



győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

Záródolgozat feladatkiírás

Tanuló(k) neve¹: Dezamics Bence, Joó Barnabás, Nyikos Kata
Képzés: nappali munkarend
Szak: 5 0613 12 03 Szoftverfejlesztő és tesztelő technikus

A záródolgozat címe:

„VetCare Connect” állatorvosi időpontfoglaló rendszer

Konzulens: Bólya Gábor

Beadási határidő: 2024. 04. 15.

Győr, 2024. 04. 15.

Módos Gábor
igazgató

¹ Szakmajegyzékes záródolgozat esetében több szerzője is lehet a dokumentumnak, OKJ-s záródolgozatnál egyetlen személy ad le záródolgozatot.



győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

Konzultációs lap²

	A konzultáció		Konzulens aláírása
	ideje	témája	
1.	2023.10.15.	Témaválasztás és specifikáció	
2.	2024.03.01.	Záródolgozat készültségi fokának értékelése	
3.	2024.04.05.	Dokumentáció véglegesítése	

Tulajdonosi nyilatkozat

Ez a dolgozat a saját munkánk eredménye. Dolgozatunk azon részeit, melyeket más szerzők munkájából vettünk át, egyértelműen megjelöltük.

Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul vesszük, hogy a szakmai vizsgabizottság a szakmai vizsgáról kizár minket és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2024. április 15.

Dezamics Bence

Joó Barnabás

Nyikos Kata

² Szakmajegyzékes, csoportos konzultációs lap

Vet Care

c o n n e c t

Készítette:

Joó Baranbás, Dezamics Bence, Nyikos Kata

Tartalomjegyzék

1.	Bevezetés.....	6
1.1	Probléma és megoldás	6
1.2	Kezdet.....	6
1.2.1	Technológiák	6
1.2.2	Mérőföldkövek	7
1.3	Jövőbeni tervek	7
2.	Csapatmunka megvalósítása	8
2.1	Feladatok felosztása	8
2.2	Git és GitHub	8
3.	Adatbázis	9
3.1	Felhasználó típusú táblák	9
3.1.1	Vet tábla	9
3.1.2	Owner tábla.....	10
3.1.3	Admin tábla	10
3.2	Nyitvatartás táblák	10
3.2.1	Opening tábla	10
3.3	Kezelésekhez kapcsolódó táblák	10
3.3.1	Cure tábla	11
3.3.2	Pet tábla	11
3.3.3	Gyik tábla.....	11
4.	Frontend.....	12
4.1	Design.....	12
4.2	Technológia	13
4.2.1	Vue.js	13
4.2.2	PrimeVue	13
4.3	Nyilvános oldalak.....	13
4.3.1	Kezdőlap	13
4.3.2	GYIK	14
4.3.3	Állatorvosok.....	14
4.3.4	Regisztráció.....	15
4.3.5	Bejelentkezés.....	16
4.3.6	Admin bejelentkezés	16
4.4	Gazda felhasználói oldalak	17
4.4.1	Időpontfoglalás.....	17

4.4.2	Naptáram.....	18
4.4.3	Kedvenceim	18
4.5	Orvos felhasználói oldalak.....	20
4.5.1	Naptáram.....	20
4.5.2	Nyitvatartás	20
4.6	Adatmódosítás	21
4.7	Admin oldal	22
4.8	Reszponzivitás	23
4.9	Frontend tesztelés.....	24
4.9.1	Selenium	24
4.9.2	Tesztek.....	24
5.	Backend	26
5.1	Technológia	26
5.2	Controllerek.....	26
5.2.1	AdminController	26
5.2.2	AuthController.....	27
5.2.3	MainController	29
5.2.4	OwnerController.....	30
5.2.5	PasswordController	31
5.2.6	VetController	32
5.2.7	MailController	34
5.3	Emailek	34
5.3.1	Regisztráció.....	34
5.3.2	Jelszó visszaállítása	35
5.3.3	Időpontfoglalás, lemondás	35
5.3.4	Időzített értesítés	35
5.4	Emailek felépítése	35
5.5	Autentikáció	36
5.6	Jelszótitkosítás.....	37
6.	Fejlesztői futtatási kézikönyv	38
7.	Publikálás.....	39
8.	Források.....	40

1. Bevezetés

A VetCare Connect egy állatorvosok és gazdák közötti rendszer.

Fő funkciói az időpontfoglalás és a felhasználók állatainak nyilvántartása, a kezelések időpontjainak számontartása, email-es emlékeztetők küldése.

1.1 Probléma és megoldás

Jelenleg Magyarországon nem létezik olyan oldal, amelyen keresztül könnyen válogathatunk az állatorvosok között, és online tudunk hozzájuk időpontot foglalni. A VetCare Connect ebben is nyújt segítséget a gazdáknak. Emellett az orvosok számára is hasznos funkciókat tartogat. Amellett, hogy a gazdák könnyebben rátalálnak egy orvosra (tehát az oldal egy „reklámfelületként” is szolgál), az orvosok időpontjainak nyilvántartásában is segít a rendszer. Egy helyen, gyorsan és egyszerűen tudja megtekinteni, hogy melyik nap, milyen kezelésekre foglaltak időpontot hozzá.

1.2 Kezdet

Akkor fogalmazódott meg bennünk az ötlet, hogy egy ilyen alkalmazást készítsünk, amikor egyikünknek állatorvoshoz kellett vinnie a kutyáját. A bonyolult időpontkérés, hosszas várakozás váltotta ki az ötletet. Az alapkoncepció után mindenkinek rengeteg elképzelése volt, hogy hogyan, milyen funkciókkal színesíthetnénk az oldalunkat.

1.2.1 Technológiák

Az alap ötlet megfogalmazása után el kellett döntenünk, hogy milyen technológiákkal szeretnénk dolgozni a munka során. Ebben egységesen egyet értettünk, hiszen közös erősségeink vannak. Fontos volt, hogy lendületesen tudjunk dolgozni, és olyan technológiákat válasszunk, amelyekbe tényleg szeretnénk beletanulni a munka során. Így esett a választásunk a következő technológiákra:

Design tervezet	Figma
Adatbázis	MySQL
Kód szerkesztő	Visual Studio Code
Frontend technológia	Vue.js JavaScripttel
Backend technológia	Laravel, PHP
Verziókövetés	Git, GitHub

1.2.2 Mérföldkövek

Már az első megbeszélés után tudtuk, hogy szeretnénk a haladás érdekében egyes nagyobb feladatoknak határidőt kitűzni. Ezekhez általában tudtuk tartani magunkat, nagyobb csúszás sosem volt.

Téma megfogalmazása	2023.09.30
Ötletek kidolgozása	2023.10.15.
Adatbázis felépítése	2023.11.10.
Látványterv elkészítése	2023.12.01.
Frontend elkészülése	2024.03.15.
Backend elkészülése	2024.03.15.
Tesztelés, hibák kijavítása	2024.04.01.
Vizsgaremek leadása	2024.04.15.

1.3 Jövőbeni tervek

Rengeteg ötletünk volt a projektünkkel kapcsolatban végig a munka során. Ahogy elkezdtük az oldalt fejleszteni, mindig feljöttek ötletek, funkciók, amelyek elengedhetetlenek voltak az oldal megfelelő, felhasználóbarát működéséhez. Azonban a fejlesztési idő végének közeledtével is jutottak eszünkbe funkciók, amelyekkel a későbbiekben tudjuk fejleszteni az oldalt:

Orvosi időpontlemondás: Az orvos is tudja lemondani az időpontokat, ne csak a felhasználó. Erről a gazdák kapjanak egy értesítő emailt.

- Orvosi adatlap: Az Állatorvosok menüpontban (lásd: [4.3.3](#)) a felhasználó tud keresni az orvosok között. Ide kerülhetne egy gomb, amely az orvosok adatlapjára vezetne. Itt láthatná a gazda az orvos nyitvatartási idejét, bemutatkozását, specializálódását.
- Orvosi specializálódás: A való életben vannak olyan állatorvosok, akik nem foglalkoznak minden állat típussal. Az orvosoknak meg lehetne adniuk az állatfajokat, amelyekre specializálódtak, és csak ilyen állatnak lehetne foglalni hozzá.
- Állat profilkép: A gazdák tudjanak egyedi profilképet beállítani az állataiknak.

A fejlesztés közben már elgondolkoztunk azon, hogy a való életben mennyire lehetne használni az állatorvoslásban a weboldalunkat. Arra jutottunk, hogy hasznos oldal, így a jövőben szeretnénk megkeresni olyan állatorvosokat, akik hajlandóak lennének használni az oldalunkat. Ajánlások és direkt megkeresések útján tudna terjedni alkalmazásunk híre. Bízunk benne, hogy néhány apróbb fejlesztés után, megtalálja oldalunk a piacot, és lesz rá kereslet.

2. Csapatmunka megvalósítása

2.1 Feladatok felosztása

A csapatmunkát a feladatok felosztásával kezdtük. Alapelvünk az volt, hogy minimális szinten mindenki szeretne minden munkafolyamatba belemenni.

A frontendet Joó Barnabás és Nyikos Kata, míg a backendet Dezamics Bence valósította meg. Ennek ellenére a frontendek is írtak API hívásokat, sőt Barnabás az email rendszer kialakításában vett részt aktívan. Kata főleg frontenden tevékenykedett, de kisebb API hívásokban és az adatbázis megtervezésében segített. Bence backendes feladatai mellett kisebb-nagyobb munkákat végzett a bejelentkezés és az állatok oldal frontendes megjelenítésével is.

2.2 Git és GitHub

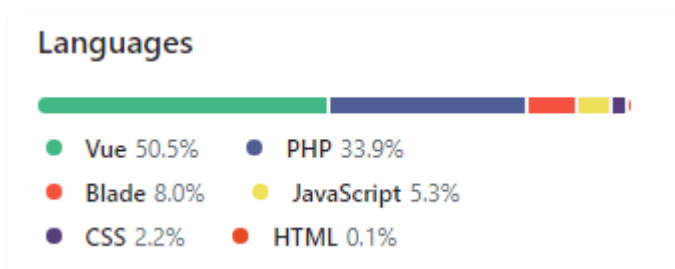
A közös munkához a GitHub webes felületét alkalmaztuk. A GitHub egy nyílt forráskódú szoftver, amely arra szolgál, hogy egy programozó nyomon követhesse a változásokat egy elosztott verziókezelő rendszerben. Git segítségével szoftverfejlesztési verziókövetés-szolgáltatást nyújt.

A Git a változásokat egy elosztott verziókezelő rendszeren keresztül követi. Követni tudja a projekt különböző verzióinak állapotát, amíg fejlesztjük őket. Elosztott, mert hozzáférhetünk a kód fájljaihoz egy másik számítógépről - és ugyanígy más fejlesztők is hozzáférhetnek.

Választásunk azért esett ezekre, mivel nagyon elterjedt, így nem is volt kérdés, hogy ezt használjuk-e. Elterjedtsége mellett rendkívül praktikus, hiszen iskolai körülmények miatt sok tanteremből, sok gépről kellett dolgoznunk projektünkön. Könnyen elérhettük a programot, gyorsan frissíthettük a legújabb verzióra. Amikor gond adódott, egyszerűen tudtuk a kód előző működő verzióit használni a commitoknak köszönhetően.

A GitHub ezek mellett segítséget nyújtott a feladatok nyomon követésében is. Mindenki törekedett tömör, kifejező commit leírásokat adni, így amikor külön dolgoztunk is megtudtuk nézni, hogy a többiek mivel foglalkoztak.

