

Vizsgaremek

Kutyakozmetika



2022. április 19.

Készítette: Barna Marcell Tamás, Dömösi András, Saceanu Krisztián

**Tartalomjegyzék:**

[Projektről: 3](#_Toc101277424)

[Projekt neve: 3](#_Toc101277425)

[Projekt célja: 3](#_Toc101277426)

[Projekt témája: 3](#_Toc101277427)

[Projekt komponensei: 3](#_Toc101277428)

[Csapatmunkáról: 3](#_Toc101277429)

[Asztali alkalmazás (Desktop Version) 4](#_Toc101277430)

[A desktop alkalmazás témája: 4](#_Toc101277431)

[Az asztali alkalmazás megvalósítása: 4](#_Toc101277432)

[1.Lépés 5](#_Toc101277433)

[2.Lépés 6](#_Toc101277434)

[3.Lépés 7](#_Toc101277435)

[4.Lépés 8](#_Toc101277436)

[5.Lépés 9](#_Toc101277437)

[6.Lépés 10](#_Toc101277438)

[7.Lépés: 11](#_Toc101277439)

[8.Lépés 11](#_Toc101277440)

[9.Lépés 12](#_Toc101277441)

[10.Lépés 13](#_Toc101277442)

[11.Lépés 14](#_Toc101277443)

[12.Lépés 15](#_Toc101277444)

[Mobil alkalmazás 16](#_Toc101277445)

[Mobil alkalmazás célja: 16](#_Toc101277446)

[Mobil alkalmazás megvalósítása: 16](#_Toc101277447)

[1.Lépés : 16](#_Toc101277448)

[2.Lépés 17](#_Toc101277449)

[3.Lépés 18](#_Toc101277450)

[4.Lépés 18](#_Toc101277451)

[5.Lépés 20](#_Toc101277452)

[6.Lépés 20](#_Toc101277453)

[7.Lépés 21](#_Toc101277454)

[Backend a mobil alkalmazáshoz 23](#_Toc101277455)

[Webes alkalmazás: 25](#_Toc101277456)

[Webes alkalmazás témája: 25](#_Toc101277457)

[Program megvalósítása 25](#_Toc101277458)

[1.Lépés 25](#_Toc101277459)

[2.lépés 26](#_Toc101277460)

[3.Lépés 27](#_Toc101277461)

[4.Lépés 27](#_Toc101277462)

[5.Lépés 28](#_Toc101277463)

[Web API a webes alkalmazáshoz 29](#_Toc101277464)

[Web API témája: 29](#_Toc101277465)

[Web Api megvalósítása: 29](#_Toc101277466)

[1.Lépés 29](#_Toc101277467)

[2.Lépés 29](#_Toc101277468)

[3.Lépés 30](#_Toc101277469)

[4.Lépés 30](#_Toc101277470)

[5.Lépés 30](#_Toc101277471)

[6.Lépés 31](#_Toc101277472)

[7.Lépés 31](#_Toc101277473)

[Adatbázis 33](#_Toc101277474)

[Tervezés 33](#_Toc101277475)

[Egyedek: 33](#_Toc101277476)

[Tulajdonságok: 33](#_Toc101277477)

[Egyedek és tulajdonságaik(adattípusok): 33](#_Toc101277478)

[Az adatbázis E-K diagramja 35](#_Toc101277479)

[Az adatbázis Bachmann-ábrája 35](#_Toc101277480)

Projektről:

# **Projekt neve:**

Doggo Session

# **Projekt célja:**

A szoftver rendszer célja, hogy a felhasználók egyszerűbben tudjanak szolgáltatást rendelni. Célja a projektnek egy újkeletű vállalkozás látványosabbá tétele. Mivel a virtuális világ küszöbén állunk, a projekt megvalósításával egy új szintre emelhető a kutyakozmetika fogalma.

# Projekt témája:

Kutyakozmetikához tartozó szolgáltatások egyszerűbbé tétele.

