints a „*Regisztráció*” gombra.

Ha már regisztráltál, kattints a „**Bejelentkezés**” gombra.

Add meg az email-címedet és jelszavadat, majd nyomd meg a „*Bejelentkezés*” gombot.

Kattints a „**Támogatható célok**” menüpontra, hogy böngéssz az elérhető termékek között.

Nézd meg a ruhákhoz tartozó képeket, leírásokat és a még szükséges támogatási összeget.

Lépj be a „**Kapcsolat**” menüpontra, ha üzenetet szeretnél küldeni a szervezetnek.

Írd be a nevedet, email-címedet és az üzenetedet, majd kattints a „*Küldés*” gombra.

Válaszd a „**Profilom**” menüpontot, ha meg szeretnéd tekinteni saját adataidat.

Itt elérheted a személyes profilon alapinformációit.

Adminisztrátorok számára — rövid útmutató

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

A belépés után tekintsd meg a fő admin felületet, ahol láthatod az összesített statisztikákat (pl. termékek száma).

Válaszd a „**Termék hozzáadása**” menüpontot, ha új adományozási célokat (pl. ruhákat) szeretnél felvinni.

Add meg a termék nevét, leírását, célösszegét, majd tölts fel képet, és kattints a „*Hozzáadás*” gombra.

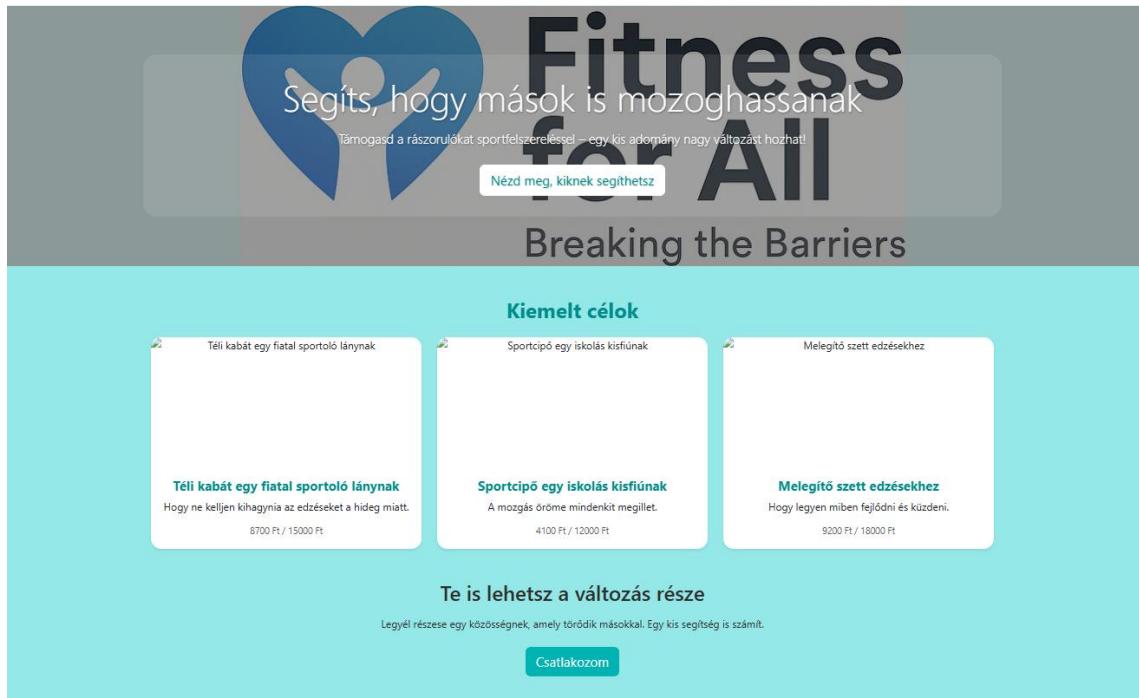
Használhatod a „Szerkesztés” és „Törlés” gombokat az egyes termékek mellett, ha módosítani vagy eltávolítani szeretnéd őket az adatbázisból.

2.5 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.



4. ábra – Kezdőlap (HomeView)

Támogatható célok oldal (ProductsView)

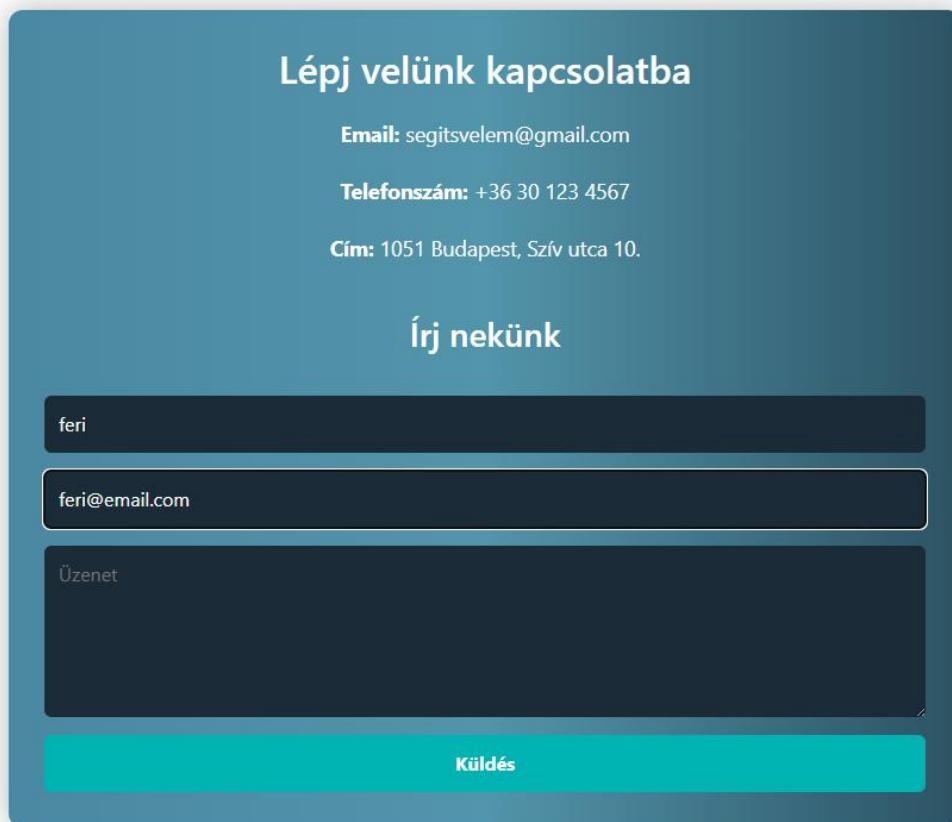
Ez az oldal listázza a támogatásra váró ruhákat és sportfelszereléseket. A termékek rácsoz galériában jelennek meg, minden egyes kártyán látható a kép, a név, a rövid leírás, valamint a gyűjtés állapota vizuálisan egy előrehaladási sáv segítségével. A kártyák reszponzív kialakításúak, de jelenleg nem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



5. ábra – Támogatható célok oldal (ProductsView)

Kapcsolat oldal (ConnectionsView)

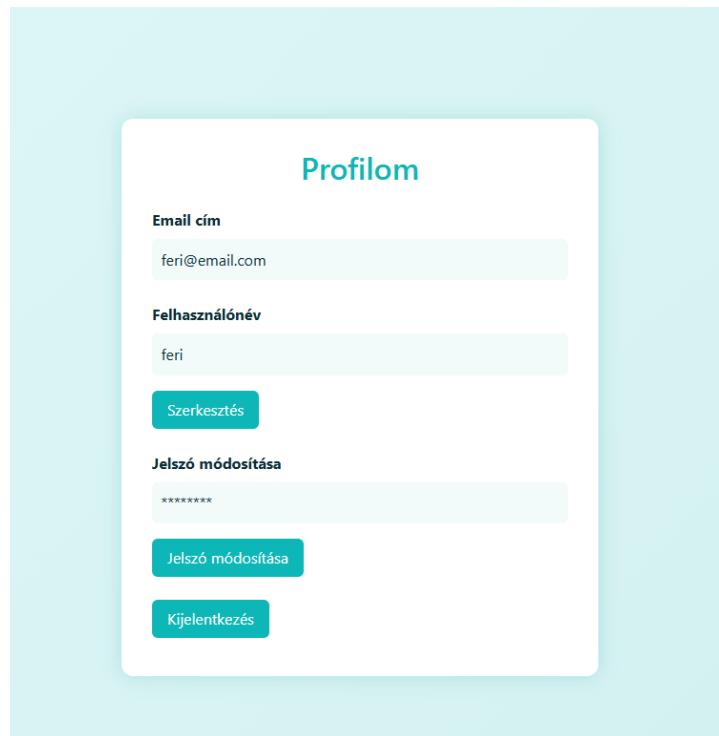
A kapcsolatfelvételi oldal egy űrlapot tartalmaz, amelyen keresztül a felhasználók üzenetet küldhetnek az oldal adminisztrátorainak. Az űrlap validációval van ellátva: hibás vagy hiányzó mező esetén hibaüzenet jelenik meg. Az üzenet sikeres beküldése után egy pozitív visszajelző üzenet (például modal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



6. ábra – Kapcsolat oldal (ConnectionsView)

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 státszok. A jelszómódosítás külön felületen történik.



