ZÁRÓDOLGOZAT

Miskolc

2024.

Wshop3 backend dokumentáció

Kurucz Attila

2024

Tartalomjegyzék

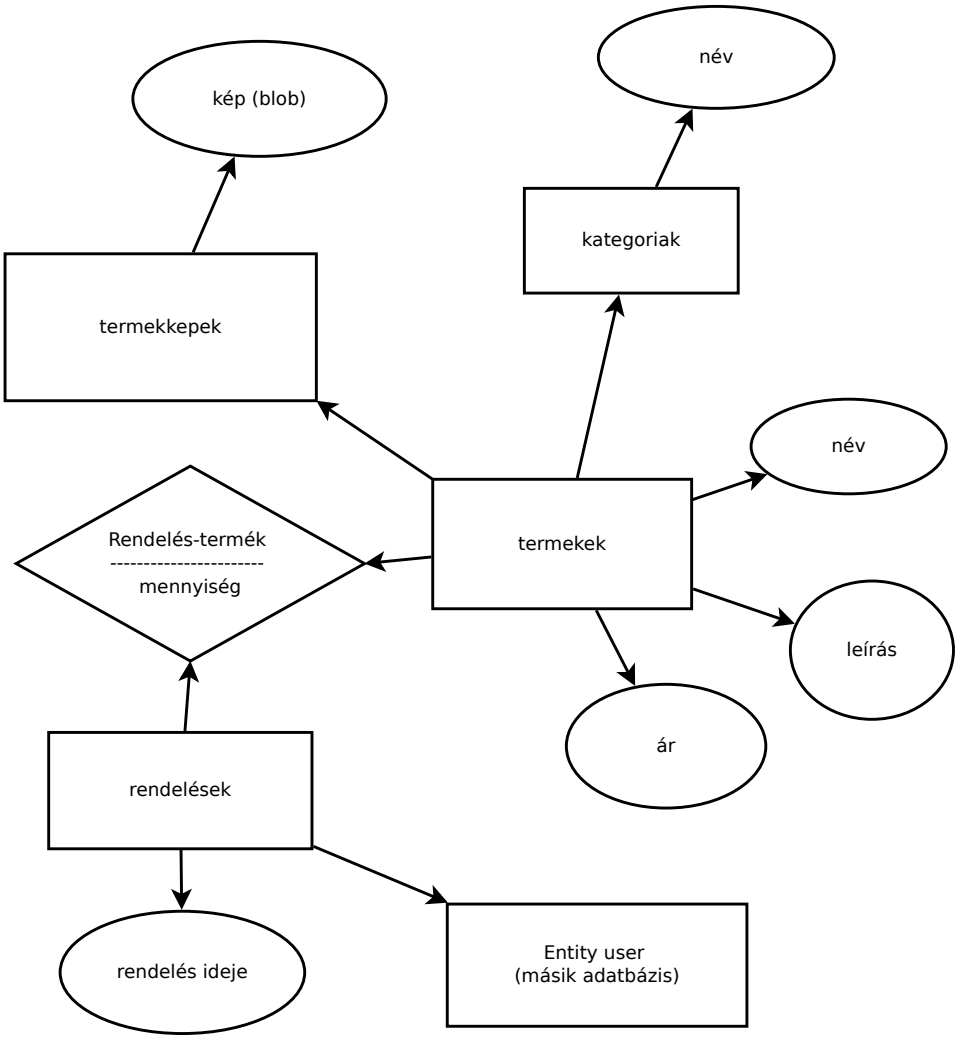
wshop3 backend dokumentáció

Bevezetés

A wshop3 egy italbolt, ami alkoholos italokat kínál megvásárlásra, úgy mint a különféle borok, sörök. A vásárláshoz regisztrálni szükséges, majd a megfelelő termékek kiválasztását követően, azok megrendelhetőek.

adabázis terv

Az adatbázis felépítése az ERD ábrán látható.