# Projekt komponensei:

* Desktop alkalmazás
* Mobil applikáció
* Webes alkalmazás
* Backend
* Adatbázis

# Csapatmunkáról:

Munkánk során fontos szerepet játszott a csapatmunka. Minden feladatot egyenlő részre bontottunk. A munkánkat GitHubra töltöttük fel különböző repositorykba. Heti konzultációkat tartottunk, ahol mindig megbeszéltük a program jelenlegi állását, illetve felmerülő problémákat elhárítottuk. A teljes dokumentációt együtt írtuk meg, minden részt közösen fogalmaztunk meg.

Asztali alkalmazás (Desktop Version)

# A desktop alkalmazás témája:

Az asztali alkalmazásunk témája, egy kutyakozmetikában alkalmazott adatkezelő program, ahol a kozmetikus nyomon követheti, a vendégek/felhasználok által rögzített adatokat. Ez egy új gondolat, amivel szeretnénk a kutyakozmetikákat kicsit modernebbé tenni. Az esetek 90%-ban még a megszokott módon történik az adatkezelés. Kozmetikánként van egy kis napló ahová felírják az időpontokat, a kutyák illetve a gazdi adatait. Napjainkban az információs technológia fejlődését nézve ez már eléggé elavultnak számít.

# Az asztali alkalmazás megvalósítása:

Az asztali alkalmazás C# nyelven íródott és a programot a Microsoft Visual Studio fejlesztői környezet használatával valósítottuk meg. Két verzióját használtuk a Visual Studio-nak a 2019-es és a 2022-eset. A programban mapparendszert alkalmaztunk:

Mivel rengeteg osztályt tartalmaz egy-egy mappa, ezért nélkülük átláthatatlan lett volna, hogy melyik osztály milyen szerepet tölt be a munkánkban.

A képen szöveg látható

Automatikusan generált leírás

1. ábra Mapparendszer

## 1.Lépés

Elsőként létrehoztuk a „Views” mappát és azon belül a MainWindow-t, ahol Grid-ek használatával felosztottuk részekre az ablakot. Itt egy Label-ben szerepel a programunk neve és annak megjelenésének formázása. Felvettük az első gombunkat, ami a kilépésért felelős. A „Kilépés” gombra kattintva a program bezárul.

A képen szöveg látható

Automatikusan generált leírás

2. ábra A „Kilépés” gomb kódolása

Ezután a „Pages” mappa jött létre, ahol egy WPF ablakban egy üdvözlő szöveg jelenik meg. Ezt a „Navigate” osztályban a „Navigation” metódus használatával átadjuk a MainWindow-nak és a program indulásakor a „WelcomePage” is betöltődik.

A képen szöveg látható

Automatikusan generált leírás

3. ábra WelcomePage

## 2.Lépés

A program fejlesztése során bekerült a „ProgramInfo” gombunk, ahol a program aktuális verzióját tekinthetjük meg. Későbbiekben itt feltüntetjük majd a programot fejlesztő céget és fejlesztők GitHub felhasználónevét is. A megvalósításhoz egy WPF ablakot és két új mappát hoztunk létre, a „Models” és a „ViewModels” mappát, ahol egy-egy új osztály segíti a program verziójának megjelenítését.

A képen szöveg látható

Automatikusan generált leírás

4. ábra A „ViewModels” mappában szereplő „ProgramInfoViewModel” kód

A kód megírása után ellenőriztük a megjelenést, a program gond nélkül elindult és az adatok helyes megjelentek. Folytattuk a fejlesztést és felvettük a szükséges string típusú változókat, amiket példányosítottunk, majd állítható és módosítható tulajdonságokkal ágyaztuk be a programba. Ezeket a változókat a „ProgramInfo.xaml” oldalon elérési útvonalként adtuk meg, ami ezután megjeleníti a kívánt adatokat.