7. ábra – Profil oldal (ProfileView)

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.

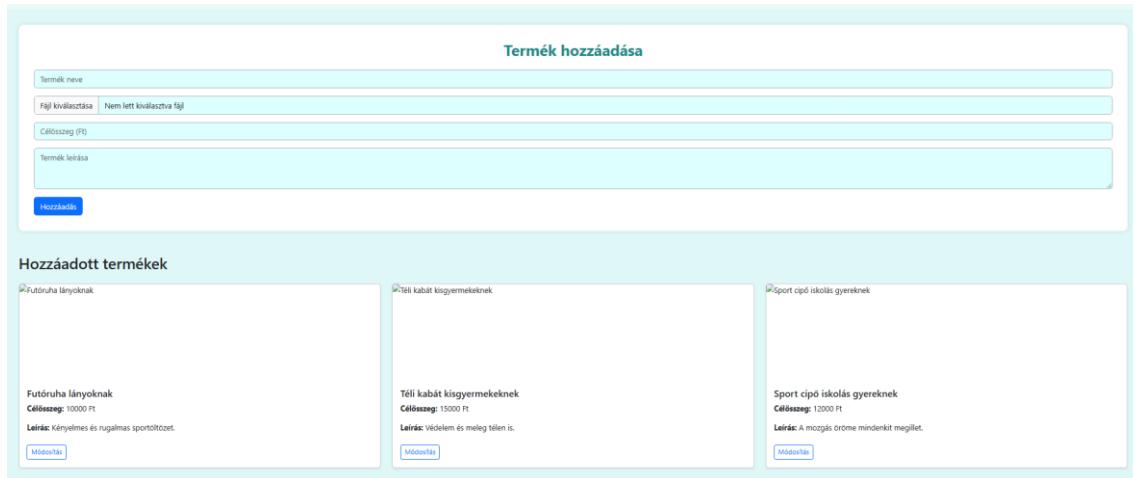
The screenshot displays two side-by-side forms. The left form is titled "Regisztráció" (Registration) and contains three input fields: "Felhasználónév" (Username), "Email cím" (Email address), and "Jelszó" (Password). Below these is a large teal "Regisztráció" (Register) button. Underneath the button are links for "Már van fiókod?" (Already have an account?) and "Bejelentkezés" (Login). At the bottom is a checkbox with the text: "A folytatással elfogadod a [Felhasználói feltételeket](#) és tudomásul veszed az [Adatvédelmi szabályzatot](#)." The right form is titled "Bejelentkezés" (Login) and contains two input fields: "Email cím" (Email address) and "Jelszó" (Password). Below these is a large teal "Belépés" (Login) button. To the right of the button is the text "Még nincs fiókod?". At the bottom is a teal "Regisztrálj itt" (Register here) button.

8. ábra – Regisztráció és bejelentkezés (RegisterView / LoginView)

Admin felület (AdminView)

A rendszer adminfelülete kizárolag admin jogosultsággal érhető el. A belépést követően lehetőség nyílik:

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



9. ábra – Admin felület (AdminView)

2.6 Felhasználói élmény és UI/UX tervezés

A SegítsVelem platform fejlesztése során kiemelt figyelmet fordítottam 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, pasztelkék és megbízhatóságot sugalló árnyalatokból áll. A szövegek jó kontrasztal jelentek meg a háttérhez képest, a címsorok kiemeltek, a betűtípus könnyen olvasható. A gombok és interaktív elemek mindenkorral 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 biztosítottam, 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.

2.7 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 gyermekéinek 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 célközönségünk től.

Egy barátja ajánlására azonban kíváncsiságból mégis felkereste az oldalamat. 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 látogatóm és rendszeres támogatóm. E-mailben szoktam beszélgetni, és elmondok neki, hogy nekem is voltak rossz élményeink a sporttal kapcsolatban. Most azon dolgozom, hogy meggyőzőm: kezdjen el újra mozogni – saját magáért.

2.8 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 örölt annak, hogy a rendszer átlátható, és vissza tudta nézni, mely termékek kerültek fel az oldalra az adományai közül.

2.9 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.

2.10 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:

Regisztráció: A felhasználó a főmenü 'Regisztráció' gombjára kattintva egy egyszerű űrlapot tölt ki (név, email, jelszó). A regisztráció sikeres lefutása után automatikusan bejelentkezik.

Bejelentkezés: A már regisztrált felhasználó a 'Bejelentkezés' oldalon adja meg hitelesítő adatait. Hibás adat esetén validációs hibaüzenet jelenik meg az űrlap alatt.

Termékek böngészése: A 'Támogatható célok' menüpontban a felhasználó megtekintheti az adományozható ruhákat. minden termékkártyán látható a név, a célösszeg, és az eddig összegyűlt támogatás.

Profil megtekintése: A 'Profilom' oldalon a felhasználó megtekintheti a saját regisztrációs adatait, valamint szerkesztheti a nevét vagy jelszavát. Igénylések státuszkezelése jelenleg nem elérhető.

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.

2.11 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 adatakat, 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.

2.12 Felhasználói interakciós folyamat

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

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 fejlesztői tevékenységek

A projektet minden elemét egyedül készítettem. Az oldal egy SPA (Single Page Application), amely Vue 3 frontend technológiát és Laravel alapú REST API-t használ a backend oldalon.

Az API-alapú felhasználói felületek fejlesztése közben az oldal teljes dizájnja is folyamatosan dolgoztam, beleértve az adminfelület vizuális megjelenését is.

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 választottam 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 tudtam megvalósítani a session-alapú tokenes bejelentkezést és felhasználói azonosítást.

A munkát GitHub segítségével koordináltam. A végleges megoldásokat pull requesteken keresztül egyesítettem a main ágba.

3.2 Telepítési dokumentáció

A projekt Laravel 12 alapokra épül, a telepítés lépései a következők:

Laravel Backend Telepítése

1. Klónozd a GitHub repót:

<https://github.com/TheGreatMrBurgir/Vizsgaremek>

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

composer install

3. Állítsd be a környezeti változókat (.env):

copy .env.example .env

4. Generálj alkalmazáskulcsot:

php artisan key:generate

5. Migráld és töltsd fel az adatbázist:

php artisan migrate --seed

6. Storage linkelése:

php artisan storage:link

7. Indítsd el a Laravel szervert:

php artisan serve

Az alkalmazás elérhető lesz a <http://127.0.0.1:8000> címen.

```
Backend (Laravel)

cd backend
composer install
cp .env.example .env
php artisan key:generate
php artisan migrate --seed
php artisan serve| You,
```

10. ábra – Backend 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.

```
Frontend (Vue.js):  
cd frontend  
npm install  
npm run dev
```

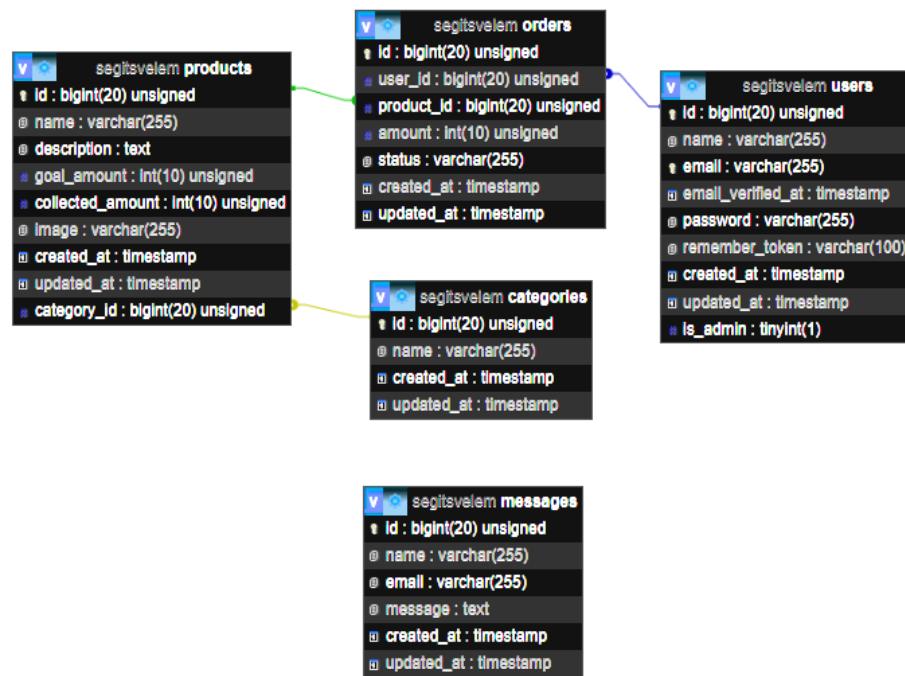
11. ábra – Frontend Telepítése

3.3 Adatbázis dokumentáció

3.3.1 Diagram

Az adatbázis 5 fő táblából áll:

- users
- products
- categories
- orders
- messages



12. ábra – ER-diagram

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
- goal_amount
- collected_amount
- image
- timestamps

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:

Felhasználó és Rendelés (users és orders): 1:N

Egy felhasználóhoz több rendelés tartozhat (orders.user_id → users.id).