1. ábra: ERD

Felhasznált technológiák

Mariadb

2. ábra: mariadb logo

A MariaDB egy nyílt forráskódú relációs adatbázis-kezelő rendszer (RDBMS)[1], amelyet a MySQL eredeti fejlesztői hoztak létre, miután az Oracle megvásárolta a Sun Microsystems -t, és vele együtt többek között a MySQL-t is. A MariaDB kompatibilis a MySQL-lel, és célja, hogy ingyenes és nyílt forráskódú szoftver maradjon a GNU General Public License alatt.[2] A MariaDB megbízható, skálázható és biztonságos adatbázis-kezelő rendszer, amely széles körben használható. Nyílt forráskódú jellege, a MySQL-lel való kompatibilitása és számos funkciója miatt népszerű választás azok számára, akik adatbázis-kezelő rendszert keresnek.

A MariaDB számos funkcióval rendelkezik, amelyek javítják a MySQL teljesítményét és skálázhatóságát, beleértve

Gyorsabb lekérdezések: A MariaDB számos ponton tartalmaz javításokat, optimalizációkat, amelyek javítják a lekérdezések teljesítményét, különösen a összetett lekérdezések esetében.

Jobb skálázhatóság: A MariaDB bizonyos körülmények között jobban skálázható mint a MySQL, és nagy terhelések kezelésére is képes.

A MariaDB számos funkciója, valamint a mysql -el való kompatibilitása miatt népszerű választás a webes alkalmazásokhoz, az üzleti alkalmazásokhoz és a beágyazott rendszerekhez.

A MariaDB egy megbízható, skálázható és biztonságos adatbázis-kezelő rendszer, amely széles körben használható. Nyílt forráskódú jellege miatt népszerű választás azok számára, akik ingyenes és nyílt forráskódú megoldást keresnek. Használata különösen jellemző ott, ahol korábban a mysql -t alkalmaztak. Utóbbit az oracle továbbra is fejleszti, és támogatja.[3]

[1] https://mariadb.com/
[2] https://github.com/MariaDB/server
[3] https://www.mysql.com/

A MariaDB fő előnyei

Nyílt forráskódú: A MariaDB ingyenes és nyílt forráskódú szoftver, ami azt jelenti, hogy bárki használhatja, módosíthatja és terjesztheti a gpl2 -es licence feltételei alapján.[4] Ez népszerű választássá teszi azok számára, akik ingyenes és nyílt forráskódú megoldást keresnek.

Kompatibilis a MySQL-lel: A MySQL-alkalmazások könnyen átvihetők MariaDB-re. Ez megkönnyíti a MariaDB bevezetését a meglévő rendszerekbe.

Megbízható: A MariaDB egy megbízható adatbázis-kezelő rendszer, amelyet széles körben használnak a kritikus fontosságú alkalmazásokban.

Skálázható: A MariaDB egy skálázható adatbázis-kezelő rendszer, amely képes kezelni a kis és nagy terheléseket is.

A MariaDB felhasználási lehetőségei

Webes alkalmazások: A MariaDB népszerű választás a webes alkalmazásokhoz, mert skálázható, megbízható és kompatibilis a MySQL-lel.

Üzleti alkalmazások: A MariaDB népszerű választás az üzleti alkalmazásokhoz, mert skálázható, megbízható és biztonságos.

Beágyazott rendszerek: A MariaDB népszerű választás a beágyazott rendszerekhez, mert viszonylag alacsony teljesítményű rendszereken is képes működni, bár ezeken a területeken vannak hasznosabb és könnyebben alkalmazható megoldások, mint pl az sqlite, ami egy file alapú sql adatbázis kezelő rendszer, ami képes akár memóriában is létrehozni kissebb adatbázisokat.

Nagyon egyszerű példa Mariadb használatára Az ennyire egyszerű esetekre általában nincs is szükségünk adatbázisra - vagy nem ilyen formában - de az egyszerűség kedvéért tároljuk emberek listáját akiknek nevük, és koruk egy egy tulajdonságuk, valamint van címüket tároljuk külön adatszerkezetben. A két tábla létrehozható a kövekező parancsokkal.