A képen szöveg látható

Automatikusan generált leírás

5. ábra A megírt kód eredménye

## 3.Lépés

A vizsgaremekünk egyik fő komponense az adatbázis helyének kiválasztása. A programunk két fajta adattal dolgozik, amit teszt adatként vittünk fel és amit a helyi adatbázisból lekér a program . A helyi adatbázist a XAMPP felhasználásával oldottuk meg. A XAMPP egy szabad és nyílt forrású platformfüggetlen webszerver-szoftvercsomag, amelynek legfőbb alkotóelemei az Apache webszerver, a MariaDB (korábban a MySQL) adatbázis-kezelő, valamint a PHP és a Perl programozási nyelvek értelmezői (végrehajtó rendszerei). Két választási lehetőséget tüntettünk fel a „test”(alapértelmezett) és a „localhost”-ot. A választott adatbázisoktól függően töltődnek be az adatok a megfelelő táblába. Itt még a táblázatokkal nem foglalkoztunk, arra majd a fejlesztés későbbi részében kerül sor.

A képen szöveg látható

Automatikusan generált leírás

6. ábra A „DatabaseSourceViewModel”-ben megírt kód

## 4.Lépés

Az alkalmazás két nyelven használható. Az alapértelmezett nyelv a magyar azonban ezt a kozmetikus szabadon átállíthatja angolra is. Erre a „Nyelv választás” gombra kattintás után lesz lehetősége. Újabb mappát hoztunk létre „Resources” néven. A mappához két „ResourceDictionary” forrás könyvtárat adtunk hozzá. Itt tároljuk el az angol és a magyar szöveget gombonként.

A képen szöveg látható

Automatikusan generált leírás

7. ábra Az angol nyelvű „Dictionary”

A nyelvválasztás egyik kulcsfontosságú kód részletét a „MainWindow.xaml.cs”-n belül írjuk meg. Ennek a „SetLanguageDictionary” metódus nevet adtuk. Ez a metódus segít abban, hogy a nyelv kiválasztása után a gombok szövege módosuljon.

*A képen szöveg látható

Automatikusan generált leírás*

8. ábra A ,,SetLanguageDictionary” kódja

## 5.Lépés

Munkánk során a csapat összes tagja létrehozott egy-egy osztályt a „Models” rétegben, amelyben a program témájához kapcsolódóan felvettük a szükséges adattagokat és a hozzájuk tartozó konstruktorokat, illetve ezeket állíthatóvá és módosíthatóvá tettük.

A képen szöveg, elektronika látható

Automatikusan generált leírás

9. ábra Models mappa osztályai

## 6.Lépés

Minden főbb osztálynak létrehoztunk egy-egy „UserControl”-t, ami a megjelenítésért felel. Mielőtt megírtuk a hozzá szükséges kódot, figyelembe kellett vennünk azt, hogy az adatbázisban, miként épülnek fel a táblák. Létrehoztunk egy táblázatot, ahova az adatok úgy töltődnek be, hogy minden oszlophoz elérési útvonalat adtunk „Binding” használatával és így minden cellába, a megfelelő adat jelenik meg.

A képen szöveg látható

Automatikusan generált leírás

10. ábra A „DogPage”-ben az adatok megjelenítéséhez megírt kód

7.Lépés:

A „Page”-kben szereplő táblázatok adatai, a „Controls” mappában használt „UserControl”-ok segítségével jelennek meg a táblázat alatt, ahol ezek az adatok módosíthatóak illetve törölhetőek. Itt használtuk a validálást, ami azért felelős, hogy a felhasználó egy adott szabály szerint módosíthat az adatokon („A mező nem lehet üres!”).

A képen asztal látható

Automatikusan generált leírás

11. ábra A táblázat alatt a kiválasztott sor adatinak betöltése „control”-ba és a validáció által jelzett figyelmeztetés.

## 8.Lépés

A „ViewModel” a nézet absztrakciója, ami publikus tulajdonságokat és metódusokat tartalmaz. Ennek megfelelően írtuk meg a szükséges metódusokat és publikus tulajdonságokat a „ViewModels”-ben szereplő mappában. Létrehoztunk egy „ViewModelBaseClass”osztályt, ami a program futása során végzett módosítások végrehajtásáért felelős.