A projektünk GitHub repository-ja elérhető a <https://github.com/dezbence/14AA-C-VetCareConnect> linken.

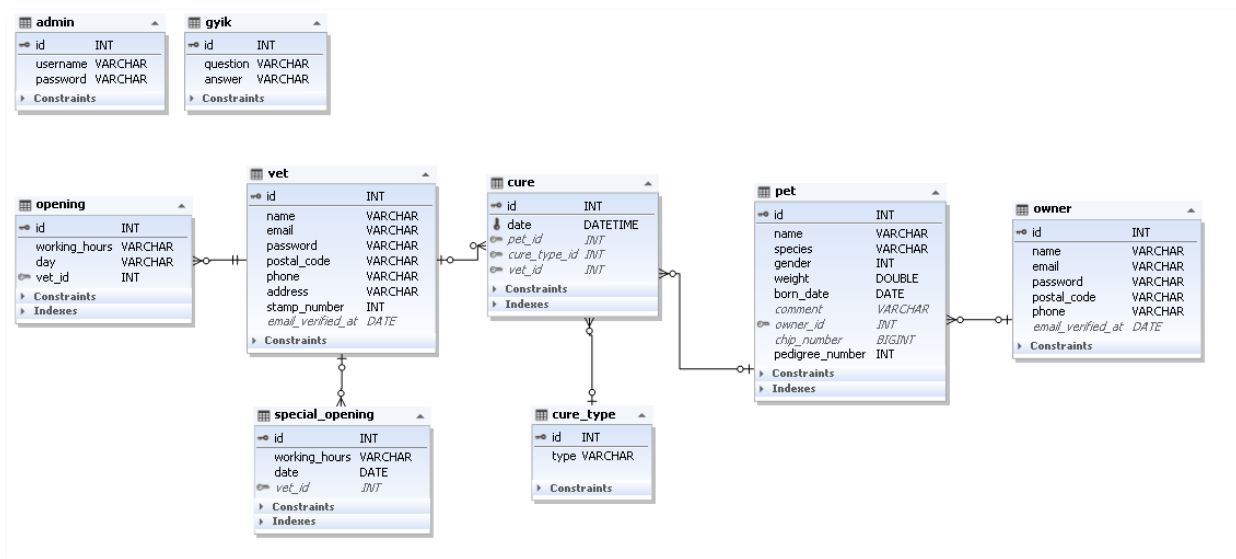


A GitHub mutatja a technológiák használatának eloszlását is

3. Adatbázis

Adatbázis kezelő rendszernek MySQL-t választottunk. A MySQL egy SQL alapú többfelhasználós, relációs adatbázis kezelő rendszer. Az adatbázis szerkesztéséhez a phpMyAdmin-t és a DB Forge Studiot hívtuk segítségül.

Azért döntöttünk a MySQL használata mellett, mivel a technológiában már mindannyian jártasak vagyunk és nem valószínű, hogy egyhamar kimegy a divatból, így programunk korszerű tud maradni.



3.1 Felhasználó típusú táblák

Az adatbázisunkban összesen 3 féle felhasználó típus létezik. Ezek tárolása külön-külön táblában történik. Ennek oka az, hogy más adatokat tárolunk el egy-egy felhasználó típusnál.

Az admin táblában csak egy egyszerű belépési adatokat tárolunk.

A vet és owner táblákban a belépési adatokon kívül különböző kapcsolattartási adatok is megtalálhatók.

3.1.1 Vet tábla

Ebben a táblában az állatorvosok adatai találhatók.

- id: egyedi azonosító, elsődleges kulcs
- name: az állatorvos neve
- email: email cím
- password: hashelt jelszó
- postal code: irányítószám
- address: cím
- phone: telefonszám

- stamp number: orvosi pecsétszám
- email_verified_at: email megerősítésének időpontja

3.1.2 Owner tábla

Ebben a táblában a gazdák adatait tároljuk.

- id: egyedi azonosító, elsődleges kulcs
- name: gazda neve
- email: email cím
- password: hashelt jelszó
- postal_code: irányítószám
- phone: telefonszám
- email_verified_at: email megerősítésének időpontja

3.1.3 Admin tábla

Ebben a táblában az adminok bejelentkezési adatai találhatók

- id: egyedi azonosító, elsődleges kulcs
- username:
- password: hashelt jelszó

3.2 Nyitvatartás táblák

Kétféle nyitvatartás létezik az adatbázisunkban. A special_opening táblában adott dátumra vonatkozóan tárolunk nyitvatartásokat, míg az opening táblában egy általános nyitvatartást mentünk, amely hetente ismétlődik.

3.2.1 Opening tábla

- id: egyedi azonosító, elsődleges kulcs
- working_hours: a nyitvatartás órái
- day: nap
- vet_id: az orvos egyedi azonosítója, akihez a nyitvatartás tartozik, idegenkulcs
- Special_opening tábla
- id: egyedi azonosító, elsődleges kulcs
- working_hours: a nyitvatartás órái
- date: dátum, amelyre a nyitvatartás vonatkozik
- vet_id: az orvos egyedi azonosítója, akihez a nyitvatartás tartozik, idegenkulcs

3.3 Kezelésekhez kapcsolódó táblák

A cure táblában tároljuk az egyes kezeléseket, időpontokat. A cure_types táblában pedig a lehetséges időpont típusok találhatók.

3.3.1 Cure tábla

- id: egyedi azonosító, elsődleges kulcs
- date: a kezelés dátuma és időpontja (pl: 2024-05-05 10:00)
- pet_id: a kezelendő állat egyedi azonosítója, idegenkulcs
- cure_type_id: a kezelés típusának egyedi azonosítója, idegenkulcs
- vet_id: a kezelő állatorvos egyedi azonosítója, idegenkulcs
- Cure_type tábla
- id: egyedi azonosító, elsődleges kulcs
- type: kezelés típusa

3.3.2 Pet tábla

A tárolt állatok adatai találhatók benne.

- id: egyedi azonosító, elsődleges kulcs
- name: az állat neve
- species: az állat fajtája
- gender: az állat ivara
- weight: az állat súlya
- born_date: az állat születési időpontja
- comment: megjegyzések (opcionális mező, pl: állat allergiái, ismertetőjegyei)
- owner_id: az állat gazdájának egyedi azonosítója
- chip_number: chip szám
- pedigree_number: törzskönyv szám

3.3.3 Gyik tábla

A gyakori kérdések adatait innen hívjuk le.

- id: egyedi azonosító, elsődleges kulcs
- question: kérdés
- answer: válasz

4. Frontend

4.1 Design

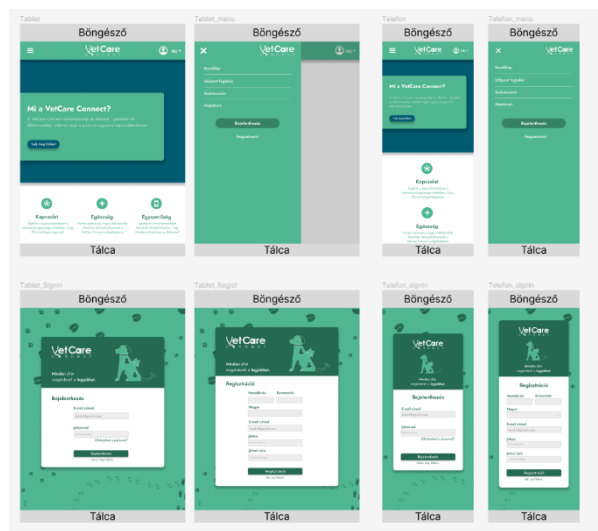
Az egységes megjelenés érdekében, egy közös minta alapján építettük fel alkalmazásunk megjelenését. Ez rendkívül fontos nekünk, de legfőképp a felhasználónak. Egy stílusos, egységes, de leginkább egyszerűen kezelhető felület nagyban hozzájárul a VetCare Connect-ről alkotott véleményben. A külső vonzó, az egység a könnyebb megértést és az egyszerű kezelhetőség pedig a gyors munkát segíti elő.

A közös minta egy Figma dokumentum volt. A Figma egy webalapú tervezői platform, amely lehetővé teszi számodra, hogy weboldal-, mobilalkalmazás-, vagy bármilyen digitális felülettervet hozh létre egyszerűen és hatékonyan. Sőt, nem csupán a kész felületek elkészítésére alkalmas, hasznos társ a kezdeti ötleteléskor és drótváz, sematikus képernyők létrehozásakor is.

Azért választottuk a Figma-t, mert már ketten is használtuk, ami könnyítette a folyamatokat, nem kellett megtanulni más program használatát. Másik előnye, hogy létre lehet hozni projekteket, amelyben lévő fájlokat a meghívott kollaborátorok egyszerre, akár egyidejűleg tudják módosítani. Ezen felül az alapsomag teljesen ingyenes.

Itt először is megállapodtunk a színekben, betűstílusokban és alakzatokban, amiket az oldalon használni fogunk. Ezek képzik a megjelenés alapját. Ezekből állítottunk össze egységes gombokat, kártyákat, beviteli mezőket, űrlapokat.

Miután lefektettük az alapokat elkezdtük a tervezést, összeállítást. Ez a folyamat rengeteg időt spórolt mindannyiunknak. A tervezés lényege, hogy hamar kiderüljenek az esetleges hibák, nehézségek, és ezeket ne utólag kelljen kijavítani. A Figma erre tökéletes, egyszerűen csak odahúzzunk gombokat, alakzatokat, szövegeket, ahova szeretnénk. Sokkal gyorsabb, mintha ugyan ezt a tervezési fázis nélkül, kódolva kellett volna véghez vinni. A közös mintánkat akár egyidejűleg mindhárman tudtuk szerkeszteni, ami ugyancsak gyorsította a folyamatot.



Figma tervek

4.2 Technológia

4.2.1 Vue.js

Frontend technológiának a Vue.js 3 keretrendszert választottuk. Választásunk azért esett erre, mert a Laravel és a Vue.js remek párost alkotnak. A Vue egy könnyen megtanulható, és használható JavaScript könyvtár, amelyet rengeteg feladatra könnyedén fel lehet használni.

A Vue.js komponensekből épül fel. Ennek kapcsán könnyedén hozhatunk létre „újrahasznosítható” elemeket az alkalmazásunkon belül. Például kártyákat, adatok módosítására létrehozott komponenseket többször használtuk weboldalunkon.

Vue segítségével képesek lehetünk akár többoldalas alkalmazásokat is építeni, ebből adódóan a routing is jól megoldott.

A Vue funkciói közül használtuk többek között a reaktív változókat, direktívákat (v-for, v-if, v-else-if, v-else, v-bind, v-model, stb.), szöveg interpolációkat ({{ változó/script }}), valamint az életciklus hurkokat (lifecycle hooks: onMounted, onBeforeMount, stb.). Ezek segítségével könnyebben kezelhetővé vált a kódunk.

A Vue.js mellé rengeteg kiváló paketget, és third party plugint készítettek. Mi a PrimeVue-t választottuk.

Composition API-t használunk, hiszen ez sokkal modernebb, egyszerűbb és szabadabb, mint az Option API, ami egy OOP szemléletű API.

4.2.2 PrimeVue

A PrimeVue egy UI komponens könyvtár. Könnyen, sokszínűen testreszabható komponenseket tartalmaz, melyekkel egyedi módon színesíthetjük a webdesign-unkat. A komponenseken kívül találhatóak benne még ikonok, gombok, beviteli mezők, templatek.