Címek

```
CREATE TABLE cimek (
  cim_id INT PRIMARY KEY AUTO_INCREMENT,
  varos VARCHAR(255) NOT NULL,
  utca VARCHAR(255) NOT NULL,
  hazszam INT
);
```

[4] https://mariadb.com/kb/en/mariadb-licenses/

Emberek

```
CREATE TABLE emberek (
  felhasznalo_id INT PRIMARY KEY AUTO_INCREMENT,
  nev VARCHAR(255) NOT NULL,
  kor INT,
  cim_id INT NOT NULL,
  FOREIGN KEY (cim_id) REFERENCES cimek(cim_id)
);
```

Cimek felvétele

```
INSERT INTO cimek (varos, utca, hazszam) VALUES
("Miskolc", "Petőfi utca", 12),
("Budapest", "Rákóczi út", 22),
("Debrecen", "Kossuth Lajos utca", 33);
```

Emberek felvétele

```
INSERT INTO emberek (nev, kor, cim_id) VALUES
("Nagy Béla", 35, 1),
("Kovács Anna", 28, 2),
("Varga Péter", 42, 3),
("Szabó Ilona", 25, 1),
("Tóth János", 50, 2);
```

Összes ember címmel

```
SELECT e.felhasznalo_id, e.nev, e.kor, c.varos,
 c.utca, c.hazszam
FROM emberek e
JOIN cimek c ON e.cim_id = c.cim_id;
```

Egy ember (2 -es id -vel) címmel

```
SELECT e.felhasznalo_id, e.nev, e.kor, c.varos,
 c.utca, c.hazszam
FROM emberek e
JOIN cimek c ON e.cim_id = c.cim_id
WHERE e.felhasznalo_id = 2;
```

Emberek akik miskolcon laknak

```
SELECT e.felhasznalo_id, e.nev, e.kor, c.varos,
 c.utca, c.hazszam
FROM emberek e
JOIN cimek c ON e.cim_id = c.cim_id
WHERE c.varos = 'Miskolc';
```

Emberek akik idősebbek 40 évnél

```
SELECT e.felhasznalo_id, e.nev, e.kor, c.varos,
 c.utca, c.hazszam
FROM emberek e
JOIN cimek c ON e.cim_id = c.cim_id
WHERE e.kor > 40;
```

Cimek a lakosok neveivel Egy bonyolultabb lekérdezés, ahol a címeket választjuk ki, ahhoz illesztjük az emberek adatait, majd cím id alapján csoportosítunk, és a lakók neveit egy funkcióval (GRPUP_CONCAT[5]) csoportonként, vesszővel elválasztja adjuk hozzá.

```
SELECT c.cim_id, c.varos, c.utca, c.hazszam,
 GROUP_CONCAT(e.nev SEPARATOR ', ') AS lakók
FROM cimek c
LEFT JOIN emberek e ON c.cim_id = e.cim_id
GROUP BY c.cim_id;
```

Dotnet (8 lts)

3. ábra: .net logo

[5] https://mariadb.com/kb/en/group_concat/

A .NET 8 általános áttekintése A .NET 8[6] a Microsoft által fejlesztett nyílt forráskódú szoftverfejlesztési platform legújabb verziója. Ez a .NET 7, illetve core utódja, számos új funkcióval és fejlesztéssel. A .NET 8 2023 novemberében jelent meg, és azóta is aktívan fejlesztik. A .net eredetileg .net framework néven indult, a java programnyelvvel kapcsolatos jogvita egyik következményeként. Egészen a 4 -es vátozatig egy kizárólag windows operációs rendszereken működni képes platform/framework volt. Népszerű választás volt abban az esetben, ha kizárólag windows -ra készítettünk alkalmazásokat. A platformfüggőség mellett nagy probléma volt vele, hogy lényegében el volt rejtve a visual studio, egy kereskedelmi fejlesztőkörnyezet mögé. Technikai értelemben ugyan volt lehetőség enélkül is dotnetre fejleszteni, ám ez rendkívül körülményes volt.

Az alapvető változás akkor következett be, mikor a framework -öt újraimplementálták nyílt forrású alapokon, szélesebb körü platform támogatással, dotnet core néven. Az elnevezésből idővel levették a "core" szócskát, így ma már csak .net néven fejlesztik tovább. Nyílt forrású software -ként, valamint a platformfüggőség lazulása miatt, ma már a dotnet, és a programozásához leginkább használt programnyelv a c#, egy sokkal használhatóbb eszköznek tekinthető, számos területen vált alternatívává a már meglévő eszközök mellett. A nagy átalakulásnak azért voltak áldozatai is. A windows platform -hoz nagyon mélyen kötődő technológiák egy része így eltűnik, ilyen például a wpf[7] is, ami ugyan használható a mai napig, de csak akkor, ha a .net framework utolsó, 4.8 -as változatát használjuk hozzá.