A képen szöveg látható

Automatikusan generált leírás

12. ábra A „ViewModelBaseClass” kódja

A képen szöveg látható

Automatikusan generált leírás

13. ábra A „ViewModels” mappában szereplő osztályok

## 9.Lépés

A képen szöveg látható

Automatikusan generált leírásA „Repositories” mappában szereplő osztályok felelnek a teszt adatok tárolásáért, illetve itt találhatóak a „localhost”-os lekéréshez használt „API” osztályok. Itt szerepel egy „Interface” mappa, amelyben egy interfész szerepel a következő metódusokkal:

14. ábra Interfész

A „Repositories” mappa minden osztálya implementálja az imént említett interfészt.

A képen szöveg látható

Automatikusan generált leírás

15. ábra A „Repositories” osztályai

## 10.Lépés

A „Commands” mappában létrehoztunk egy „CommandBase” absztrakt osztályt, ami implementálja az „Icommand” beépített interfészt. Ez az osztály felelős a műveletek végrehajtásáért.

A képen szöveg látható

Automatikusan generált leírás

16. ábra „CommandBase” absztrakt osztály

Emellett a mappában megtalálható minden műveletre (Update, Delete) egy külön osztály. Az imént említett összes osztály örökli „CommandBase” nevezetű absztrakt class-t. Ahogyan a lenti ábrán látható, a törlés ID szerint történik.

A képen szöveg látható

Automatikusan generált leírás

17. ábra „DeleteDogCommand” osztály kódja

## 11.Lépés

Ebben a lépésben a backend részét írtuk meg a programnak. A backend a programoknak, weboldalaknak a hátsó, a felhasználó elől rejtett, a tényleges számításokat végző része. Feladata a frontend felől érkező adatok feldolgozása, és az eredményeknek a frontend felé történő visszajuttatása. Ebben az esetben a phpMyAdmin felületen szereplő adatbázisunkat kötöttük össze az asztali alkalmazásunkkal, amit úgy hajtottunk végre, hogy a Package Manager Console-on kiadtuk a scaffold parancsot és ez legenerálta a context osztályt. A parancs kiadása előtt feltelepítettük a Pomelo.EntitiryFrameworkCore.MySql csomagot, ami szükséges ahhoz, hogy kommunikáljon egymással a két felület.

A képen szöveg, képernyőkép, képernyő, fekete látható

Automatikusan generált leírás

18. ábra „Scaffold” parancs

## 12.Lépés

Az utolsó lépésben a teszteket hoztuk létre. A teszteket osztályonként a „GetAll” és az „Update” metódusokra írtuk meg. A „GetAllTest”-ben megvizsgáljuk, hogy helyes számban töltődnek-e be az adatok. A „Users” osztály esetében 3 teszt felhasználót vittünk fel. A teszt csak akkor fut le sikeresen, ha az elvárt értéket háromra állítjuk, más esetben sikertelen.

A képen szöveg látható

Automatikusan generált leírás

19. ábra „GetAllTest” teszt kódja

A képen szöveg, monitor, képernyő, fekete látható

Automatikusan generált leírás

20. ábra Sikeres tesztek

Mobil alkalmazás

# Mobil alkalmazás célja:

A telefonos alkalmazás célja, hogy a kutyakozmetikushoz a vendégek könnyebben és gyorsabban foglaljanak időpontot. Az applikációban tudnak regisztrálni majd bejelentkezés után tudják megadni a kis kedvenc adatait és időpontot foglalni.

# Mobil alkalmazás megvalósítása:

A telefonos alkalmazást Java nyelven írtuk meg az Android Studio fejlesztői környezet használatával.