Használtuk a naptár komponenseket, melyeket az időpontfoglalásnál, időpont áttekintésnél használtunk. A beviteli mezők tudják validálni a szövegbevitelt (hány karakter, milyen formátum, milyen típusú karakter), így ez nagyban segítette a munkánkat. Visszatérő elem a weboldalunkon a PrimeVue-s toast értesítés. Ezt használtuk végig visszajelzések, értesítések közvetítésére a felhasználó felé.

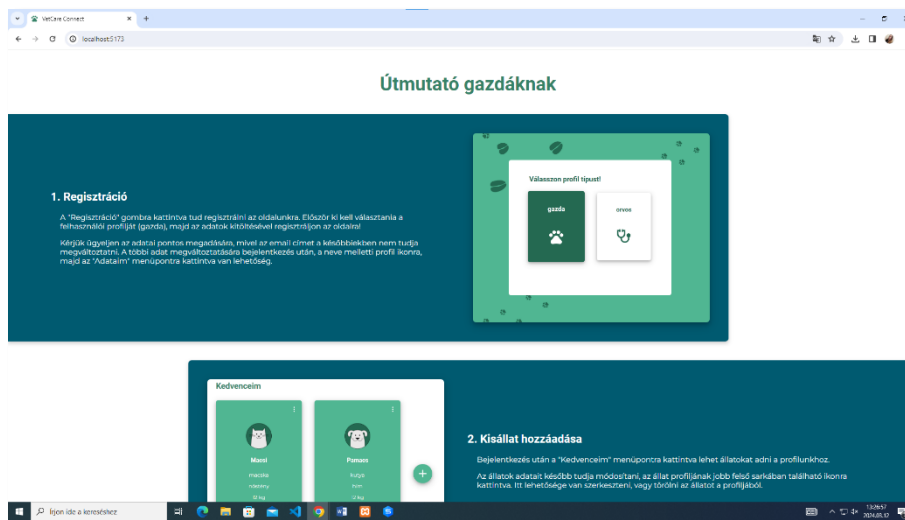
A PrimeVue-nak köszönhetően gyorsan tudtuk fejleszteni a frontendet, hiszen sok komponens csak módosítani kellett, nem kellett teljesen felépíteni. Ennek ellenére egyedi komponenseket tudtunk alkotni, nem egy „tömeg” oldal született.

4.3 Nyilvános oldalak

4.3.1 Kezdőlap

A kezdőlap bemutatja azt, hogy mi is az a VetCare Connect, valamint elérhető rajta egy-egy útmutató gazdák, és orvosok számára is. Ezt igyekeztünk látványosan megoldani, animációk és design elemek segítségével. Az útmutatók a típusnak megfelelő gombok segítségével érhetőek el.

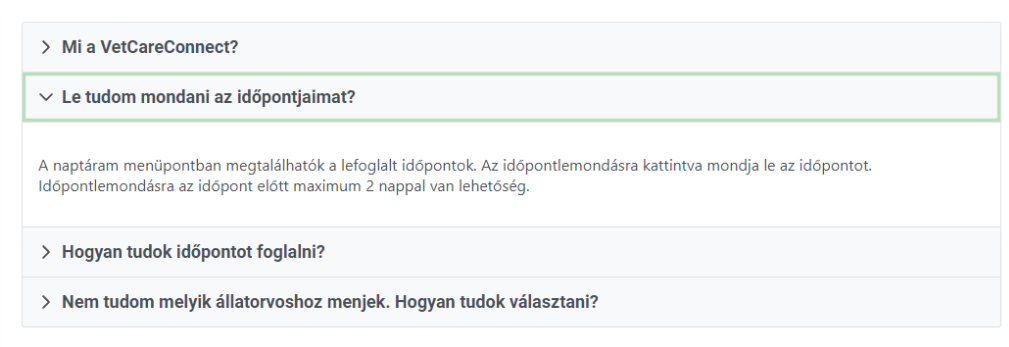
Az útmutatókban szerepelnek képek, annak érdekében, hogy könnyen értelmezhető legyen a mondanivaló. Itt ismertetjük az oldal funkcióit a felhasználókkal.



Útmutató a főoldalon

4.3.2 GYIK

A gyakori kérdések menüpont a /gyik route-ra vezeti a felhasználót. Itt egy primevue komponens segítségével oldottuk meg a kérdések és a válaszok megjelenítését. A kérdéseket adatbázisból hívjuk le, majd jelenítjük meg az oldalon.



Gyakori kérdések

4.3.3 Állatorvosok


Az Állatorvosok menüpont egy állatorvos keresőhöz vezet a /allatorvosok route-on keresztül. Itt található egy szűrő, melynek segítségével név, irányítószám és település alapján kereshetünk az állatorvosok között. Az állatorvosok neve mellett található „Időpontot foglallok” gombra kattintva, egyből az adott orvoshoz foglalhatunk időpontot. Ez a funkció csak akkor érhető el, ha a felhasználó be van jelentkezve.



Állatorvos böngésző, szűrés után

4.3.4 Regisztráció

A főoldalon a jobb felső sarokba kattintva a „Regisztráció” gombra kattintva tud regisztrálni mindkét típusú felhasználó. Ezzel a /regisztracio route-ra irányul az oldal. Itt először ki kell választani a felhasználói fiók típusát.



Fióktípus választása

A tovább gombra kattintva érhető el a regisztráció. Itt meg kell adnia a felhasználónak az adatait. Minden mező validálva van:

- A névben nem szerepelhetnek speciális karakterek (pl. @ vagy !, stb.)
- A telefonszám és az irányítószám mezők a primevue segítségével készültek. Input mask-ot használunk, így ezen mezőkbe csak számok kerülhetnek, a megfelelő formátumban
- Az email cím mezőnél ellenőrizzük, hogy megfelelő formátumban adja-e meg a felhasználó az email címét (pl.: bodri@gmail.com)
- A jelszónak meg kell felelnie az alábbi feltételeknek: 8-64 karakter, minimum 1 nagybetű, minimum 1 kisbetű és minimum 1 szám
- A jelszót meg kell erősíteni, annak érdekében, hogy ne történhessen elírás abban
- Valamint el kell fogadni a felhasználói feltételeket

Amennyiben a felhasználó nem tölt ki minden mezőt, vagy nem fogadja el a felhasználói feltételeket, akkor egy primevue-s komponens, a toast segítségével jelenítünk meg hibaüzenetet.

A regisztráció gomb megnyomása után a felhasználó kap egy emailt, melyben meg kell erősítenie a regisztrációját. Csak ezek után tud bejelentkezni az oldalra.

Regisztráció



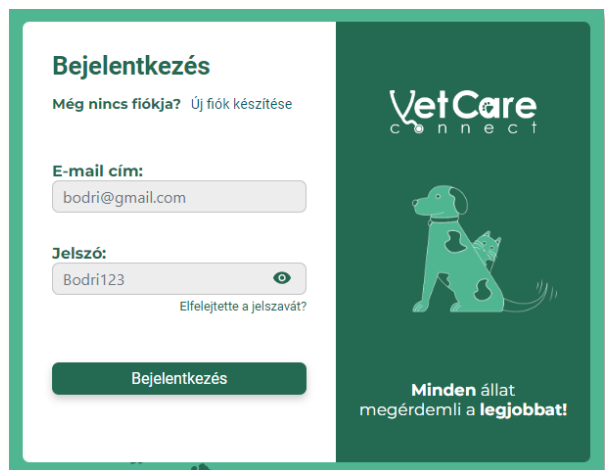
4.3.5 Bejelentkezés

A főoldalon a jobb felső sarokban a Bejelentkezés gombra kattintva van lehetősége a felhasználóknak bejelentkezni. Ekkor az oldal a /bejelentkezés végpontot tölti be.

Bejelentkezéskor a felhasználó email címét és a jelszavát kell megadnia. A felhasználó megtekintheti a beírt jelszavát a szem ikonra kattintva, majd le is rejtheti azt.

Amennyiben sikeresen adta meg a bejelentkezési adatait, akkor a Bejelentkezés gombra kattintva az oldal a főoldalra irányítja a felhasználót. Ekkor az oldalon megjelennek különböző

menüpontok, amelyeket csak bejelentkezés után lehet használni.



Bejelentkezés

Ha a felhasználó még nem erősítette meg az email címét, akkor egy komponens jelenik meg, emlékezteti a felhasználót, hogy erősítse meg az email fiókját. Amikor megerősíti azt, akkor bekerül az adatbázis „email_verified_at” mezőjébe a megerősítés dátuma.

4.3.6 Admin bejelentkezés

Adminként a bejelentkezés a /bejelentkezés/admin route-on elérhető. Az admin bejelentkezés a felhasználói bejelentkezéssel ellentétben nem emaillel, hanem felhasználónév – jelszó párossal történik.

Admint csak adatbázisból tudunk felvenni, hiszen nem szeretnénk, hogy bárki hozzáférjen az admin funkciókhoz. Bejelentkezés után a /admin végpontra ugrik az oldal, amelyet csak admin jogosultsággal lehet elérni.

4.4 Gazda felhasználói oldalak

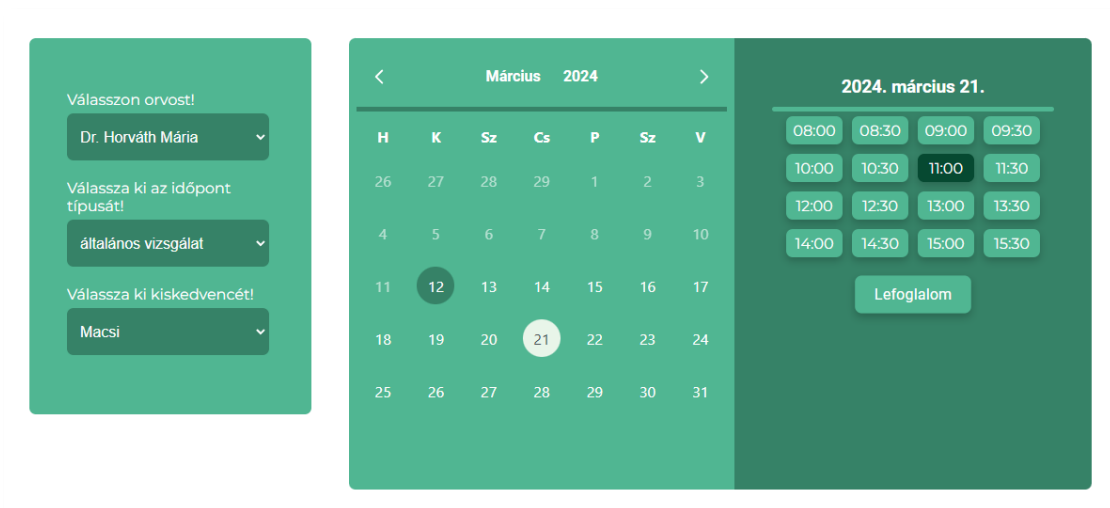
Bizonyos oldalak csak a gazdák számára elérhetőek. Ilyenek az például időpontfoglalás vagy kedvenceim.

4.4.1 Időpontfoglalás

Gazda bejelentkezés után elérhetővé válik az oldalon az időpontfoglalás funkció. Ez a /idopontfoglalas route-on található.