A .NET 8 a következő platformokat támogatja

- Windows: A .NET 8 natív támogatást nyújt a Windows operációs rendszerhez.
- macOS: A .NET 8 natív támogatást nyújt a macOS operációs rendszerhez.
- Linux: A .NET 8 natív támogatást nyújt a Linux operációs rendszerhez.

Rövid példa c# programnyelv használatára

Egyszerű console alkalmazás létrehozása

```
user@debian:/mnt/temp$ dotnet new console -o teszt
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring /mnt/temp/teszt/teszt.csproj:
  Determining projects to restore...
  Restored /mnt/temp/teszt/teszt.csproj (in 122 ms).
Restore succeeded.
```

[6] https://dotnet.microsoft.com/en-us/
[7] https://learn.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-8.0

```
user@debian:/mnt/temp$ cd teszt/
user@debian:/mnt/temp/teszt$ cat Program.cs
// See https://aka.ms/new-console-template for more information
Console.WriteLine("Hello, World!");
user@debian:/mnt/temp/teszt$ dotnet run
Hello, World!
```

Egyszerű dotnet alkalmazás

```
public class Felhasznalo
{
    public string nev { get; set; } = string.Empty;
    public int kor { get; set; }
}

internal class Program
{
    private static void Main(string[] args)
    {
        var felhasznalok = new List<Felhasznalo>()
        {
            new Felhasznalo { nev = "Takacs Péter", kor = 32 },
            new Felhasznalo { nev = "Nagy Ágnes", kor = 27 },
            new Felhasznalo { nev = "Szabó István", kor = 45 },
            new Felhasznalo { nev = "Kis Zsuzsanna", kor = 19 }
        };

        var akitkeresek = "Zsuzsanna";

        //keressük meg az összes Zsuzsát
        foreach (var felhasznalo in felhasznalok)
        {
            if (felhasznalo.nev.Contains(akitkeresek))
            {
                //a dokumentumban való elhelyezés miatt
                //tördelve
                Console.WriteLine($"egy {akitkeresek}");
                Console.WriteLine($"{felhasznalo.kor} éves");
            }
        }
    }
}
```

React-js

4. ábra: react icon

A React[8] egy népszerű JavaScript könyvtár, amelyet felhasználói felületek (UI) létrehozására használnak. Komponensen alapú megközelítést alkalmaz, ami azt jelenti, hogy a komplex felületeket kisebb, újrafelhasználható kódtömbökre, komponensekre bontja. Ez a megközelítés megkönnyíti az összetett felhasználói felületek kezelését és karbantartását. Ha interaktív webes alkalmazásokat szeretne fejleszteni, akkor a React kiváló választás lehet.

React néhány általános jellemzői

Komponensek: A React alkalmazások komponensekből épülnek fel. Minden komponens egy JavaScript függvény, amely leírja, hogyan jelenjen meg az UI egy bizonyos része.

JSX: A React JSX egy egyszerűsített javascript formátum, ami arra szolgál, hogy megkönnyítse a html komponensek leírását, így nem szükséges nagy mennyiségű string -ként gondolni rájuk, azt a fordító fogja létrehozni amikor elkészítjük az oldalt.

Virtuális DOM: A React virtuális DOM-ot használ a UI frissítésének optimalizálására. Amikor az adatok megváltoznak az alkalmazásban, a React összehasonlítja a virtuális DOM aktuális állapotát a kívánt állapottal, és csak a szükséges módosításokat hajtja végre a valódi DOM-on. Ez jelentősen javíthatja az alkalmazás teljesítményét.

A React előnyei

Könnyen tanulható: A React viszonylag könnyen tanulható azok számára, akik már rendelkeznek alapvető JavaScript ismeretekkel.

Újrafelhasználható komponensek: A komponensek újrafelhasználhatósága miatt a React nagyszerűbben használható összetett felhasználói felületek létrehozásához.

[8] React

A React hátrányai

Komplexebb alkalmazások elkészítése bonyolultabb lehet: Bár a React alapjai könnyen elsajátíthatók, a komplexebb alkalmazásokhoz további könyvtárak és eszközök elsajátítása válhat szükségessé.