## 1.Lépés

A teljesen üres projekt létrehozása után a főoldalon kezdtünk el dolgozni, a UserMenu-n. Először a layout-ot csináltuk meg melyben látható az applikáció neve, logója és két gomb az időpontfoglalás és a kutya adatainak regisztrálásához emellett egy GIF az oldal alján.

A képen szöveg látható

Automatikusan generált leírás

21. ábra UserMenu kinézete

Ezután a UserMenu.java fájlban írtuk meg a gombokra a navigálást, Intent használatával.

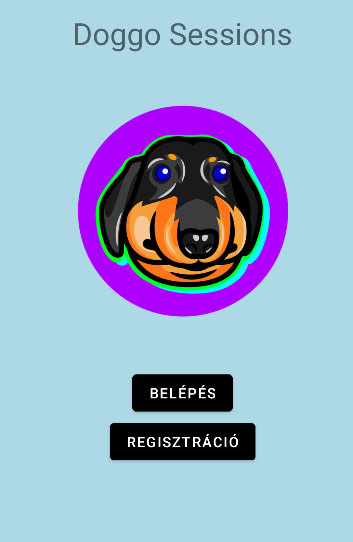
A képen szöveg látható

Automatikusan generált leírás

22. ábra UserMenu.Java

## 2.Lépés

Második lépésként a RegLogPage-et csináltuk. Ezen az oldalon található egy ”regisztráció” és egy ”bejelentkezés” gomb melyek újabb oldalakra navigálják a felhasználót.



23. ábra Reglog kinézete

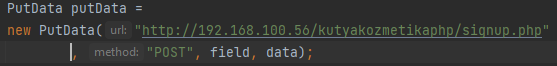
## 3.Lépés

A képen szöveg látható

Automatikusan generált leírásKövetkezőnek a Regisztráción kezdtünk el dolgozni. Előszőr létre hoztunk egy User-osztályt ahol felvettük a regisztrációhoz szükséges adattagokat majd elkészítettük a konstruktort, getter és settert.

24. ábra RegisterUser.Java

Ezután elkészítettük a RegisterUser osztályt. Ebben egy PutaDataToDB metódust írtunk melyben két tömböt hoztunk létre, az egyik az adatbázis felhasználó tábla oszlopai neveit kapta a másik pedig a User osztály-ból a getter segítségével kapja meg. A PutData beépített osztály segítségével pedig megadtuk az útvonalat a php-fájlhoz amely segítségével POST-olni tudjuk az adatokat az adatbázisba.



25. ábra PutData beépített osztály

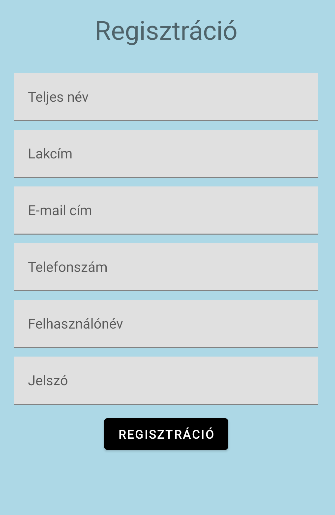
## 4.Lépés

A harmadik lépés után a Registration osztályt írtuk meg. Itt példányosítottuk a RegisterUser osztályt és a regisztrációs gomb OnClick eseménynél a User osztályt is. Majd a user objektumhoz kiolvassuk a felhasználó által megadott adatokat és a RegisterUser osztályban létrehozott putDataToDB metódussal feltöltjük az adatokat az adatbázisba.