Időpontot foglalni bármelyik orvoshoz lehet. Az állatorvos böngésző (lásd: [4.3.3](#)) segíti a felhasználót a választásban. Amennyiben az Állatorvosok fülön kattint a felhasználó az orvos neve melletti „Időpontot foglalok” gombra, akkor az időpontfoglalás állatorvos mezőjében automatikusan kiválasztásra kerül a választott orvos.

A felhasználónak minden mezőt ki kell töltenie: választott orvos, időpont típusa, kezelendő kisállat, illetve a pontos időpont (nap, óra). Amennyiben a felhasználó az előbb felsoroltak közül valamelyiket nem tölti ki, akkor primevue-s toast értesítés formájában hibaüzenet jelenik meg a felhasználónak.



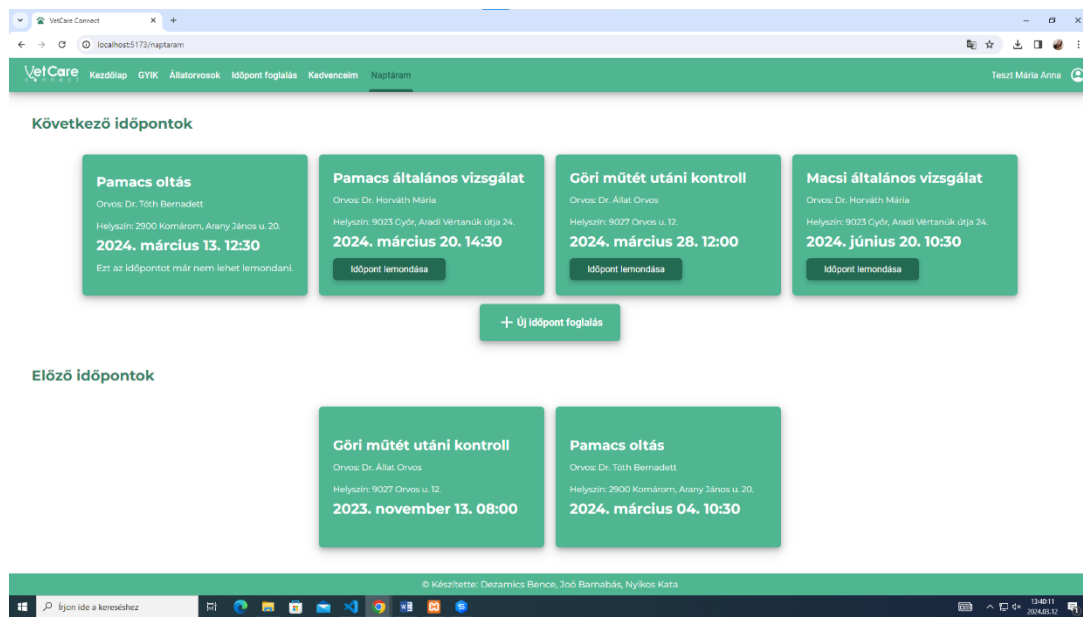
Időpontfoglalás

Időpontot foglalni az orvos megadott nyitvatartásának időtartamán belül fél óras intervallumokban lehet. Ellenőrizve van, hogy az adott napi dátum és idő előtti időpontokra ne lehessen foglalni. A szabad időpontokat a backendről kérjük le minden részlet változtatása után (orvos, nap), ezzel elkerülve azt, hogy esetleg egy időpontra két felhasználó foglalhasson kezelést a kiskedvencének.

Amikor minden mezőt kitöltött a felhasználó, a Lefoglalom gombra kattintva megjelenik neki egy komponens, ahol összegezve látja az időpont részleteit. Itt még vissza tud menni módosítani az időpont részleteit, vagy véglegesítheti a kezelést. A véglegesítés után a Naptáram menüponthoz dobja az oldal a felhasználót.

4.4.2 Naptáram

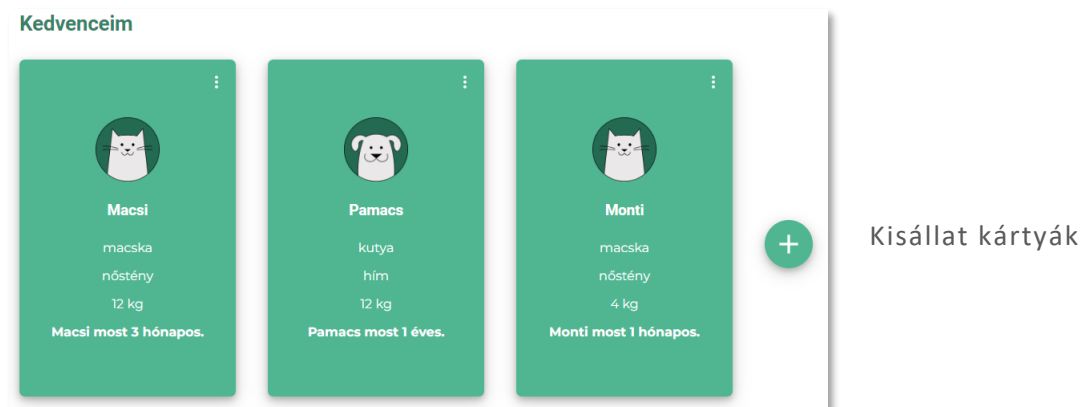
A Naptáram menüpont a /naptaam route-on keresztül érhető el a gazdák számára. Itt nyomon követheti az előző és jövőbeli időpontjait a felhasználó. Amennyiben az időpont 2 napon belül van, akkor még le lehet mondani az „Időpont lemondása” gombra kattintva.



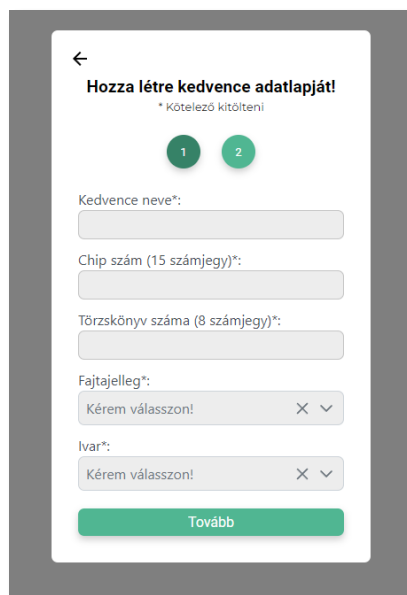
Gazda naptár

4.4.3 Kedvenceim

A felhasználó a Kedvenceim fül alatt, a /kedvenceim végponton tarthatja nyilván kisállatait. Lehetőség van állatok hozzáadására, módosítására és törlésére.



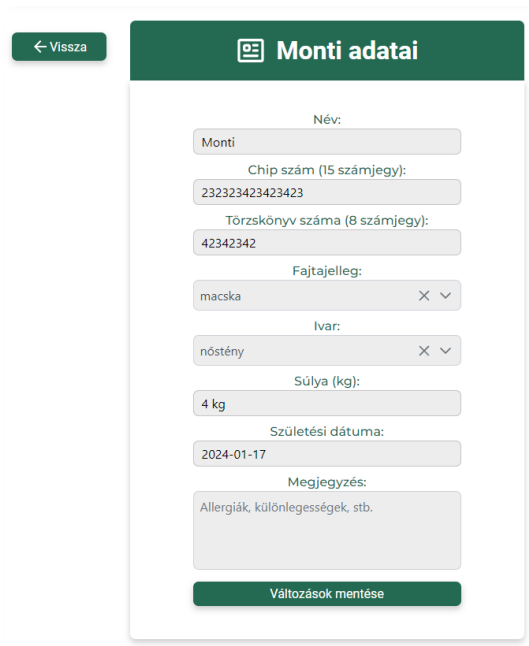
A + gombra kattintva adhat hozzá a gazda állatokat a profiljához. Meg kell adnia a megfelelő adatokat. Itt is – ugyancsak a regisztrációnál – minden mező validálva van. A törzskönyvszám, chipszám és a súly csak szám, míg a születési idő csak dátum típusú lehet. A megjegyzés mező kivételével – mely arra szolgál, hogy az állat bizonyos ismertetőjegyeit, allergiáit, stb. eltárolhassuk – minden mező kötelező, ezt jelezzük a felhasználó felé is (* kötelező kitölteni). Ha úgy próbál állatot elmenteni, hogy nem tölt ki minden kötelező mezőt, akkor itt is a primevue toast értesítésének segítségével jelezzük a felhasználónak a hibát. Amennyiben minden mezőt hiánytalanul kitöltött a gazda, belekerül az állat az adatbázisba.



Állat hozzáadása

Az állatok kártyákon jelennek meg a menüpontban. A kártyák jobb felső sarkába kattintva van lehetőségünk módosítani vagy törölni az állat adatait. A törlésre kattintva meg kell erősíteni, hogy biztosan ki szeretné-e törölni állatát. Megerősítés esetén az állat és annak időpontjai törölődnek az adatbázisból.

Módosítás választása esetén lehetőségünk van az állat valamennyi adatának módosítására. Amennyiben úgy akarjuk elmenteni az adatokat, hogy nem történik semmilyen változás, akkor erre vonatkozó figyelmeztetést kapunk egy toast értesítés segítségével. Sikeres adatmódosításkor erre vonatkozó értesítést kapunk.



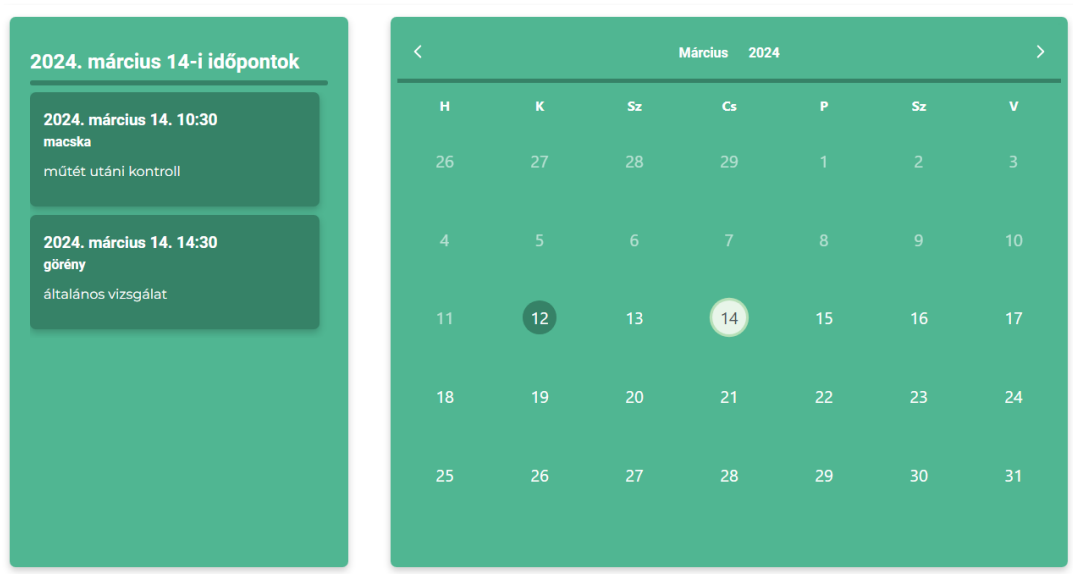
Állat adatainak szerkesztése

4.5 Orvos felhasználói oldalak

Az orvosi felhasználói fiók fő funkciója az állatorvos időpontjainak nyilvántartása, nyomon követése.