Termék és Rendelés (products és orders): 1:N

Egy termékre több rendelés is érkezhet (orders.product_id → products.id).

Felhasználó és Hozzáférési tokenek (users és personal_access_tokens): 1:N (polimorf, Sanctum)

Egy felhasználóhoz több API-token tartozhat (Sanctum: personal_access_tokens.tokenable_type/id a userre mutat).

Admin jogosultság jelölése:

Az admin státusz a users táblában az is_admin mezővel van jelölve.

3.4 Komponens/struktúra dokumentáció

3.4.1 Backend elemei

A projekt Backendje Laravel 12 keretrendszerre épül, és MVC architektúrát követ. A fő komponensek:

- **Modellek**: A `User`, `Product`, `Order`, `Category`, `Message` osztályok reprezentálják az adatbázis táblákat.
- **Kontrollerek**: A `ProductController`, `OrderController`, `AuthController`, `AdminController` felelnek az üzleti logikáért és az API kiszolgálásért.
- **Routes**: Az `api.php` az összes RESTful végpontot tartalmazza.

3.4.2 Főbb elemek részletes ismertetése

ProductController:

- `index()`: Termékek kílistázása.
- `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 bővítte.

AdminController:

- admin végpontok: orders, messages, products; dashboard UI tervezett.

3.5 API dokumentáció

A Laravel 12 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 az 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.

```
// Nyitott auth végpontok
Route::post('/register', [AuthController::class, 'register'])->withoutMiddleware('auth:sanctum');
Route::post('/login', [AuthController::class, 'login'])->withoutMiddleware('auth:sanctum');

// Publikus
Route::get('/products', [ProductController::class, 'index']);
Route::get('/products/{id}', [ProductController::class, 'show']);
Route::post('/products', [ProductController::class, 'store']);
Route::put('/products/{id}', [ProductController::class, 'update']);
Route::delete('/products/{id}', [ProductController::class, 'destroy']);
//Route::post('/messages', [MessageController::class, 'store']); // NEM /contact
Route::post('/contact', [MessageController::class, 'store']);

// Bejelentkezett / Admin-only
Route::middleware('auth:sanctum')->group(function () {
    // User + logout
    Route::put('/changepassword', [AuthController::class, 'passwordReset']);
    Route::get('/user', fn(Request $r) => $r->user());
    Route::post('/logout', function (Request $r) {
        $r->user()->tokens()->delete();
        return response()->json(['message' => 'Kijelentkezve']);
    });

    // Saját rendelések
    Route::get('/orders', [OrderController::class, 'index']);
    Route::post('/orders', [OrderController::class, 'store']);

    // Admin-only
    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']);
    Route::get('/admin/messages', [MessageController::class, 'index']);
});


```

13. ábra – api.php kódja

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 a felhasználói támogatásokat rögzítik. minden rendelés egy konkrét termékhez tartozik és 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 végpontok auth:sanctum védelmet használnak, az adatok az orders táblában tárolódnak.

3.5.4 Kapcsolatfelvétel

A projekt egyik kulcsfontosságú funkciója, hogy a felhasználók üzenetet küldhessenek az üzemeltetőknek.

A felhasználók üzenetet tudnak küldeni az üzemeltetőknek.

Végpont: POST /api/messages – validált mezők: name, email, message.

A backend a MessageController@store metódusban:

elmenti az üzenetet a messages táblába,

a Mailgun REST API-n keresztül e-mailt küld az üzemeltetői címre.

A Mailgun hitelesítéshez a környezeti változókat használja (MAILGUN_API_KEY, MAILGUN_DOMAIN), amelyeket a projekt README is jelez. Siker esetén 200-as választ ad vissza (JSON üzenettel).

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 alkalmaztam.

Automatikus tesztelés:

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

Manuális tesztelés:

- Oldalak betöltése és navigáció (SPA)
- Reszponzív nézetek ellenőrzése
- Űrlapok validációja (regisztráció, belépés, üzenetküldés)
- Védett végpontok elérése bejelentkezve (/api/orders), admin műveletek ellenőrzése (approve/reject)

A manuális tesztek sikeresek voltak, a fő funkciók stabilan működnek.

3.7 Backend működés részletezése

A backend REST stílusú kontrollerekkel épül fel. A fő erőforrásokhoz egységes metódusok tartoznak (index, show, store, update, destroy). A hitelesítés Laravel Sanctum Bearer tokennel történik.

Fő kontrollerek:

- ProductController – termékek listája és részletei.
- OrderController – felhasználói rendelések listázása és létrehozása.
- AdminOrderController – rendelések áttekintése és státuszkezelés (approve/reject).
- MessageController – kapcsolatfelvételi üzenetek mentése.
- AuthController – regisztráció, belépés (token kiadás), kijelentkezés.

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 felenni vagy egyéb kéréseket közvetítenének. Az alkalmazás e-mail küldésre a Mailgun szolgáltatót használja. A kapcsolat a Mailgun hivatalos driverén keresztül történik, a hitelesítést környezeti változók biztosítják. A levelek küldése HTTP API-val történik, a beállításokat a .env és a Laravel mailer konfiguráció tartalmazza.

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 Mailgunhoz szükséges kulcsok és domain az .env fájlban szerepelnek:

```
MAIL_MAILER=mailgun
MAIL_FROM_ADDRESS=no-reply@<SAJAT-DOMAINED>
MAIL_FROM_NAME="Segíts Velem"
MAILGUN_DOMAIN=<SAJAT-DOMAINED>
MAILGUN_SECRET=<MAILGUN_API_KEY>
```

A küldés a Laravel Mail alrendszerén keresztül zajlik, a Mailgun driverrel. A konfiguráció lehetővé teszi a fejlesztői és éles környezetek szétválasztását (külön .env értékekkel)

3.7.3 Küldési folyamat (Kapcsolatfelvétel)

A kapcsolatfelvételi űrlap a POST /api/messages végponton keresztül működik. A backend a MessageController@store metódusban:

validálja a mezőket (name, email, message),

elmenti az adatot a messages táblába,

Mailgunon keresztül e-mailt küld az üzemeltetői címre.

Siker esetén JSON választ ad vissza (200 OK). A küldés a Mailgun HTTP API-ján keresztül történik a beállított feladói címről (MAIL_FROM_ADDRESS).

3.7.4 .env konfiguráció és e-mail beállítások

Az alkalmazás e-mail küldése Mailgun szolgáltatón keresztül történik. A szükséges beállítások az .env fájlból:

```
MAIL_MAILER=mailgun
MAIL_FROM_ADDRESS=no-reply@<SAJAT-DOMAINED>
MAIL_FROM_NAME="Segíts Velem"
MAILGUN_DOMAIN=<SAJAT-DOMAINED>
MAILGUN_SECRET=<MAILGUN_API_KEY>
```

Fejlesztéskor a tényleges kiküldés elkerülésére használható: MAIL_MAILER=log (ilyenkor a levelek a storage/logs/laravel.log fájlba kerülnek).

3.7.5 Jogosultság és hitelesítés

Végpont: POST /api/messages – validált mezők: name, email, message.

A kérés a MessageController@store metódust hívja, amely elmenti az üzenetet az adatbázisba, majd Mailgun API-n keresztül e-mailt küld az üzemeltetői címre.

A végpont a publikusan elérhető kapcsolati űrlapot szolgálja ki; a spam-kockázatot szerveroldali validációval és sebességkorlát-middleware-rel (throttling) csökkentjük.

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

- **Validációs hiba (422)**: a hiányzó/hibás name, email, message mezőkre részletes JSON válasz érkezik; a frontend ezeket a mezők alatt jeleníti meg.
- **Mailgun hiba (5xx / 401)**: a küldés sikertelenségekor a backend 500-as választ ad és logolja a kivételt; a kliens „Sikertelen küldés, próbáld újra később” üzenetet kap.
- **Throttling (429)**: túl sok próbálkozás esetén az API visszajelzi a sebességkorlátot.
- **Logolás**: minden hiba a storage/logs/laravel.log fájlba kerül.

3.8 Admin dashboard működése

Az admin dashboard kizárolag admin jogosultságú felhasználók számára érhető el. Bejelentkezés után a rendszer a felhasználó `is_admin` mezője alapján engedélyezi az admin felület elérését. A felületen az admin az igényléseket (orders) tudja áttekinteni és státuszt kezelni (jóváhagyás / elutasítás). A navigáció Vue Routerrel történik, a megjelenés Bootstrap 5 alapú.

3.8.1 A dashboard fő funkciói

Termékek kezelése

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.