A képen szöveg látható

Automatikusan generált leírás

26. ábra Registration.java

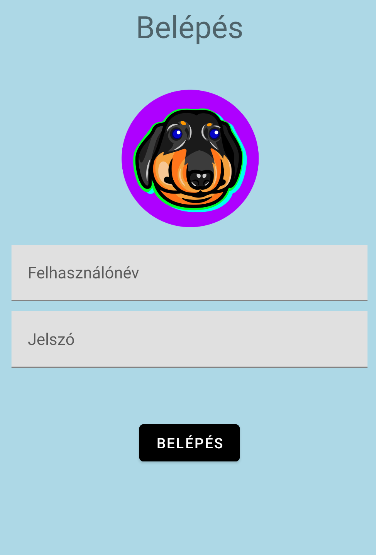


27. ábra Regisztrációs felület kinézete

## 5.Lépés

Következőnek a bejelentkezésen dolgoztunk. A bejelentkezés gombra kattintva ki olvassuk a felhasználó által megadott felhasználónevet és jelszót majd ezeket az adatokat PHP-ban ellenőrizzük, hogy létezik-e ilyen felhasználónév – jelszó páros, és ha létezik akkor a fő oldalra navigál az applikáció.

A bejelentkezés gombra kattintva a felhasználónevet tároljuk és mindig tovább adjuk a következő oldalnak az ”intent.putExtra”-val, ugyanis ezt a felhasználónevet fel használjuk az időpontfoglaláshoz.



28. ábra Belépés felület kinézete

## 6.Lépés

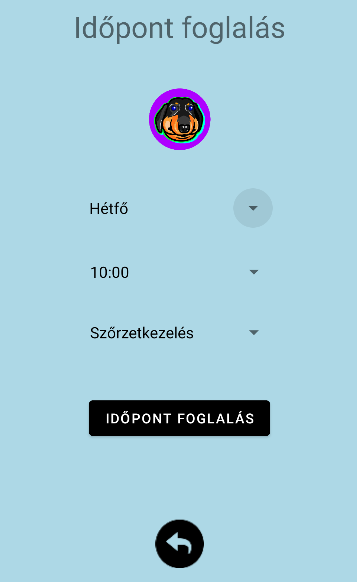
Hatodik lépésként a kutya adatainak regisztrálásán dolgoztunk. A kis kedvenc adatait bejelentkezés után tudják regisztrálni a felhasználók. Ezt a felhasználók regisztrálásához hasonló módon tettük meg. A különbség egy ImageButton ami arra szolgál, hogy a felhasználó vissza tudjon jutni a fő oldalra, ha esetleg félre kattintott volna.



29. ábra Kutya regisztrációs felület kinézete

## 7.Lépés

Utolsó lépésként az időpontfoglalást készítettük el. Az időpont foglalásnál Spinner-ek által tudjuk kiválasztani a kívánt napot és órát amikor el szeretnénk vinni. Végül a kívánt szolgáltatást tudjuk kiválasztani. A szolgáltatások nevét az adatbázisból kérdezzük le JSON formában majd Stringként megjelenítjük a Spinnerbe. Majd miután kiválasztottunk minden adatot az ”Időpont foglalás” gombra kattintva le is foglaltuk az időpontot.



30. ábra Időpontfoglalás felület kinézete

Backend a mobil alkalmazáshoz

A telefonos alkalmazáshoz a backendet PHP-ban írtuk meg, ez a backend arra szolgál, hogy az alkalmazás tudjon kommunikálni az adatbázisunkkal. Ezeket a PHP-fileokat a xamp/htdocs mappába kellett létrehozni, hogy el tudjuk érni őket az android studioból.

Először a DataBaseConfig.php-t írtuk meg ahol megadtuk melyik szerveren és adatbázisban dolgozunk :

A képen szöveg látható

Automatikusan generált leírás

31. ábra DataBaseConfig.php

Következőnek a DataBase.php-t csináltuk meg. Itt több függvényt írtunk meg az adatbázishoz való kapcsolódásra, regisztrációra, bejelentkezésre, időpont foglalásra és a szolgáltatások fetchelésére.

A bejelentkezésnél itt ellenőrizzük, hogy a beolvasott felhasználónév és jelszó páros létezik-e.

A képen szöveg látható

Automatikusan generált leírás

32. ábra Database.php-ban a login függvény

A függvények mellé további PHP fileokat hoztunk létre mint például login.php vagy szolgaltatasFetch.php.