4.5.1 Naptáram

Az orvosok az időpontjaikat a /orvosi-naptar route-on érhetik el. Az oldalon található egy naptár, amelyben ki kell választania az orvosnak, hogy melyik napi időpontjait szeretné megtekinteni. A teendők komponensben látja az orvos az adott napi kezeléseket, azok időpontját, valamint, hogy milyen állatnak foglalták az időpontot.



Orvosi naptár

4.5.2 Nyitvatartás

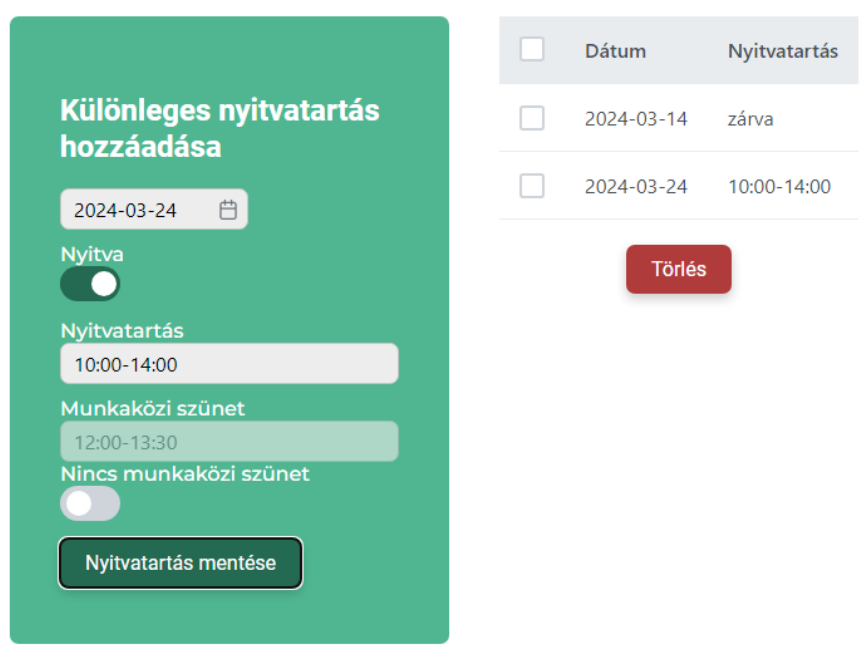
Az orvosok maguknak tudják állítani a nyitvatartási idejüket. Van egy általános nyitvatartás, amely minden hétre érvényes. Lehetőség van naponta különböző nyitvatartást beállítani, vagy pedig lehet hétköznapokra vagy minden napra beállítani nyitvatartást. A nyitvatartás háromféleképpen nézhet ki: zárva, nyitva – adott időtartamon belül (pl.: 08:00 – 16:00), nyitva – munkaközi szünettel (pl.: 08:00 – 12:00, 14:00 – 17:00).



Nyitvatartás módosítás

Az orvosoknak lehetőségük van speciális nyitvatartást is beállítani. Ez azt jelenti, hogy olyan nyitvatartást ad hozzá, ami egy bizonyos dátumra vonatkozik csak. A nyitvatartás-hozzáadás ugyanúgy működik, mint az általános nyitvatartás hozzáadása, viszont itt meg kell adni egy konkrét dátumot is.

Nyitvatartás hozzáadásnál validáljuk az időtartamokat: ne lehet olyan nyitvatartást hozzáadni, ami nem valid (pl.: 16:00-10:00 vagy a munkaközi szünet kilóg a munkaidőből).



The image shows a form for adding a special opening hours and a table of existing ones.

Különleges nyitvatartás hozzáadása

Date: 2024-03-24

Open: ☒

Opening hours: 10:00-14:00

Work break: 12:00-13:30

No work break: ☐

Save opening hours

Dátum	Nyitvatartás
2024-03-14	zárva
2024-03-24	10:00-14:00

Delete

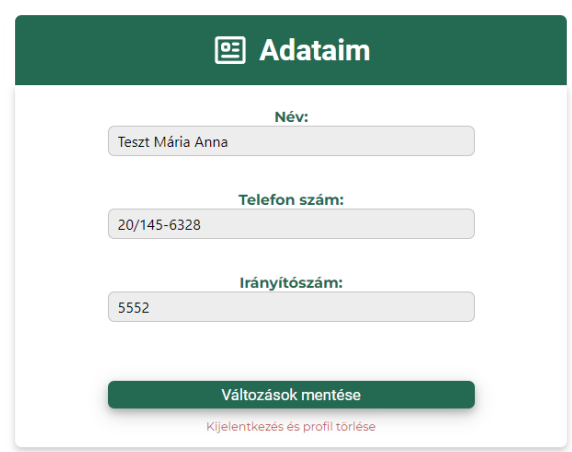
Különleges nyitvatartás

4.6 Adatmódosítás

Mindkét user típusnál van lehetőség az adatok módosítására. Ez mező van, amely nem módosítható felhasználó által, ez az email. Azonban a felhasználók email címét az admin tudja módosítani.

Mind a gazdák, mind az orvosok a /adataim route-on keresztül módosíthatják az adataikat. Itt megjelennek a felhasználó adatai. Amennyiben nem történik változás az adatokban, akkor a rendszer figyelmezteti a felhasználót primevue-s toast komponens segítségével. Sikeres mentés esetén is tájékoztatjuk a felhasználót a sikeres adatmódosításról.

Adatmódosítás



The image shows the 'Adataim' (My Data) form.

Adataim

Name: Teszt Mária Anna

Phone number: 20/145-6328

Postal code: 5552

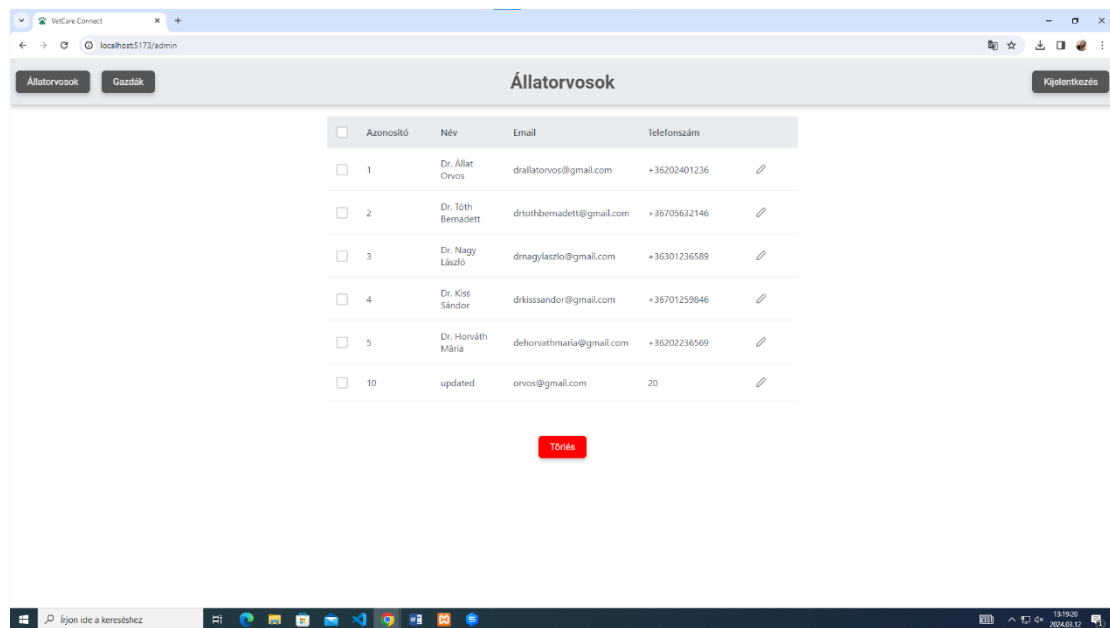
Save changes

Kijelentkezés és profil törlése

4.7 Admin oldal

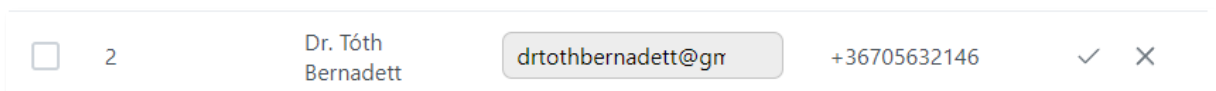
Vannak előre regisztrált adminok az oldalon. Ők arra szolgálnak, hogy tudják módosítani a felhasználók email címét, valamint törölni tudják a felhasználókat.

Bejelentkezés után (lásd: 4.3.6) a /admin végpontra irányítja a felhasználót az oldal. Itt az admin választani tud, hogy melyik felhasználó típust szeretné módosítani (állatorvos vagy gazda). Miután az admin kiválasztotta a felhasználó típust, utána megjelennek a felhasználók az oldalon.



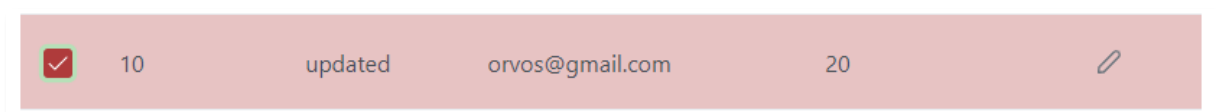
Admin oldal, állatorvosok adatai

A ceruza ikonra kattintva módosíthatóvá válik az email cím. Miután módosította az adatot, a pipára kattintva mentheti el, vagy vetheti el a módosított adatot.



Admin oldal – adatmódosítás

Törölni a cellák kijelölésével lehetséges. Miután kijelölte a törölni kívánt felhasználókat, a törlés gombra kattintva véglegesítheti a törlést. Minden esetben toast értesítés formájában tudatjuk a felhasználóval a művelet sikerességét.



Admin oldal – felhasználó törlés

4.8 Reszponzivitás

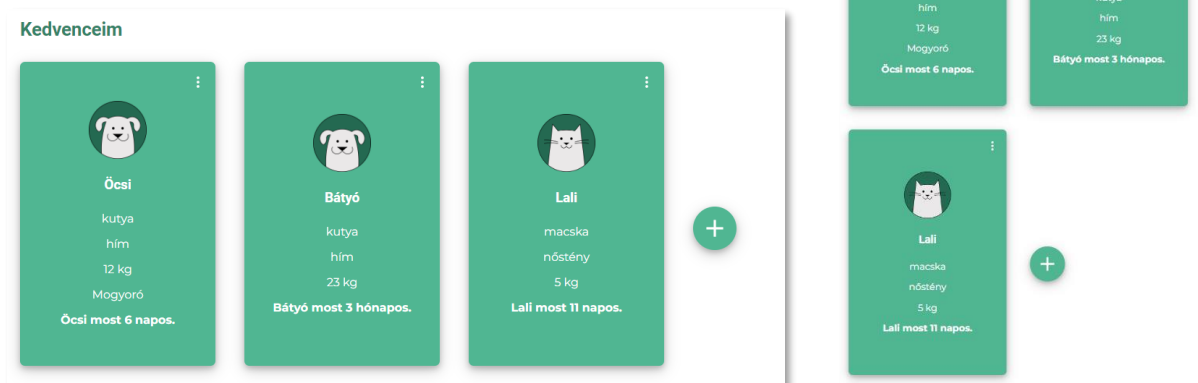
Egy részponzív elv alapján tervezett oldal tökéletesen igazodik a megjelenítő eszközhöz, mindezt rugalmas felépítéssel, flexibilis képekkel. Az asztali monitoroktól kezdve egészen a vékony kijelzőjű mobil telefonokig. Ezzel is segítve a felhasználókat a könnyebb kezelés érdekében. Akár az utcán sétálva is kényelmesen lehet foglalni időpontot, amikor éppen eszébe jutott, nem kell hazaérnie és az asztali gépén intézkednie.