Statisztikai áttekintés

A főoldalon alapvető adatok jelennek meg, mint például a feltöltött termékek száma.

Kapcsolati űrlap üzenetei (*jövőbeli lehetőség*)

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).

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

Az API végpontok az api csoport alatt futnak (kérésszám-korlát és paraméter-binding). A projekt Laravel 12-es konfigurációja a bootstrap/app.php fájlban állítja be az api csoportot (ThrottleRequests, SubstituteBindings), és a védett végpontokat az auth:sanctum middleware védi.

```
return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        api: __DIR__.'/../routes/api.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware(function (Middleware $middleware) {
        $middleware->appendToGroup('api', [
            \Illuminate\Routing\Middleware\ThrottleRequests::class . ':60,1',
            \Illuminate\Routing\Middleware\SubstituteBindings::class,
        ]);
        $middleware->appendToGroup('web', [
            //\App\Http\Middleware\EncryptCookies::class,
            \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
            \Illuminate\Session\Middleware\StartSession::class,
            \Illuminate\View\Middleware\ShareErrorsFromSession::class,
            \Illuminate\Foundation\Http\Middleware\VerifyCsrfToken::class,
            \Illuminate\Routing\Middleware\SubstituteBindings::class,
        ]);
    })
    ->withExceptions(function (Exceptions $exceptions) {
        //
    })->create();
```

14. ábra – app.php kódja

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 hitelesítés personal access tokennel (Bearer) történik. A bejelentkezés (POST /api/login) után a backend egy Bearer tokent ad vissza; a kliens ezt minden kéréshez elküldi az Authorization: Bearer <token> fejlécvel. A védett végpontok az auth:sanctum middleware alatt érhetők el. 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 domainról (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 adatfogadó végpont validációt használ (pl. POST /api/register, POST /api/login, POST /api/orders, POST /api/messages). A validációt a kontrollerek ->validate([...]) hívásai végezik, a frontend a hibákat mezőnként jeleníti meg. 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 kezelní és megjeleníteni.

3.10.5. Admin jogosultság

A session-kezelés SESSION_DRIVER=database beállítással történik (belépés a webes részekhez). Az API védett hívásai Bearer tokennel működnek, ezért CSRF védelem az API-hoz nem szükséges. A CSRF a web csoporthoz tartozó űrlapoknál aktív.

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 alkalmaztam, amelyek lehetővé tették egy stabil, biztonságos és jól skálázható rendszer kialakítását:

Laravel 12 – 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

users tábla – Felhasználók

A regisztrált felhasználók adatait tartalmazza.

products tábla – Támogatási célok vagy kampányok

Azokat az elemeket tartalmazza, amelyekhez támogatást lehet küldeni.

orders tábla – Megrendelések / támogatások

A felhasználók által leadott rendeléseket, illetve támogatásokat rögzíti.

Kapcsolatok:

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

sessions tábla – Munkamenetek

A bejelentkezett felhasználók munkameneteinek kezelésére szolgál.

messages tábla – Üzenetek

A kapcsolatfelvételi űrlapon keresztül beküldött üzeneteket tartalmazza.

orders tábla – Megrendelések / támogatások

A felhasználók által leadott rendeléseket, illetve támogatásokat rögzíti.

Kapcsolatok:

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

sessions tábla – Munkamenetek

A bejelentkezett felhasználók munkameneteinek kezelésére szolgál.

messages tábla – Üzenetek

A kapcsolatfelvételi űrlapon keresztül beküldött üzeneteket tartalmazza.

3.13 Admin funkciók workflow

Bejelentkezés után admin felhasználó az admin dashboardra kerül. Itt az alábbi funkciók érhetők el:

- **Rendelések listázása** (összes beérkezett igénylés)
- **Státuszkezelés:** rendelés jóváhagyása / elutasítása

A műveletek a backend admin végpontjait használják (GET /api/admin/orders, PUT /api/admin/orders/{id}/approve, PUT /api/admin/orders/{id}/reject).

3.14 Adatvédelem és biztonság

A rendszer fejlesztése során kiemelt figyelmet fordítottunk a felhasználói adatok védelmére és a biztonságos működésre. A Laravel 12 keretrendszer biztonsági funkciót a projekt teljes mértékben kihasználja az alábbi módokon:

Jelszavak titkosítása: bcrypt.

Token alapú hitelesítés: Laravel Sanctum Bearer token; a kliens az Authorization: Bearer <token> fejléccel küld kérést.

Input validáció: minden adatbevitel backend oldalon validált (SQLi/XSS ellen).

Jogosultságkezelés: auth:sanctum a védett végpontokon; admin hozzáférés a users.is_admin mező alapján.

CORS: a Vue fejlesztői origin engedélyezett.

Logolás: LOG_CHANNEL=stack, hibák a storage/logs/laravel.log-ba kerülnek.

4. Részletes fejlesztői tevékenységek

4.1 Felhasználói felület és dizájn

A frontend struktúrája Vue 3 + Vite alapokon készült, komponens-orientált felépítéssel és Bootstrap 5 támogatással. A fő nézetek közé tartozik a kezdőlap, a termékek galériás listázása, a kapcsolatfelvételi űrlap és a profiltnézet.

A felület reszponzív, mobilon és asztali nézetben is kényelmesen használható; a felhasználói élményt valós idejű visszajelzések, hibás mezők kiemelése, valamint szükség esetén modális ablakok és egységes UI-elemek erősítik.

A termékmegjelenítéshez kliensoldali logikát alkalmazok (lista, részletek), az API-hívásokhoz egységesített axios-réteg tartozik.

```
import axios from 'axios';

const http = axios.create({ baseURL: 'http://localhost:8000/api' });

http.interceptors.request.use((config) => {
  const token = localStorage.getItem('token');
  if (token) config.headers.Authorization = `Bearer ${token}`;
  config.headers['X-Requested-With'] = 'XMLHttpRequest';
  return config;
});

export const setToken = (token) => {
  if (token) localStorage.setItem('token', token);
  else localStorage.removeItem('token');
  if (token) axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;
  else delete axios.defaults.headers.common['Authorization'];
};

export default http;
```

15. ábra – axios hívás

Az admin felületen a rendelések áttekintése és státuszkezelése érhető el, a nézetek route-alapú navigációval és újrafelhasználható komponensekkel épülnek fel.

A backend oldalon az adatmodellek a Laravel Eloquent ORM segítségével valósulnak meg; a frontend integráció során a védett végpontok Bearer tokenes (Sanctum) hitelesítéssel működnek.

4.2 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. Fő modellek: User, Product, Order, Message.

- Product: name, description, image, goal_amount, collected_amount.
- Order: user_id, product_id, status (pl. pending/approved/rejected).
- Message: name, email, message – POST /api/messages ment az adatbázisba és Mailgun-on keresztül e-mailt küld.

A User–Order és Product–Order kapcsolatok 1:N kapcsolatban állnak.

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 példáulhasMany 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 Admin input mezők fejlesztése

Az adminfelület űrlapjait és vezérlőelemeit egységes minták szerint alakítottam ki: a rendelések áttekintéséhez tartozó nézetben a státuszváltást (jóváhagyás / elutasítás) külön komponensek kezelik, megerősítő interakcióval, betöltési állapotokkal és hibakezeléssel. A gombok és űrlap mezők következetes visszajelzést adnak (siker/hiba), a bevitelt kliensoldali ellenőrzés segíti, és minden művelet aszinkron API-hívással fut (Bearer tokennel). A komponenseket úgy terveztem, hogy újra felhasználhatók, reszponzívak és hozzáférhetők legyenek, így az admin folyamatai átláthatók és gyorsan végrehajthatók.

4.4 Designelvek és UI irányelvek részletesen

A frontend megjelenésén és a felhasználói élményen dolgoztam Vue 3 + Vite alapokon. Az interaktív elemeket (gombok, űrlapmezők, vizuális visszajelzések, modális ablakok, ikonok, rácsos elrendezések) én valósítottam meg, a dizájnt pedig letisztult, egyszerű és könnyen kezelhető irányelvek szerint alakítottam ki.

Vizuális és UX alapelvek:

Kontrasztos, a tartalomtól jól elkülönülő interaktív elemek.

Kiemelt színek az elsődleges funkcióknál (pl. bejelentkezés, regisztráció).

Rendezett űrlapok; piros hibajelzés és zöld sikeres-visszajelzés.

Mobilon hamburger menü, a termékrács listanézetre vált.

Újrafelhasználható Vue-komponenseket készítettem (termékkártyák, modális ablakok, űrlap inputok), hogy a projekt több részén egységesen és hatékonyan legyenek használhatók. A frontend stílusát és szerkezetét ennek megfelelően alakítottam ki.

4.5 API architektúra részletesen

A SegítsVelem projekt API architektúráját úgy terveztem 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 (például 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 útvonalcsoporthoz 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 regisztrációtól 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 kizárálag egy előre megadott emailcímmel rendelkező felhasználó (például) férhet hozzá ezekhez a funkcióhoz. 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 minden következetesek legyenek.