Ezek a PHP fileok arra szolgálnak, hogy a felhasználó értesüljön a felmerülő hibákról. Például a bejelentkezésnél ki írja üzenetként, hogy belépés sikeres, ha nem sikeres akkor pedig hiba üzenetet ad vissza.

A képen szöveg látható

Automatikusan generált leírás

33. ábra login.php

Webes alkalmazás:

# Webes alkalmazás témája:

A webes alkalmazást egy kutyakozmetikának készítettük. A weblap kizárólag tájékoztatás szempontjából készült. A weblapon megtekinthető a kutyakozmetika neve, rövid leírás róla. Ezek mellett még megtalálható az oldalon a szolgáltatások neve és ára, valamint a kozmetikus címe és elérhetősége.

# Program megvalósítása

## 1.Lépés

Létrehoztuk a projektet Vue.js (egy javaScript könyvtár) keretrendszert és Vue Routert használva.

A projekt létrejötte után elkezdtünk dolgozni a főoldalon a Home.vue-ban.

Text

Description automatically generatedA Home.vue template része tartalmaz egy rövid leírást a kozmetikusról és egy Slideshow-t, ami a script részben megírt tömbből olvassa ki az adatokat és jeleníti meg. Az oldal kinézetéért a style-ban megírt kódok felelnek.

34. ábra Slideshow template-be megírt része

Text

Description automatically generated

35. ábra Slideshow script-ben megírt rész

## 2.lépés

A főoldal után a szolgáltatások oldallal foglalkoztunk, melynek a kódját a Szolgaltatasok.vue-ban írtuk meg. Ezen az oldalon kártyákban megjelenik a szolgáltatások neve és időtartama.

Graphical user interface, text

Description automatically generated

36. ábra Kártyák kódja a template-ben

A kártyákat bootstrap segítségével formáztuk és tettük reszponzívvá. A kártyákon megjelenő adatokat az adatbázisból olvassa ki a program egy axios lekérés segítségével. Ehhez importálnunk kellett az axios és bootstrap könyvtárakat.

Text

Description automatically generated

37. ábra axios lekérés a script-ben

## 3.Lépés

A szolgáltatások után az árlista oldallal foglalkoztunk, ahol a szolgáltatások neve és ára jelenik meg egy táblázatban. A szolgáltatásokat ezen az oldalon is egy axios lekérés segítségével jelenítjük meg a táblázatban (lásd 37. ábra).

Text

Description automatically generated

38. ábra Táblázat kódja a template-ben

## 4.Lépés

Az árak megjelenítése után következett a kapcsolat oldal, ahol megjelenik a kozmetikus elérhetősége és címe. A címe alatt megjelenik egy térkép, ami úgy van beállítva, hogy a kozmetikus címét mutassa alapértelmezetten.

Text

Description automatically generated

39. ábra Térkép kódja a template-ben

## 5.Lépés

Az App.vue-ban és index.js-ben beállítottuk, hogy a különböző oldalak között lehessen váltani.

A screenshot of a computer

Description automatically generated with medium confidence

40. ábra App.vue-ban megírt kódrészlet

Text

Description automatically generated

41. ábra Az index.js-ben ez a rész felel az oldalak közötti navigálásért

Web API a webes alkalmazáshoz

# Web API témája:

A Web API biztosítja a kapcsolatot az adatbázis és webes alkalmazás között.

# Web API megvalósítása:

## 1.Lépés

Elkészítettünk egy ASP.NET Core Web API projektet .NET 5.0-ás Frameworkot használva.

Graphical user interface, text, application, email

Description automatically generated

42. ábra Projekt létrehozása

## 2.Lépés

Telepítettük a szükséges Nuget csomagokat

Text

Description automatically generated

43. ábra Nuget csomagok

## 3.Lépés

Kiadtuk a Scaffold-DbContext parancsot a Package Manager Consolon:

*Scaffold-DbContext "server=localhost;user id=root;database=kutyakozmetika\_2019n" Pomelo.EntityFrameworkCore.MySql -OutputDir Models -DataAnnotations -UseDatabaseNames -Force -NoPluralize*

Ez a parancs olvassa be az adatbázisból a táblákat és hozza létre a Context osztályt.