A CSS media query segít nekünk definiálni egy ilyen weboldal stíluslapját. A rugalmas felosztású koncepció alapján a honlap minden elemének az igazítása százalékosan, relatívan van meghatározva. A flexibilis képek úgyszintén a befoglaló elemhez képest, százalékosan határozódnak meg. A media query alkalmazásával megvalósíthatjuk, hogy a weboldalon mindig olyan CSS szabályok lépjenek érvénybe, amelyek a megjelenítő eszközön optimálisak.

Oldaltól függően 3-4 töréspontot definiáltunk, komplexebb oldalnál akár ez lehet 5-6 is. Az optimális töréspontok megtalálása nagyon fontos, feleslegesen nem használunk sokat. Ezek általában 1330px, 991px, 891px, 720px, 430px, de ezektől néha, alkalmazkodva az adott oldal felépítéséhez egy kicsit eltérünk.

A részponzív weboldal felépítésében nagy szerepet játszott még a flexbox layout. Ennek a segítségével válik lehetségessé, hogy az elemeket egy adott sorba, oszlopba rendezzük. Meghatározhatjuk a fő irányokat, az elemek térközeit, illetve azt is, hogy az elemek új sorba csúszzanak-e, amennyiben már nem férnének ki, vagy pedig az elemek mérete csökkenjen, de azok maradjanak egy sorban.

A flexbox-ot szinte minden elem elhelyezéséhez, igazításához használtuk, mert alkalmazkodik a felhasználó képernyőjéhez, egyszerű használni, és százalékosan határozza meg a távolságokat a megadott helyeken.



Reszponzív megjelenés a kedvenceim oldalon

4.9 Frontend tesztelés

4.9.1 Selenium

A tesztelés nem csak a fejlesztés folyamatában fontos, hanem megakadályozhatjuk vele az esetleges felhasználói élmény csökkenését, azaz javítjuk a UX-et (User Experience – felhasználói élmény). Teszteléssel kiküszöbölhetőek az esetleges hibák, valamint a fejlesztés során keletkező problémák.

A Selenium webalkalmazások automatikus tesztelésére szolgáló keretrendszer, melyet a ThoughtWorks cég ír, és szabad szoftverként az Apache 2.0 licenc alatt ad ki. A Selenium széles körben használható eszköz, és ez az egyik legismertebb nyílt forrású teszteszköz.

A Selenium használatához először telepíteni kellett a node csomagot, amely tartalmazza a Selenium Webdrivert (`npm install selenium-webdriver`). Ezek után telepítettük a Chrome-hoz tartozó megfelelő verziójú webdrivert.

A tesztelés a `node .\tests.js` paranccsal futtatható.

4.9.2 Tesztek

A tesztelést oldalanként végeztük el.

```
PS D:\Nyikos_Kata\14AA-C-VetCareConnect\VetCareConnect-Frontend\tests> node .\tests.js
DevTools listening on ws://127.0.0.1:62689/devtools/browser/143431d5-49f9-4a5a-96a4-c375e3836333
✓ Testing faq page (1.7053ms)
✓ Testing vetsearch page (0.3031ms)
✓ Testing appointment booking page (0.2869ms)
✓ Testing pets page (0.2597ms)
✓ Testing appointments page (0.2637ms)
i tests 5
i suites 0
i pass 5
i fail 0
i cancelled 0
i skipped 0
i todo 0
i duration_ms 0.2762
```

Tesztek futtatása

Testing faq page:

- egyik oldal tesztelése
 - cím megjelenik
 - a cím megfelelő szöveggel jelenik meg
 - 4 gyakori kérdés található az oldalon

Testing vetsearch page:

- állatorvos böngésző oldal tesztelése
 - cím megjelenik
 - a cím megfelelő szöveggel jelenik meg
 - 3 beviteli mező jelenik meg

Testing appointment booking page:

- időpontfoglalás oldal tesztelése
 - cím megjelenik
 - a cím megfelelő szöveggel jelenik meg
 - 3 beviteli mező jelenik meg
 - a „Kérem válasszon!” az alapértelmezett szöveg a beviteli mezőkben
 - ha nincs kiválasztva orvos, akkor nem jelenik meg foglalható időpont

Testing pets page:

- kedvenceim oldal tesztelése
 - cím megjelenik
 - a cím megfelelő szöveggel jelenik meg
 - megjelenik az állat hozzáadó funkció gomb

Testing appointments page:

- naptáram oldal tesztelése
 - címek megjelennek
 - a címek megfelelő szöveggel jelennek meg
 - megjelenik az időpont hozzáadó funkció gomb

5. Backend

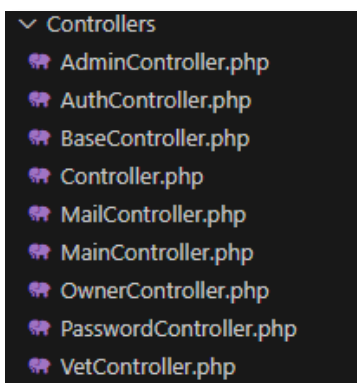
5.1 Technológia

A PHP keretrendszerek közül a Laravel az egyik legnépszerűbb, illetve legbiztonságosabb. Tökéletes a teljesítménye, segít felgyorsítani a fejlesztési folyamatot pár funkció segítségével.

Ez volt az egyik oka annak, hogy a Laravel-re esett a választás. A másik főbb ok, hogy kiváló párost alkot a Vue.js-el. A Laravel rengeteg beépített funkciót rejteget magában, amelyeket rendkívül egyszerű használni, bár ezek feltárása időigényesebb volt a vártnál. Mi után ezekkel tisztában voltunk, akkor már gyorsan lehetett haladni. Ilyen beépített funkciók például az email-ek verifikációjánál voltak hasznosak, ezen kívül a tesztekénél is Laravel-es teszt adatokat használtunk.

A Laravel egyszerű megoldást kínál a több nyelvű oldalaknak is, de jelen esetben ezt nem használtuk.

5.2 Controllerek



A controllerekben valósítjuk meg a megfelelő adatok biztosítását a végpontok számára. Több controllerre bontottuk szét a kódot, a könnyen olvashatóság és átláthatóság kedvéért.

A controllerek

5.2.1 AdminController

Az AdminController osztályba szerveztük azokat az API hívásokat, amelyeket, csak az admin érhet el. Ilyenek a felhasználók törlése, azok email címeinek módosítása.

getAllVet()

- Kérés típusa: GET
- Paraméter: nincs
- Jogosultság: admin
- Feladat: az összes orvos kilistázása
- Elérési útvonal: /api/vet-all

GET <http://127.0.0.1:8000/api/vet-all>

getAllOwner()

- Kérés típusa: GET
- Paraméter: nincs
- Jogosultság: admin
- Feladat: az összes gazda kilistázása
- Elérési útvonal: /api/owner-all

GET http://127.0.0.1:8000/api/owner-all

deleteVet(\$id)

- Kérés típusa: DELETE
- Paraméter: id – a törölni kívánt állatorvos egyedi azonosítója
- Jogosultság: admin
- Feladat: állatorvos törlése
- Elérési útvonal: /api/delete-vet/{id}

DELETE http://127.0.0.1:8000/api/delete-vet/{id}

deleteOwner(\$id)

- Kérés típusa: DELETE
- Paraméter: id – a törölni kívánt gazda egyedi azonosítója
- Jogosultság: admin
- Feladat: gazda törlése
- Elérési útvonal: /api/delete-owner/{id}

DELETE http://127.0.0.1:8000/api/delete-owner/{id}

modifyUserEmail (Request \$request)

- Kérés típusa: POST
- Paraméter: JSON objektum, amely tartalmazza a módosítani kívánt felhasználó egyedi azonosítóját (id), role-ját (0 – orvos, 1 – gazda, 2 – admin), illetve a módosított email címet
- Jogosultság: admin
- Feladat: felhasználók email címének módosítása
- Elérési útvonal: /api/modify-user-email

POST http://127.0.0.1:8000/api/modify-user-email

5.2.2 AuthController

Az AuthControllerben találhatók a bejelentkezéssel és regisztrációval kapcsolatos függvények.

register (Request \$request)

- Kérés típusa: POST
- Paraméter: JSON objektum, amely tartalmazza a felhasználók regisztrációjának adatait, felhasználó típustól függően (más-más adatai vannak egy gazdának, mint egy orvosnak)
- Jogosultság: olyan felhasználó, aki nincs bejelentkezve
- Feladat: regisztráció
- Elérési útvonal: /api/register

POST

<http://127.0.0.1:8000/api/register>

login (Request \$request)

- Kérés típusa: POST
- Paraméter: JSON objektum, amely tartalmazza a felhasználók bejelentkezésének adatait (email, jelszó)
- Jogosultság: olyan felhasználó, aki nincs bejelentkezve
- Feladat: regisztráció
- Elérési útvonal: /api/login

POST

<http://127.0.0.1:8000/api/login>

adminLogin (Request \$request)

- Kérés típusa: POST
- Paraméter: JSON objektum, amely tartalmazza az admin bejelentkezésének adatait (felhasználónév, jelszó)
- Jogosultság: olyan felhasználó, aki nincs bejelentkezve
- Feladat: admin bejelentkezés
- Elérési útvonal: /api/admin-login

POST

<http://127.0.0.1:8000/api/admin-login>

logout ()

- Kérés típusa: POST
- Paraméter: nincs
- Jogosultság: olyan felhasználó, aki be van jelentkezve
- Feladat: kijelentkezés
- Elérési útvonal: /api/logout

POST

<http://127.0.0.1:8000/api/logout>

logoutAllDevice ()

- Kérés típusa: POST
- Paraméter: nincs
- Jogosultság: olyan felhasználó, aki be van jelentkezve
- Feladat: kijelentkezés minden eszközről
- Elérési útvonal: /api/logout

POST

<http://127.0.0.1:8000/api/logout-all-device>

5.2.3 MainController

A MainControllerbe szerveztük azokat a függvényeket, amelyekhez nem kell bejelentkezni, vagy az összes bejelentkezett felhasználó típus elérheti.

searchVets (Request \$request)

- Kérés típusa: POST
- Paraméter: JSON objektum, amely tartalmazza a keresési paramétereket (név, irányítószám, cím)
- Jogosultság: mindenki
- Feladat: keresés az állatorvosok között, név, irányítószám és cím alapján
- Elérési útvonal: /api/search-vets

POST http://127.0.0.1:8000/api/search-vets

getAllCureTypes ()

- Kérés típusa: GET
- Paraméter: nincs
- Jogosultság: mindenki
- Feladat: állatorvosokhoz foglalható időpontok típusai
- Elérési útvonal: /api/cure-types-all

GET http://127.0.0.1:8000/api/cure-types-all

getAllQuestions ()

- Kérés típusa: GET
- Paraméter: nincs
- Jogosultság: mindenki
- Feladat: gyakori kérdések
- Elérési útvonal: /api/faq-all

GET http://127.0.0.1:8000/api/faq-all

getUserData ()

- Kérés típusa: GET
- Paraméter: nincs
- Jogosultság: bejelentkezett felhasználók
- Feladat: visszaadja a bejelentkezett felhasználó adatait
- Elérési útvonal: /api/user-data