4.6 Felhasználói élmény és vizuális egység megvalósítása

Vue 3 + Vite környezetben dolgoztam egy átlátható, reszponzív, mobilbarát és vizuálisan egységes felület kialakításán. Elkészítettem a fő felhasználói nézeteket (kezdőoldal, termékgaléria, kapcsolat oldal, proflnézet), valamint a komponens-alapú rendszert (egyedi termékkártyák, navigációs elemek, modális ablakok).

Alkalmazott vizuális és UX elvek:

Mobilnézetben a navigáció hamburger menüre vált.

Kontrasztos színpaletta és letisztult tipográfia.

A rácsos termékmegjelenítés a képernyőmérethez igazodik.

Űrlapoknál egyértelmű hibajelzések (piros kiemelés), sikeres műveleteknél pozitív visszajelzés.

Interaktív hover-effektusok a kattintható elemeknél.

Egységes gomb-, input- és modál minták a teljes felületen.

Akadálymentességi szempontok (olvasható betűméret, megfelelő színkontraszt).

A vizuális felépítést és az újra használható komponenseket ennek az irányelvnek megfelelően alakítottam ki. A frontend kódstruktúrát logikusan szerveztem, hogy a karbantartás és a későbbi bővítés egyszerű legyen.

4.7 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.

4.7.1. Kódstrukturálás és optimalizált controller használat

A Laravel keretrendszer lehetőséget biztosít a vezérlők logikus szétválasztására. Ahelyett, hogy minden logikát egyetlen controllerbe írtam volna, külön controller szolgál:

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

Ez a komponens-orientált struktúra átláthatóbb kódot eredményez. A vezérlőkön belül az egyes műveletek önálló, könnyen tesztelhető metódusokra vannak bontva. Például az OrderController::store() az adatellenőrzésre, mentésre és a JSON válasz visszaadására koncentrál.

4.7.2. Adatbázis-lekérdezések hatékonysága

A Laravel Eloquent ORM használata mellett, ahol releváns, **eager loading**-ot alkalmazok az N+1 lekérdezési probléma elkerülésére. Például az admin rendelések listázásánál:

```
Order::with(['user', 'product'])->get();
```

Ez a lekérdezés előre betölti a kapcsolódó user és product adatokat, csökkentve az adatbázis-hívások számát. Emellett, ahol csak lehet, a select() használatával csak a szükséges mezőket kérem le:

```
User::select('id', 'name', 'email')->get();
```

Így kisebb adatmennyiség mozog, ami javítja a válaszidőt.

4.7.3. Middleware szűrés és jogosultság-ellenőrzés

A védett API-útvonalak **Bearer tokenes** hitelesítéssel érhetők el, auth:sanctum middleware alatt. Az útvonalak csoportosítása egyszerűsíti a route-fájlokat és csökkenti a hibalehetőségeket:

```
// Védett (bejelentkezett) végpontok
```

```
Route::middleware('auth:sanctum')->group(function () {
```

```
    Route::post('/orders', [OrderController::class, 'store']);
```

```
    // Admin műveletek
```

```
    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']);
```

```
});
```

```
// Publikus végpont (kapcsolatfelvétel) throttlinggal
```

```
Route::middleware('throttle:api')->post('/messages', [MessageController::class, 'store']);
```

Az admin jogosultság ellenőrzése **backend oldalon**, a felhasználó szerepére támaszkodva történik:

```
if (!$request->user() || !$request->user()->is_admin) {
```

```
    return response()->json(['message' => 'Nincs jogosultság'], 403);
```

```
}
```

4.7.4. Cache és session kezelés

A session-kezelés adatbázisban történik: SESSION_DRIVER=database.

A gyorsítótárazás előkészítve: .env-ben **CACHE_STORE=database** (a projekt jelenlegi méreténél nem kritikus, de bővítésnél hasznos).

4.7.5. Tesztelhetőség – gyakorlati megközelítés

A projektben a funkciók **manuális teszteléssel** kerültek ellenőrzésre (böngésző / Postman). A főbb ellenőrzési lépések:

Felhasználói regisztráció és bejelentkezés (POST /api/register, POST /api/login)

Rendelés leadása bejelentkezett felhasználóval (POST /api/orders)

Admin rendelések áttekintése és státuszkezelés (GET /api/admin/orders, PUT /api/admin/orders/{id}/approve|reject)

Kapcsolatfelvételi üzenet küldése (POST /api/messages), **DB-mentés + Mailgun küldés**

Az automatizált (PHPUnit) teszteléshez a kódstruktúra előkészített (önálló metódusok, validáció a kontrollerekben), így később könnyen bővíthető Feature tesztekkel (pl. RefreshDatabase traittel izolált környezetben).

4.8 Reszponzív tervezés és felhasználói visszajelzések kezelése

A frontend kialakítását úgy végeztem, hogy az oldal minden képernyőméreten – mobilon, táblagépen és asztali gépen – könnyen kezelhető legyen. A reszponzivitást Bootstrap 5 osztályokkal alakítottam ki, 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 visszajelzéseket Vue-alapú komponensekkel jelenítem meg. Hibás űrlapkitöltés esetén a mezők alatt hibaüzenetek jelennek meg, sikeres művelet esetén szöveges visszajelzések tájékoztatják a felhasználót.

Az UI-megoldásokat úgy alkalmaztam, hogy segítsék az aktuális művelet értelmezését; a színkódolás és az elrendezések vizuálisan is vissza jelzik a rendszer állapotát.

4.9 Adatstruktúrák optimalizálása és relációk finomítása

Az adatbázis úgy épül fel, hogy minden fő entitás külön táblát kap, a kapcsolatok pedig normalizált módon jelennek meg. Fő táblák:

users – a regisztrált felhasználók adatai

products – a támogatási célok/termékek (név, leírás, kép, célösszeg, eddig gyűlt összeg)

orders – a felhasználók által leadott támogatási igények (rendelések)

messages – a kapcsolatfelvételi űrlapon küldött üzenetek

A normalizálásnak megfelelően az **orders** táblában csak a hivatkozó kulcsok szerepelnek (user_id, product_id) és a státusz, felesleges adatduplikáció nélkül.

4.9.1. Relációk megvalósítása

A relációk egyszerű metódusokkal írhatók le:

```
// 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 például egy felhasználó összes rendelésének, vagy egy rendeléshez tartozó termék adatainak közvetlen lekérdezését.

4.9.2. Eager Loading és teljesítmény

A „**eager loading**” csökkenti az N+1 lekérdezések számát, így gyorsítja a listázásokat (pl. admin nézetben):

```
Order::with(['user', 'product'])->get();  
Szükség esetén mezőszűkítést is alkalmazunk, például:  
User::select('id', 'name', 'email')->get();  
Ez csökkenti az átvitt adatmennyiséget, javítva a válaszidőt.
```

4.9.3. Adattípusok optimalizálása

A migrációkban a táblák típusai a Laravel ajánlásaihoz igazodnak:

Elsődleges kulcsok: id() (big integer, auto-increment)

Rövidebb szövegek: string mezők (pl. name, email)

Hosszabb tartalmak: text (pl. description, message)

Számok: a pénzjellegű értékekhez **egész** mezők kerültek (goal_amount, collected_amount)

Időbelyegek: timestamps() automatikus kezeléssel

4.9.4. Migrációs fájlok struktúrája

Minden tábla külön migrációból épül fel. Példa (részlet) az **orders** tábla létrehozására:

```
Schema::create('orders', function (Blueprint $table) {  
  
    $table->id();  
    $table->foreignId('user_id')->constrained('users')->onDelete('cascade');  
    $table->foreignId('product_id')->constrained('products')->onDelete('cascade');  
    $table->string('status')->default('pending');  
    $table->timestamps();  
});
```

A constrained('...') és az onDelete('cascade') gondoskodik az érvényes kapcsolatról és az árva rekordok elkerüléséről.

4.9.5. Jövőbeli bővíthetőség

Az adatmodell a bővítés lehetőségét szem előtt tartva készült. Lehetséges irányok:

Felhasználói szerepkörök: a users táblában dedikált szerepkör-mező bevezetése (pl. role)

Termékkategóriák: külön categories tábla és products.category_id idegen kulcs felvétele

Üzenetkezelés: a messages táblában státusz mező (pl. „új”, „válaszolt”)

Ezekkel a struktúrákkal a rendszer hosszú távon is karbantartható és jól skálázható marad.

4.10 Felhasználói pszichológia és interaktív élmény tervezése

A felület tervezésekor az egyszerű használhatóságra, átláthatóságra és logikus felépítésre törekedtem. A **Vue 3** komponensrendszerrel készült frontend célja, hogy a látogatók gyorsan megtalálják a keresett funkciókat, és a kezelőfelület minimális tanulási időt igényeljen.

A kulcskomponensek közé tartoznak a **termékkártyák** (ProductCard.vue), a **regisztrációs nézet** (RegisterView.vue) és a **navigációs sáv** (Navbar.vue). Ezek a komponensek jól strukturáltak, a különféle kijelzőmóréteket figyelembe veszik: mobilnézetben a navigáció **hamburger menüre** vált, a termékek **rácsos elrendezésben** jelennek meg, az ürlapelemek pedig kisebb képernyőn **egymás alá** rendeződnek.

A felhasználói élményt támogató vizuális visszajelzések:

Űrlaphibáknál **mező alatti hibaüzenetek** jelennek meg.

A gombok **elhelyezése** és **stílusa** egyértelművé teszi a funkciójukat.

Hover-effektusok segítik a navigációt és az interakciók felismerését.

A projektben az alapvető visszajelzések vizuálisan is érzékelhetők; külön animációk és toast-üzenetek nélkül is egyértelmű a rendszer állapota. A színezés jelenleg az **alap Bootstrap mintákat** követi; explicit, külön definiált színséma vagy WCAG-specifikáció még nincs, ugyanakkor a tipográfia és a kontraszt általánosan jól

használható.

A komponenseket a **felhasználói útvonalakhoz** igazítottam: a fő funkciók (regisztráció, bejelentkezés, termékböngészés, kapcsolatfelvétel) a menüből egyértelműen elérhetők, és minden oldalról néhány kattintással megközelíthetők. A cél egy **vizuálisan visszafogott**, funkcióközpontú felület volt, amely nem terheli túl a felhasználót. A **Vue 3 komponensstruktúra** karbantartható és bővíthető alapot biztosít, így a későbbi UX-fejlesztések könnyen beilleszthetők.

4.11 Működésbiztonság és hibakezelés

A SegítsVelem projektben a működésbiztonság és hibakezelés kiemelt szerepet kap. Az alábbiakban összefoglalom a backend oldali megoldásokat, amelyek a biztonságos adatkezelést, a megfelelő hibakezelést és a védelmi mechanizmusokat biztosítják.

4.11.1. Autentikáció és hozzáférés-kezelés

A felhasználói hitelesítés **Laravel Sanctum**-mal történik **Bearer token** alapon (personal access token), ami jól illeszkedik az SPA architektúrához.

A védett végpontok **auth:sanctum** middleware alatt érhetők el (pl. /api/user, /api/orders, admin útvonalak).

A kliens a bejelentkezés után kapott tokent az **Authorization: Bearer <token>** fejlécvel küldi minden kérésnél.

Nem használunk stateful/cookie-s hitelesítést az API-hoz, így **CSRF-cookie** sem szükséges az API-hívásokhoz.

4.11.2. Adatvalidáció és hibakezelés

Minden adatbevitel szerveroldali validációt megy keresztül. Példa:

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

A validációs hibák **422 (Unprocessable Entity)** JSON választ adnak, amelyet a Vue komponensek mezőnként jeleznek.

Az érzékeny műveletek (pl. e-mail küldés) köré **try–catch** kerül, és hiba esetén **500-as**

választ adunk, a részletek naplózásra kerülnek.

4.11.3. Központi hibakezelés és naplózás

A kivételeket a Laravel **Handler** kezeli, a naplózás a .env alapján történik:

```
LOG_CHANNEL=stack
```

```
LOG_LEVEL=debug
```

A hibák a storage/logs/laravel.log fájlba kerülnek, így visszakövethetők és elemezhetők (különösen éles környezetben hasznos).

4.11.4. CORS beállítások és HTTP védelem

A config/cors.php csak a fejlesztői frontendet engedi (példa):

```
'paths' => ['api/*', 'sanctum/csrf-cookie'],  
'allowed_origins' => ['http://localhost:5173', 'http://127.0.0.1:5173'],  
'supports_credentials' => false, // Bearer token mellett nem szükséges
```

Az API útvonalak az api.php-ban definiáltak; publikus végpont csak indokolt esetben elérhető.

4.11.5. SQL injekció elleni védelem

Az **Eloquent ORM** és a query builder **kötött paraméterezést** használ, ami védi a lekérdezéseket az SQL injection ellen. Példa:

```
$user = User::where('email', $email)->first();
```

A bemenet validálása és a tömeges kitöltés (mass assignment) kontrollja (\$fillable / \$guarded) további védelmet ad.

4.11.6. Session és cookie biztonság

A session-kezelés adatbázisban történik:

```
SESSION_DRIVER=database
```

```
SESSION_LIFETIME=120
```

```
SESSION_ENCRYPT=false
```

A **session a webes réteghez** tartozik; az **API-hívásokhoz Bearer token** használatos, így **CSRF** az API-ban nem szükséges. (Ha később stateful/cookie-s API-ra váltanál, akkor jönne képbe a CSRF-cookie és a stateful middleware.)

4.11.7. E-mail küldés és továbbítás

Az alkalmazás e-mailt **Mailgun** szolgáltatón keresztül küld (HTTP API). Példa beállítás:

MAIL_MAILER=mailgun

MAIL_FROM_ADDRESS=no-reply@<sajat-domainind>

MAIL_FROM_NAME="Segíts Velem"

MAILGUN_DOMAIN=<sajat-domainind>

MAILGUN_SECRET=<mailgun_api_key>

A kapcsolatfelvételi űrlap a **POST /api/messages** végpontot hívja; a backend **adatbázisba ment**, majd **Mailgunon** keresztül levelet küld az üzemeltetői címre. (Fejlesztéskor használható MAIL_MAILER=log, ekkor a levelek a logba kerülnek.)

4.11.8. Felhasználói jogosultságok szétválasztása

Az admin jogosultság **backend oldalon** dől el, a users táblában lévő **is_admin** mező alapján.

Az admin útvonalak (/api/admin/orders, approve/reject) **auth:sanctum + admin** ellenőrzéssel érhetők el.

Frontenden lehet **kényelmi átirányítás** admin belépéskor, de a **valódi védelem** a backend ellenőrzés (nem e-mail egyezés).

4.12 Komponensalapú frontend struktúra Vue 3 keretrendszerben

Én feletem 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 funkcióira 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 igénylelés 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 Bootstrap 5 osztályokat használtam. 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 oszlopból, nagyobb képernyőn több oszlopból 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 kizárolag 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

A komponenseket úgy alakítottam 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. E struktúra kialakításával megalapoztam a projekt további bővíthetőségét, legyen szó új oldalakról, új funkciókról vagy reszponzív átalakításról.

4.13 Teljes oldalkészítési folyamat és komponenslogika

A SegítsVelem frontend oldalait Vue 3 keretrendszer és Bootstrap 5 használatával hoztam létre. A Vue komponensalapú működése lehetővé tette, hogy minden oldal önálló nézetként (view) és logikailag szétválasztott komponensekkel épüljön fel.

Munkafolyamatom a következő lépésekkel állt:

4.13.1. 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.

4.13.2. Szerkezeti felépítés Vue komponensben

A src/views mappában létrehozott új nézettfájlokat, 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.

4.13.3. Komponensek beillesztése

Én minden oldalon újrahasználható Vue komponenseket helyezett el:

A ProductCard.vue a terméklisták megjelenítésére szolgál.

A Navbar.vue a menürendszer 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.

4.13.4. 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 megjelenítette a megfelelő mező alatt egy v-if feltétel, piros szövegként.

4.13.5. 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 teszteltem különböző kijelzőmérétek mellett.

4.13.6. Tesztelés és iteráció

Miután elkészült egy oldal, manuálisan végig próbáltam az összes mezőt, gombot és folyamatot. Ha hibát tapasztaltam, javítottam az adatkezelés vagy megjelenítés logikáját.

4.13.7. Komponenslogika és újrahasznosíthatóság

A projekt során figyelt arra is, hogy minden komponens önállóan is működjön. A ProductCard.vue például props-on keresztül kapja meg a termékkatákat, 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 Adminfelület

Az adminfelület kialakítása a projekt előrehaladtával teljes funkcionálitással és egységes vizuális megjelenéssel valósult meg. A `src/views/AdminView.vue` nézet az **igénylések (orders)** kezelésére fókuszál: az admin áttekintheti a beérkezett kérelmeket, és státuszt állíthat.

AdminView fő elemei (rendeléskezelés):

Rendelések listája – felhasználó és termék adatokkal, áttekinthető kártya/lista nézetben.

Státuszműveletek – Jóváhagyás és Elutasítás gombok megerősítéssel.

Visszajelzések – betöltési állapot, hiba/siker üzenetek, frissített lista.

API-integráció – axios + **Bearer token** (Sanctum); végpontok:

GET /api/admin/orders – listázás

PUT /api/admin/orders/{id}/approve – jóváhagyás

PUT /api/admin/orders/{id}/reject – elutasítás

4.15 Frontend és adminpanel dizájn

Az adminfelület teljes frontend kinézetét, a komponensek vizuális kialakítását és az interaktív funkciókat Vue 3 és **Bootstrap 5** alapokon készítettem el. A nézet modern, jól strukturált, a **rendelések (orders)** áttekintésére és státuszkezelésére fókuszál.

Az adatkötéseket v-model használatával oldottam meg, az állapotokat egy átlátható data() struktúrában kezelem:

```
data() {
  return {
    orders: [],
    loading: false,
    error: null,
    success: null,
    selectedOrderId: null, // épp művelet alatt álló rendelés
    confirmAction: null, // 'approve' | 'reject'
  }
}
```

Az AdminView.vue komponens főbb, általam készített elemei:

Rendeléslista kártyás / táblás megjelenítéssel (Bootstrap rács: row, col-md-*).

Státuszműveletek (Jóváhagyás / Elutasítás) gombokkal, megerősítéssel.

Visszajelzések: sikeres/hiba üzenetek (alert-success, alert-danger), betöltési állapot.

Megerősítő modal Bootstrap alapon a véglegesítés előtt (törles nélküli, státuszváltáshoz).

API-integráció axios-szal, **Bearer token** (Sanctum) fejléc kezeléssel.

A modalt a mounted() ciklusban példányosítom, és ref hivatkozásokkal vezérlek Vue 3-ban, hogy a megerősítés/feloldás konzisztensen működjön.

A felület világos, letisztult színsémát követ: világos háttér, fehér tartalomdobozok, finom árnyék és lekerekített sarkok. Az elemek következetes margó/padding beállításokat kapnak, a műveleti gombok egyértelmű színkódval jelennek meg.

4.16 Backend API és adatkezelés

Az API útvonalai a routes/api.php fájlban vannak definiálva, ezért minden végpont **/api** prefix alatt érhető el (a frontend axios **baseURL** ehhez igazodik: <http://localhost:8000/api>).

4.16.1. Fő erőforrások (jelenleg használt végpontok):

Auth: POST /api/register, POST /api/login, POST /api/logout, GET /api/user

Products: GET /api/products, GET /api/products/{id} (*jelenleg olvasás; create/update/delete végpontok nincsenek kitéve*)

Orders: GET /api/orders, POST /api/orders (*bejelentkezve*)

Contact: POST /api/messages (*publikus kapcsolati űrlap*)

Admin (auth + admin): GET /api/admin/orders, PUT /api/admin/orders/{id}/approve, PUT /api/admin/orders/{id}/reject

A HTTP metódusok (GET/POST/PUT) az adott művelet jellegét tükrözik (listázás, létrehozás, módosítás).

4.16.2. Adatkezelés Eloquent ORM-mel

Az adateléréshez a **Laravel Eloquent ORM**-et használom, így objektumorientált módon, olvashatóan lehet dolgozni az adatokkal.

Példák (a tényleges mezőkhöz igazítva):

```
// Termékek listázása (csak szükséges mezők)  
  
$products  
Product::select('id','name','description','image','goal_amount','collected_amount')->get();  
  
// Egy termék lekérése  
$product = Product::findOrFail($id);  
  
// Rendelés létrehozása (bejelentkezett userrel)  
$order = Order::create([  
    'user_id' => auth()->id(),  
    'product_id' => $request->product_id,  
    'status' => 'pending',  
]);
```

Az **Eloquent** tömeges kitöltés (mass assignment) védelme miatt a modellekben beállítottam a megfelelő \$fillable mezőlistákat.

4.16.3. Felhasználói adatok kezelése

Regisztrációkor új **User** rekord jön létre, a jelszó **bcrypt** hash-sel kerül tárolásra. Bejelentkezés után **personal access token (Sanctum)** készül, amellyel a kliens azonosítja magát. Az aktuális felhasználó adatai a GET /api/user végponton kérhetők le:

```
$user = Auth::user();
```

Ez lehetővé teszi, hogy a frontend megjelenítse a bejelentkezett felhasználó nevét/e-mailjét, és személyhez kötött funkciókat (pl. rendelés leadása) biztosítson.

4.16.4. Adminisztrációs funkciók

Az admin végpontok külön útvonalakon érhetők el, és **auth:sanctum + admin** ellenőrzéssel védettek:

GET /api/admin/orders – összes rendelés listája

PUT /api/admin/orders/{id}/approve – jóváhagyás

PUT /api/admin/orders/{id}/reject – elutasítás

Az admin jogosultság **backend oldalon** dől el a users.is_admin mező alapján (nem e-mail egyezés szerint), így később könnyen bővíthető igény esetén szerepkör-alapú jogosultsággal (RBAC).

4.16.5. Üzenetküldés e-mailben

A kapcsolatfelvételi űrlap a **POST /api/messages** végpontot hívja. A backend:

Validálja a mezőket (name, email, message),

Elmenti az üzenetet a messages táblába,

E-mailt küld az üzemeltetőnek **Mailgun** szolgáltatón keresztül (HTTP API driver).

Fejlesztői környezetben használható a MAIL_MAILER=log, ekkor a levelek a logba kerülnek (nincs tényleges kiküldés).

4.16.6. Biztonságos adatfeldolgozás

Minden adatfogadó végpontnál szerveroldali **validáció** történik:

```
$request->validate([
    'email' => 'required|email',
    'message' => 'required|string'
]);
```

A validációs hibák **422** (Unprocessable Entity) JSON válaszban érkeznek, a frontend mezőnként jeleníti meg őket. A védetű útvonalak ellenőrzését az **auth:sanctum** middleware végzi; a publikus POST /api/messages végpontot throttling védi a túlterheléstől.

4.16.7. Tokenkezelés és authentikáció

Az autentikáció **Sanctum Bearer tokennel** működik (nem cookie-alapú stateful módon). Bejelentkezéskor a backend **token** értéket ad vissza, a kliens ezt a kérésekhez az **Authorization: Bearer <token>** fejlécuccel küldi (axios interceptor). A felhasználó

hitelesített marad, amíg a token érvényes és a kliens elküldi azt; kijelentkezéskor a backend érvényteleníti a felhasználó tokenjeit, a kliens oldalon pedig törlődik a tárolt token.

4.17 Stíluskezelés és vizuális visszajelzések beépítése Vue 3 frontendben

A frontend stíluskezelését és a felhasználói visszajelzések megjelenítését **Vue 3** és **Bootstrap 5** alapokon valósítottam meg. A dizájn és a layout teljes egészében ezekre a technológiákra támaszkodik. minden oldalon figyelembe vettet a célközönséget (szülők, diákok, adminisztrátorok), és ennek megfelelően alakítottam ki egy letisztult, barátságos és strukturált felhasználói felületet. Az egységes megjelenést Bootstrap osztályokkal, szükség esetén **inline** vagy **scoped CSS**-sel biztosítottam.

Vizuális stíluskezelés

- Az elrendezések **Bootstrap rácsszerkezettel** készültek (row, col-md-*, container).
- A szín–tipográfia alapvetően a Bootstrap témáit követi, egységes kontraszttal és olvashatósággal.
- A form elemek (input, label, button) következetesen formázottak, jól olvashatók és kontrasztosak.
- A navigációs sáv minden oldalon látható, **mobilon hamburger menüre vált**.

Az interaktív felületeknél törekedtem arra, hogy minden művelet után egyértelmű visszajelzés jelenjen meg. Külön értesítési rendszert (toast/modal feedback) nem használok, de az alap vizuális visszajelzések minden fontos helyen megjelennek.

Felhasználói visszacsatolás formái

- **Hibás adatbevitel:** ha a backend validáció hibát ad (pl. rövid jelszó, hiányzó mező), a mező alatt **piros hibaüzenet** jelenik meg.
- **Sikeress művelet:** pl. regisztráció után átirányítás történik, és az új nézeten **megerősítő szöveg** látható.
- **Fókusz állapot:** a Bootstrap :focus stílusai segítik a mezők közti navigációt.
- **Hover-effektusok:** a gombok és linkek vizuálisan reagálnak az egérmozgásra.

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

rendszerben, és mikor milyen művelet hajtódott végre.

Logikai szempontok

A visszajelzéseket **Vue-komponenseken** belül kezelem. A form mezők állapotát v-model kötések figyelik valós időben, a backendből érkező válaszokat az **axios** hívások then/catch ágain dolgozom fel. A hibák és üzenetek megjelenítése egyszerű v-if blokkokkal történik közvetlenül az érintett mezők mellett.

Arra is figyeltem, hogy a fő funkciók **minimális navigációval** elérhetők legyenek, így a felhasználók kevés kattintással juthatnak el a célhoz (pl. termék igényléséhez vagy kapcsolatfelvételhez).

4.18 Backend

A backend implementációjánál minden funkciót a valós használati elvárásokhoz igazítottam. **Minden POST kérelem** előtt szerveroldali **validáció** fut, így az érvénytelen adatok nem jutnak be a rendszerbe. A **kapcsolatfelvétel** biztonságos feldolgozással működik: az üzenet először **adatbázisba** mentődik, majd **e-mail** értesítés megy az üzemeltetői címre **Mailgunon** keresztül.

A dokumentációban szereplő példák (pl. **AuthController**, **MessageController**) szemléltetik a vezérlők szerkezetét, a használt **validációs szabályokat** és az adatok kezelését. A Mail integráció célja, hogy a rendszer a felhasználói űrlap alapján **közvetlen értesítést** küldjön az adminoknak—fejlesztői környezetben a küldés **log**-ba irányítható, élesben **Mailgun** végzi a továbbítást.

A **hitelesítés** a **Laravel Sanctum** megoldásával történik, **Bearer token** alapon (personal access token). Ez jól illeszkedik a **Vue 3 + Vite** SPA-hoz. A védett útvonalakat az **auth:sanctum** middleware korlátozza (pl. rendelés leadása, saját profil-lekérés, admin műveletek). **Stateful/cookie-s** módot **nem** használok az API-hoz, ezért **CSRF-cookie** és **SANCTUM_STATEFUL_DOMAINS** beállítás **nem szükséges** az API-hoz.

Az adatbázis-struktúrát **migrációkkal** állítottam össze; a táblák (**users**, **products**, **orders**, **messages**) közötti relációk a működést tükrözik. Az **orders** minden igénylést naplóz, a felhasználó így követheti a múltbeli kéréseit, az **admin felület** pedig

moderálhatja a státuszokat (jóváhagyás/elutasítás).

A **hibák azonosítását** a Laravel beépített hibakezelője támogatja; fejlesztéskor APP_DEBUG=true mellett részletes stack trace segíti a konfigurációs/validációs/útvonal-hibák feltárását. A .env célzottan az **API-alapú** működéshez van beállítva (CORS a frontend originre, naplázás stack csatornával, Mailgun kulcsok, adatbázis).

Összegzés:

A backend stabil, biztonságos és az SPA-val zökkenőmentesen integrált. A tokenes hitelesítés, a szigorú validáció, a jól definiált útvonalak és az egyértelmű adatmodell együtt biztosítják a megbízható működést.

4.19 Interaktív funkciók és felhasználói bevonás

A SegítsVelem frontend oldalainak **interaktív viselkedésén** dolgoztam. Nemcsak a felületeket készítettem el, hanem azokat az elemeket is, amelyek közvetlenül a felhasználói műveletekre reagálnak. A rendszer teljes mértékben **Vue 3** és **Bootstrap 5** alapokra épül, így az interakciók megvalósítása is ezekre támaszkodik.

Kiemelten figyeltem arra, hogy minden fontos művelethez **egyértelmű visszajelzés** tartozzon — a felhasználó mindenkor lássa, hogy sikeresen járt-e, hibát követett el, vagy **mi a következő lépés**.

A legfontosabb interaktív funkciók:

– Termékkártyák interaktív működése

A ProductCard.vue komponens jeleníti meg az egyes termékeket. Az „Igényelem” gombra kattintva **backend-hívás** indul; a művelet után a gomb/állapot **vizuálisan frissül**, hogy egyértelmű legyen a visszajelzés.

– Űrlapvalidáció és visszajelzés

A RegisterView.vue, LoginView.vue és ConnectionsView.vue űrlapjai v-model + v-if alapú hibakezelést használnak. Backend-validációs hiba esetén a mező alatt **piros hibaüzenet** jelenik meg.

– Felhasználói visszajelzés műveletek után

Sikeressé regisztráció/bejelentkezés után **átirányítás** történik, és az új nézeten

megerősítő tartalom látható. Külön toast/modal rendszer nélkül is biztosított az alap vizuális visszajelzés.

– Navigáció állapotkezelés

A Navbar.vue mobilon **hamburger menüre** vált. Az **aktív útvonal** kiemelést kap, így a felhasználó minden tudja, hol jár.

– Hover-effektek és interakciók

A termékkártyák és fő gombok **hoverre** vizuálisan reagálnak (árnyék/szín). Ezt Bootstrap osztályokkal és egyszerű **scoped CSS**-sel oldottam meg.

UX- és hozzáférhetőségi szempontok

Arra törekedtem, hogy a fő funkciók **kevés kattintással** elérhetők legyenek, és a felhasználó **sose maradjon visszajelzés nélkül**. Bár dedikált akadálymentesítési kiegészítők (pl. aria-label, egyedi tabindex) jelenleg nincsenek, a Bootstrap-alapú markup és gombkezelés **részben** kiszolgálja a billentyűzetes navigációt és a képernyőolvasókat.

A Vue komponenseket úgy terveztem, hogy **önállóan is működjene**; minden interakcióhoz tartozó adatmozgás (igénylés, űrlapküldés) **Axios**-szal történik, oldal-újratöltés nélkül.

Összegzés

A cél egy **interaktív, dinamikusan reagáló** webalkalmazás volt. Az elkészített komponensek és nézetek nemcsak esztétikusak, hanem **funkcionálisan is azonnal visszajeleznek** a felhasználói műveletekre, így a rendszer modern, reszponzív és élményszerű marad minden felhasználói szinten.

5. Továbbfejlesztési lehetőségek

A rendszer alapverziója teljes mértékben működőképes, és Vue 3 + 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 téma választással, 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:

Mobilalkalmazás

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

Többnyelvűség

A Laravel lokalizációs rendszere támogatja a többnyelvű működést, így az oldal később angol, német vagy roma nyelven is elérhetővé válhat.

Push értesítések

Mobil vagy böngésző alapú értesítések bevezetésével a felhasználók valós időben kaphatnának információt új termékekről vagy igénylés státuszváltozásokról.

Gamifikációs elemek

Jelvények, ranglisták vagy statisztikák ösztönözhethetnek az aktivitást, különösen az adományozók és önkéntesek körében.

Térképes keresés

Ruhák lakhely vagy körzet szerint is böngészhetők lennének, és lehetőség nyílna gyűjtőpontok kezelésére is.

PDF generálás

Igénylésekhez vagy átvételekhez kapcsolódó igazolások, regisztrációs nyugták PDF formátumban letölthetők lennének.

Admin szerepkörök bővítése

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 szereztem a csapatunka megszervezésében, a modern webfejlesztési technológiák alkalmazásában, valamint a REST API szemléletű backend és a Vue 3 frontend integrációjában. A fejlesztés során számos kihívásba ütköztem – különösen a

jogosultságkezelés, az adatkezelés és a komponensalapú felépítés terén –, de ezek mindenhol hozzájárultak szakmai fejlődésemhez.

A projektet egy társadalmilag hasznos cél szolgálatába állítottam: 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 megtanultam:

Hogyan alkalmazzam a Gitet és a GitHubot verziókövetésre és átláthatóságra,

Hogyan tervezünk adatbázismodellt és API-struktúrát Laravel keretrendszerben,

Hogyan építünk Vue 3 alapú felhasználói felületeket,

Hogyan kezeljük a jogosultságokat és validáljuk a bemeneti adatokat biztonságosan.

Büszke vagyok arra, hogy a projektet a tervezett irányban meg tudtam valósítani, és úgy érzem, hogy nemcsak technikai tudásban, hanem közösen végzett fejlesztési gyakorlatban is sokat fejlődtem.

6. A rendszer társadalmi haszna és fenntarthatóság

A SegítsVelem rendszer nem csupán technikai kihívás volt számomra, hanem tudatos társadalmi szerepvállalás is. Generációinkat 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 gondolom, inkább arról van szó, hogy engem más fórumokon és talán más megközelítéssel lehet elérni, mint a szüleink generációját. Kiemelten fontosnak tartom a környezetvédelmet, a környezettudatos életmódot szemben a fogyasztói társadalom vásárlási kényszerével, az újra felhasználás sokféle lehetőségének támogatását.

Projektünk tudatos céljai:

****Környezeti fenntarthatóság**:** a jó állapotú használt felszerelések és ruhák esetében egyértelmű, hogy mind a hulladék mennyiségeinek, 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.

****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.

****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.

****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 iranya lehet a felhasználói visszajelzések rendszerszintű gyűjtése és az ezekből származó statisztikai elemzések automatizálása. A visszajelzések alapján jobban megérhetjük, hogyan használják az emberek a rendszert, mely funkciókat tartják hasznosnak, és hol tapasztalnak nehézséget.

****Lehetséges visszajelzási funkciók:****

Visszajelzés az admin által nyújtott kommunikációról (segítőkészseg, gyorsaság)

Szöveges megjegyzések beküldése profiloldalról

Esetleges hibák jelentése a felhasználói felületen keresztül

****Statisztikai elemzések:****

Legnepszerűbb terméktípusok, márkok, 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

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>

10. Ábrajegyzék

1.	ábra – kezdőoldal PC	6
2.	ábra – kezdőoldal mobil	6
3.	ábra – fejléc	7
4.	ábra – Kezdőlap (HomeView)	11
5.	ábra – Támogatható célok oldal (ProductsView)	11
6.	ábra – Kapcsolat oldal (ConnectionsView)	12
7.	ábra – Profil oldal (ProfileView)	13
8.	ábra – Regisztráció és bejelentkezés (RegisterView / LoginView)	13
9.	ábra – Admin felület (AdminView)	14
10.	ábra – Backend Telepítése	20
11.	ábra – Frontend Telepítése	21
12.	ábra – ER-diagram	22
13.	ábra – api.php kódja	25
14.	ábra – app.php kódja	33
15.	ábra – axios hívás	38