Graphical user interface, text, application

Description automatically generated

44. ábra beolvasott táblák és Context osztály

## 4.Lépés

Az appsettings.json-ben ezután megírtuk a szükséges kapcsolatot a Web API és az adatbázis között.



45. ábra kapcsolat a Web API és adatbázis között

## 5.Lépés

A Startup.cs-ben szintén beállítottuk az adatbázis és Web API közti kommunikációhoz szükséges metódust. Illetve felülírtuk a CORS policyt, hogy a webes alkalmazás megtudja jeleníteni az adatokat.

Text

Description automatically generated

46. ábra kommunikációhoz szükséges metódus

Text

Description automatically generated

47. ábra CORS policy felülírása

## 6.Lépés

Létrehoztunk egy üres Controllert és megírtuk bele a szükséges lekérést.

Text

Description automatically generated

48. ábra Get lekérés

## 7.Lépés

A Web API megírása után a lekérést Google Chrome-ban a Rested bővítményen keresztül teszteltük, ahol sikeresen megjelent minden adat.

Graphical user interface, text, application

Description automatically generated

49. ábra Adatok sikeres lekérése

Adatbázis

# Tervezés

Egyedek:

* felhasznalo
* kutya
* megrendeles
* szolgaltatas

## Tulajdonságok:

felhasznalo (**felhasznaloID**, nev, cim, email, telefonszam, felhasznalonev, jelszo)

kutya (**kutyaID**, agresszivE, eletkor, fajta, tulajNev)

megrendeles (**megrendelesID**, foglalasNapja, foglalasOraja, felhasznalonev, szolgaltatasNev)

szolgaltatas (**szolgaltatasID**, szolgaltatasNev, idotartam)

# Egyedek és tulajdonságaik(adattípusok):

*felhasznalo*

**felhasznaloID**  szám (INT) – AUTO\_INCREMENT – elsődleges kulcs

nev szöveg (VARCHAR(100)) – a felhasználó neve

cim szöveg (VARCHAR(100)) – a felhasználó címe

email szöveg (VARCHAR(100)) – a felhasználó e-mail címe

telefonszam szöveg (VARCHAR(100)) – a felhasználó telefonszáma

felhasznalonev szöveg (VARCHAR(100)) – a felhasználó felhasználóneve

jelszo szöveg (VARCHAR(100)) – a felhasználó jelszava

*kutya*

**kutyaID**  szám (INT) – AUTO\_INCREMENT – elsődleges kulcs

agresszivE szöveg (VARCHAR(255)) – a kutya agresszív-e

eletkor szöveg (VARCHAR(255)) – a kutya életkora

fajta szöveg (VARCHAR(255)) – a kutya fajtája

tulajNev szöveg (VARCHAR(255)) – a kutya tulajának a neve

*megrendeles*

**megrendelesID** szám (INT) – AUTO\_INCREMENT – elsődleges kulcs

foglalasNapja szöveg (VARCHAR(15)) – a foglalás napja

foglalasOraja szöveg (VARCHAR(15)) – a foglalás órája

felhasznalonev szöveg (VARCHAR(255)) – a felhasználó felhasználóneve

szolgaltatasNev szöveg (VARCHAR(255)) – a szolgáltatás neve

*szolgaltatas*

**szolgaltatasID** szám (INT) – AUTO\_INCREMENT – elsődleges kulcs

szolgaltatasNev szöveg (VARCHAR(100)) – a szolgáltatás neve

idotartam szám (INT) – a szolgáltatás időtartama

# Az adatbázis E-K diagramja

Diagram

Description automatically generated

50. ábra E-K diagram

# Az adatbázis Bachmann-ábrája

Diagram

Description automatically generateds

51. ábra Bachmann-ábra