GET http://127.0.0.1:8000/api/user-data

modifyUserData ()

- Kérés típusa: PUT
- Paraméter: JSON objektum, amely tartalmazza a módosított adatokat
- Jogosultság: bejelentkezett felhasználók
- Feladat: adatmódosítás
- Elérési útvonal: /api/modify-user-data

PUT http://127.0.0.1:8000/api/modify-user-data

5.2.4 OwnerController

Az OwnerControllerben olyan API-k találhatók, amelyeket csak a gazdák érhetnek el.

getPets ()

- Kérés típusa: GET
- Paraméter: nincs
- Jogosultság: bejelentkezett gazdák
- Feladat: visszaadja a gazda állatainak adatait
- Elérési útvonal: /api/pets

GET <http://127.0.0.1:8000/api/pets>

addNewPet (Request \$request)

- Kérés típusa: POST
- Paraméter: JSON objektum, amely tartalmazza az új állat adatait
- Jogosultság: bejelentkezett gazdák
- Feladat: új állat hozzáadása a felhasználó profiljához
- Elérési útvonal: /api/new-pet

POST <http://127.0.0.1:8000/api/new-pet>

modifyPet (Request \$request)

- Kérés típusa: PUT
- Paraméter: JSON objektum, amely tartalmazza az állat módosított adatait
- Jogosultság: bejelentkezett gazdák
- Feladat: állat adatainak módosítása
- Elérési útvonal: /api/modify-pet

PUT <http://127.0.0.1:8000/api/modify-pet>

deletePet (\$id)

- Kérés típusa: DELETE
- Paraméter: id – a törölni kívánt állat egyedi azonosítója
- Jogosultság: bejelentkezett gazdák
- Feladat: állat törlése
- Elérési útvonal: /api/delete-pet/{id}

DELETE <http://127.0.0.1:8000/api/delete-pet/{id}>

getFreeAppointments(\$id, \$date)

- Kérés típusa: GET
- Paraméter: id – orvos egyedi azonosítója, date – dátum, amelyik napra időpontot szeretne foglalni
- Jogosultság: bejelentkezett gazdák
- Feladat: szabad időpontok lekérése, dátum és orvos alapján
- Elérési útvonal: /api/free-appointments/{id}/{date}

GET <http://127.0.0.1:8000/api/free-appointments/{id}/{date}>

addNewAppointment (Request \$request)

- Kérés típusa: POST
- Paraméter: JSON objektum, amely tartalmazza az időpontfoglalás adatait
- Jogosultság: bejelentkezett gazdák
- Feladat: időpontfoglalás
- Elérési útvonal: /api/new-appointment

POST http://127.0.0.1:8000/api/new-appointment

getOwnerAppointments()

- Kérés típusa: GET
- Paraméter: nincs
- Jogosultság: bejelentkezett gazdák
- Feladat: visszaadja a felhasználó összes időpontját
- Elérési útvonal: /api/owner-appointments

GET http://127.0.0.1:8000/api/owner-appointments

deleteAppointment (\$id)

- Kérés típusa: DELETE
- Paraméter: id – a törölni kívánt időpont egyedi azonosítója
- Jogosultság: bejelentkezett gazdák
- Feladat: időpont törlése
- Elérési útvonal: /api/delete-pet/{id}

DELETE http://127.0.0.1:8000/api/delete-appointment/{id}

5.2.5 PasswordController

A jelszóhelyreállítással kapcsolatos függvények a PasswordControllerbe kerültek.

forgotPassword (Request \$request)

- Kérés típusa: POST
- Paraméter: JSON objektum, amely tartalmazza a felhasználó email-ét.
- Jogosultság: olyan felhasználó, aki nincs bejelentkezve
- Feladat: token generálása és jelszó visszaállító email küldése
- Elérési útvonal: /api/forgot-password

POST http://localhost:8000/api/forgot-password

resetPassword (Request \$request)

- Kérés típusa: PATCH
- Paraméter: JSON objektum, amely tartalmazza a felhasználó adatait (email, új jelszó, jelszó újra, szerep).
- Jogosultság: olyan felhasználó, aki nincs bejelentkezve
- Feladat: jelszó módosítása
- Elérési útvonal: /api/reset-password

PATCH http://localhost:8000/api/reset-password

5.2.6 VetController

A VetControllerben találhatóak, azok az API hívások, amelyekhez csak az állatorvosoknak van hozzáférésük.

getVetAppointments (\$date)

- Kérés típusa: GET
- Paraméter: dátum
- Jogosultság: bejelentkezett orvos
- Feladat: orvos adott napi időpontjainak adatait hívja elő
- Elérési útvonal: /api/vet-appointments/{date}

GET <http://localhost:8000/api/vet-appointments/{date}>

getOpenings ()

- Kérés típusa: GET
- Paraméter: nincs
- Jogosultság: bejelentkezett orvos
- Feladat: nyitvatartások előhívása
- Elérési útvonal: /api/openings

GET <http://127.0.0.1:8000/api/openings>

addOpenings (Request \$request)

- Kérés típusa: POST
- Paraméter: JSON objektum, amely tartalmazza a nyitvatartás adatait (mettől-meddig, hét napja).
- Jogosultság: bejelentkezett orvos
- Feladat: új nyitvatartási időpont létrehozása
- Elérési útvonal: /api/new-openings

POST <http://127.0.0.1:8000/api/new-openings>

modifyOpening (Request \$request)

- Kérés típusa: PUT
- Paraméter: JSON objektum, amely tartalmazza a nyitvatartás adatait (egyedi azonosító, mettől-meddig, hét napja).
- Jogosultság: bejelentkezett orvos
- Feladat: nyitvatartás módosítása
- Elérési útvonal: /api/modify-opening

PUT <http://127.0.0.1:8000/api/modify-opening>

deleteOpening (\$day)

- Kérés típusa: DELETE
- Paraméter: a hét napja
- Jogosultság: bejelentkezett orvos
- Feladat: nyitvatartás törlése
- Elérési útvonal: /api/delete-opening/{day}

DELETE

<http://127.0.0.1:8000/api/delete-opening/{day}>

getSpecialOpenings ()

- Kérés típusa: GET
- Paraméter: nincs
- Jogosultság: bejelentkezett orvos
- Feladat: speciális nyitvatartások előhívása
- Elérési útvonal: /api/special-openings

GET

<http://127.0.0.1:8000/api/special-openings>

addSpecialOpenings (Request \$request)

- Kérés típusa: POST
- Paraméter: JSON objektum, amely tartalmazza a nyitvatartás adatait (mettől-meddig, hét napja).
- Jogosultság: bejelentkezett orvos
- Feladat: új speciális nyitvatartás hozzáadása
- Elérési útvonal: /api/new-special-openings

POST

<http://127.0.0.1:8000/api/new-special-openings>

modifySpecialOpening (Request \$request)

- Kérés típusa: PUT
- Paraméter: JSON objektum, amely tartalmazza a nyitvatartás adatait (egyedi azonosító, mettől-meddig, hét napja).
- Jogosultság: speciális nyitvatartás módosítása
- Feladat: jelszó módosítása
- Elérési útvonal: /api/modify-special-opening

PUT

<http://127.0.0.1:8000/api/modify-special-opening>

deleteSpecialOpening (\$id)

- Kérés típusa: DELETE
- Paraméter: egyedi azonosító
- Jogosultság: bejelentkezett orvos
- Feladat: speciális nyitvatartás törlése
- Elérési útvonal: /api/delete-special-opening/{id}

DELETE


<http://127.0.0.1:8000/api/delete-special-opening/{id}>

5.2.7 MailController

A regisztrációkor meg kell erősíteni az email fiókból a regisztrációt. Az ezzel kapcsolatos API-k kerültek bele a MailControllerbe.

verify(\$user_id, Request \$request)

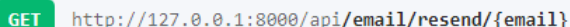
- Kérés típusa: GET
- Paraméter: user_id – felhasználó egyedi azonosítója
- Jogosultság: olyan felhasználó, aki regisztrál
- Feladat: regisztráció utáni megerősítő email küldése
- Elérési útvonal: /api/email/verify/{user_id}



```
GET http://127.0.0.1:8000/api/email/verify/{user_id}
```

resend (\$email)

- Kérés típusa: GET
- Paraméter: email – a felhasználó email címe
- Jogosultság: regisztrált felhasználók
- Feladat: regisztráció utáni megerősítő email újraküldése
- Elérési útvonal: /api/email/resend/{email}



```
GET http://127.0.0.1:8000/api/email/resend/{email}
```

5.3 Emailek

Az alkalmazásunk megfelelő működéséhez nagyon fontos, hogy informáljuk a felhasználókat, de az email-ek küldése nem csak ezt a célt szolgálja. Ezen kívül biztonsági funkciót is ellát, ugyanis a regisztrációhoz szükséges email megerősítése egyben a felhasználó személy azonosságát is megerősíti. Másrészt azért fontos ez az email cím, mert, ha elfelejtené a jelszavát, email-en keresztül lehet újat létrehozni. A VetCare Connect öt esemény bekövetkezésénél küld levelet: regisztráció, jelszó visszaállítás, időpont foglalásnál, időpont lemondásnál és időpont előtt egy nappal.

Az email-ek küldéséhez SMTP-t használunk, ami a Simple Mail Transfer Protocol rövidítése, ami kommunikációs protokoll az e-mailek Interneten történő továbbítására. Ehhez az 587-es portot használjuk. Másik kötelező beállítás a két faktoros email fiók.

5.3.1 Regisztráció

Abban az esetben, ha a felhasználó (akár gazda, akár orvos) sikeresen megadja a regisztrációs adatokat, melyek meg is felelnek a feltételeinknek, küldünk a megadott email fiókjába egy levelet, amelyben könnyedén egy gomb segítségével, vagy link böngészőbe másolásával meg erősítheti az email címét. Ezzel kiszűrhetjük azt, hogy illetéktelen emberek használják mások email fiókját.

Ha a linkben található token esetleg nem működne, vagy (gyakrabban) lejárna, a felhasználó újra kérheti az email-t új tokenrel. A token 30 perc alatt jár le.

5.3.2 Jelszó visszaállítása

Gyakran előfordul, hogy elfelejtjük a jelszót, ezért új jelszó létrehozást is biztosítunk, amihez szintén kulcs fontosságú az email. A felhasználónak, akinek ilyen problémája van, email cím megadása után egy generált 60 karakter hosszú token-t generálunk, amit elküldünk neki levélben, egy link formájában. Megkönnyítettük a felhasználók dolgát, mivel egy szimpla gombnyomással hitelesítjük, és átirányítjuk újra a megfelelő oldalra.

5.3.3 Időpontfoglalás, lemondás

Miután egy felhasználó sikeresen lefoglalt egy időpontot, fontos, hogy visszaigazoljuk erről levélben, minden időponttal kapcsolatos adattal. Erről csak a gazdát értesítjük, az orvost nem. Ennek az az oka, hogy ne árásszuk el levelekkel, hiszen úgy is megtudja tekinteni az orvosi naptárában.

Az időpont lemondásáról mindkét felet értesítjük, hiszen szeretnénk, hogy az orvos időben értesüljön erről, ne akadályozzuk a munkavégzésben nem naprakész információkkal. Az időpontot csak kettő vagy több nappal az időpont előtt lehet lemondani.

5.3.4 Időzített értesítés

A kellemetlenségek elkerülése érdekében értesítjük a gazdákat a közelgő időpontjukról, megelőzve azt, hogy az orvos ne hasznosan töltse el a munkaidejét. Ezt a levelet egy nappal az időpontja előtt küldjük, minden információval.

Ezt naponta ellenőrizzük, hogy mely gazdák időpontjai közelegnek, és nekik küldünk róla értesítést.

5.4 Emailek felépítése

Az email-ek felépítéséhez egy előre összeállított mintát alakítottunk át olyanná, amilyen illik az oldalunk megjelenéséhez. Fontos volt számunkra az egységes megjelenés minden platformon: a weboldalunkon és az emailekben egyaránt.

Minden egyes levél megjelenését külön lehet összeállítani, alakítani, de a Laraveles template minden elvárásunknak megfelelt.

Igyekeztünk úgy kialakítani a levelek formátumát és tartalmát, hogy a lehető legjobb felhasználói élményt nyújtsák. Lényegre törő, tömör, minden adatot tartalmazó emaileket igyekeztünk küldeni és emellett esztétikus megjelenést biztosítani nekik.

Ezeket a leveleket úgynevezett „blade” fájlokkal csináltuk. Ezek a view fájlok egy Laravel-es Blade keretrendszer szolgáltatás részei.



Az emailek design-ja

5.5 Autentikáció

Az autentikáció szerepe elengedhetetlen, mert a felhasználók adatait nem hagyhatjuk védtelenül. Minden felhasználó csak a saját adatait láthatja, illetve módosíthatja.

Alapvetően 3 felhasználót különböztetünk meg, az állatorvosokat, gazdákat, illetve az adminokat. Külön táblában tároljuk a felhasználókat, mivel mindegyiknek egyedi attribútumai és funkciói vannak. Állatorvosokat és gazdákat email címmel és jelszóval azonosítunk, míg adminokat felhasználónév jelszó párossal. A felhasználónévnek és az email címnek egyedinek kell lennie a megfelelő azonosítás érdekében.

A tokenekben tárolt információkból tudjuk megállapítani a felhasználók jogosultságait. Az API végpontokat middlewarekkel védjük a jogosulatlan felhasználók ellen.

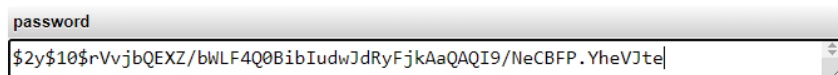
vetcareconnect vet	vetcareconnect owner	vetcareconnect admin
id : int(11)	id : int(11)	id : int(15)
name : varchar(255)	name : varchar(255)	username : varchar(150)
email : varchar(50)	email : varchar(255)	password : varchar(150)
password : varchar(255)	password : varchar(255)	
postal_code : int(4)	postal_code : int(4)	
phone : varchar(255)	phone : varchar(15)	
address : varchar(255)		
stamp_number : int(4)		
email_verified_at : date	email_verified_at : date	

Felhasználó táblák

5.6 Jelszótitkosítás

Az adatbázisban titkosítva tároljuk a jelszavakat, az érzékeny adat védelme miatt, hogy aki hozzáfér az adatbázishoz se jusson hozzá a felhasználók jelszavaihoz.

A jelszavak tárolásához bcrypt titkosítást használunk, melynek adaptív hashelési funkciója lehetővé teszi a számítási költség beállítását, így ellenállóvá teszi a „brute force” támadásokkal szemben. A bcrypt tartalmaz egy „salt” értéket is a „rainbow” táblák elleni védelem érdekében, és beépített iterációs számot, mely lassítja a hashelési folyamatot.



Titkosított jelszó

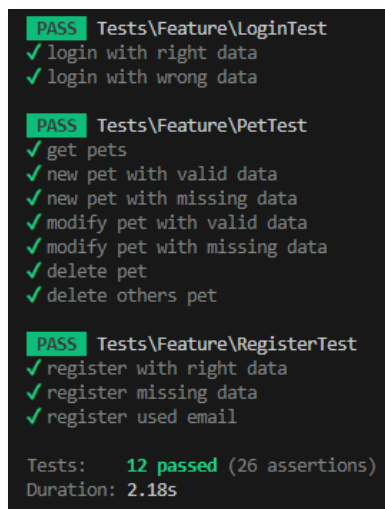
5.7 Backend tesztek

A backend tesztekhez laravel feature tesztjeit alkalmaztuk. Az átláthatóság és a tiszta kód miatt factorykat használunk, amikhez az adatokat faker beépített könyvtár segítségével állítjuk elő.

A teszteknel üres adatbázist használunk és a factoryk, illetve api hívások segítségével állítjuk elő az adott teszthez szükséges adatokat. Minden teszt után laravel DatabaseTransactions segítségével visszaállítjuk az adatbázis tartalmát a teszt előtti állapotába.

```
class OwnerFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        return [
            "name" => fake()->name(),
            "email" => fake()->unique()->safeEmail(),
            "password" => fake()->password(),
            "postal_code" => fake()->numberBetween(1000, 9999),
            "phone" => fake()->randomNumber(9, true),
            "email_verified_at" => fake()->date(),
        ];
    }
}
```

Owner factory



A végpontokat teszteltük státuszkód, adatbázis tartalma és a visszaküldött adat alapján. Nem csak helyes, hanem hiányos vagy nem megfelelő adatokkal egyaránt ellenőrizzük.

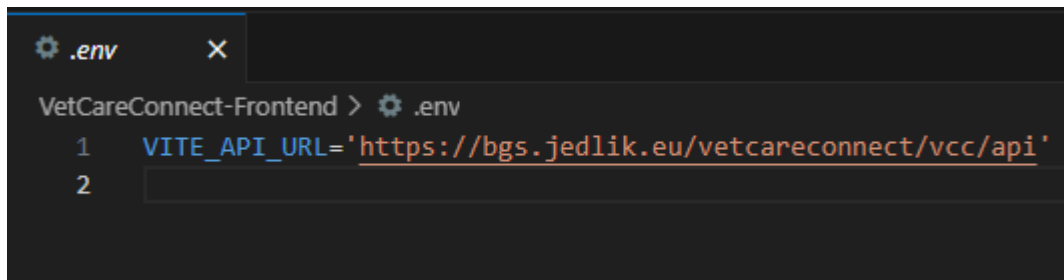
Tesztelés eredménye

6. Fejlesztői futtatási kézikönyv

A projektünk repository-ja GitHubon elérhető <https://github.com/dezbence/14AA-C-VetCareConnect> linken. Parancssorból, a megfelelő mappa kiválasztása után, a `git clone repositoryURL` parancsot kiadva tölthető le a projekt. A VetCareConnect-Backend mappában egy `composer install`, a VetCareConnect-Frontend mappában pedig egy `npm install` parancsra van szükség, ahhoz, hogy a csomagok települjenek.

Visual Studio Code-dal megnyitva a projektet a backend és a frontend fájljait külön mappában találjuk. A megfelelő mappában futtatva a `php artisan serve` parancssal tudjuk a backendet elindítani, míg a frontend az `npm run dev` parancssal indul.

A repository Frontend mappájában lévő `.env` fájl a backendet a szerverrel köti össze, így a backendes módosítások a weboldalon nem jelennek meg egyből. Ezért fejlesztéshez érdemes a lokálisan futó backend url-jét átírni a `.env` fájlban.



```

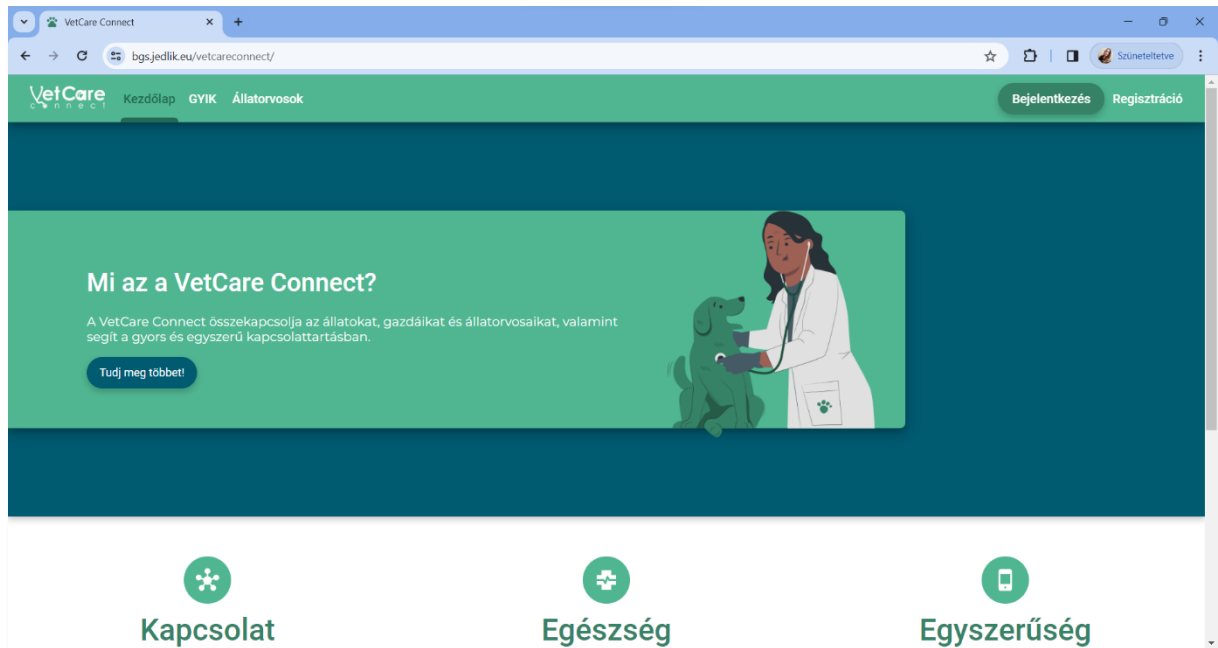
VetCareConnect-Frontend > .env
1  VITE_API_URL='https://bgs.jedlik.eu/vetcareconnect/vcc/api'
2

```

.env fájl, és a backend forrását tartalmazó környezeti változó

7. Publikálás

A VetCare Connect oldala a <https://bgs.jedlik.eu/vetcareconnect/> url-en érhető el. Úgy döntöttünk, hogy projektünket publikáljuk, ehhez az iskolai szervert használtuk. A frontendet ehhez le kellett build-elni, a backend pedig kisebb átalakításokon esett át, ugyanis a php-s public mappát külön kellett feltölteni a szerverre. Emiatt néhány php fájlban át kellett írni az elérési utakat.



VetCare Connect a <https://bgs.jedlik.eu/vetcareconnect/> url-en

8. Források

- Wikipédia: <https://hu.wikipedia.org/wiki/GitHub> (2024.04.11.)
- HelloWP Hub: <https://hub.hellowp.io/docs/tudasbazis/oktatoanyagok/github/github-kezdoknek/> (2024.04.11.)
- Wikipédia: <https://hu.wikipedia.org/wiki/MySQL> (2024.04.11.)
- A&K akadémia: <https://ak-akademia.hu/mi-az-a-vue-js/> (2024.04.11.)
- Wikipédia: <https://hu.wikipedia.org/wiki/Selenium> (2024.04.11.)
- Wikipédia: <https://en.wikipedia.org/wiki/Bcrypt> (2024.04.11.)
- NordVPN: <https://nordvpn.com/blog/what-is-bcrypt/> (2024.04.11.)
- Wikipédia: <https://en.wikipedia.org/wiki/Laravel> (2024.04.11.)