Verzió függőség: Nagy mennyiségű külső komponens/könyvtár használata esetén gyakran szükség lehet a meglévő kód módosítására, hogy kompatibilis maradjon a saját alkalmazásunk, a használt könyvtárakkal.

Egyszerű react példa

```
import React, { Component } from 'react';
import './App.css';

function Hello(adat) {
  return <h1>Szervusz {adat.Nev}</h1>
}

function Bekezdes(adat) {
  return <p>{adat.darab + 1}. bekezdés.</p>
}

function Bekezdesek(adat) {
  return Array(adat.darab).fill().map((_, i)
   => <Bekezdes darab={i} />)
}

export default function Alap(params) {
  return (
    <div>
      <Hello Nev="Bor Virág" />
      <Bekezdesek darab={2} />
    </div>
  )
}
```

HTML

A HTML[9] (HyperText Markup Language) egy leíró nyelv, melyet weboldalak készítéséhez fejlesztettek ki, és mára már internetes szabvánnyá vált a W3C (World Wide Web Consortium) támogatásával. A HTML nem programozási nyelv, hanem egy jelölőnyelv, amely parancsokat és utasításokat ad a böngészőnek a weboldal megjelenítésével kapcsolatban.

[9] https://html.spec.whatwg.org/

5. ábra: html logo

A HTML alapvető elemei

Dokumentumtípus-deklaráció: Ez az elem megadja a böngészőnek, hogy milyen típusú HTML-dokumentummal van dolga.

HTML elem: A HTML elemek építőkövei a weboldalaknak. Minden elemnek van egy nyitó címkéje (), egy záró címkéje () és tartalma. Illetve bizonyos elemek használhatóak önlezáró formában is. ()

Attribútumok: Az attribútumok további információkat adnak meg egy HTML elem tulajdonságairól. Az attribútumokat a nyitó címke után, név-érték párok formájában kell megadni.

Példa egy egyszerű HTML dokumentumra

```
<!DOCTYPE html>
<html>
<head>
  <title>A cím</title>
</head>
<body>
  <h1>Helló</h1>
  <p>Egy bekezdés</p>
</body>
</html>
```

A HTML használatának előnyei

Könnyen tanulható: A HTML viszonylag egyszerű nyelv, amelyet bárki megtanulhat alapvető számítógépes ismeretekkel.

Széles körben támogatott: A HTML-t minden modern böngésző támogatja.

A HTML hátrányai

Nem interaktív: A HTML önmagában nem képes interaktív elemek létrehozására - eredetileg ez nem is volt célja -, mint például űrlapok vagy animációk. Ehhez további technológiákra használatára lehet szükség, mint például JavaScript vagy CSS.

Nehézkes bonyolult elrendezések esetén: A HTML bonyolult elrendezések létrehozásakor nehezen kezelhetővé válhat.

CSS

6. ábra: css logo

A CSS[10] (Cascading Style Sheets) egy stílusleíró nyelv, amely a weboldalak megjelenését határozza meg. A HTML a weboldal tartalmát és szerkezetét definiálja, míg a CSS a weboldal vizuális megjelenítését szabályozza.

A CSS főbb jellemzői

Szelektorok: A CSS szelektorok segítségével határozza meg, hogy melyik HTML elemet kell a stílusokkal formázni. A szelektorok lehetnek azonosítók (#id), osztályok (.class), elemek nevei (p), vagy ezek kombinációi.

Deklarációk: A deklarációk megadják a stílus tulajdonságát és értékét. Például: color: red; font-size: 16px;.

Stíluslapok: A CSS stílusokat külön fájlban (.css kiterjesztés) is tárolhatjuk, és több HTML dokumentumhoz is csatolhatjuk. Ez megkönnyíti a stílusok karbantartását és újrafelhasználását.

[10] https://www.w3.org/TR/CSS2/

Egyszerű példa css használatára

```
/* A címek formázása */
h1 {
  color: blue; /* Kék színűre állítja a címeket */
  font-size: 2em; /* A betűméretet 2-szeresére növeli */
}

/* A bekezdések formázása */
p {
  font-family: Arial, sans-serif; /* Betűtípus beállítása */
  line-height: 1.5; /* Sorok közötti távolság növelése */
}

/* A linkek formázása */
a {
  color: green; /* Zöld színűre állítja a linkeket */
  text-decoration: underline; /* Aláhúzza a linkeket */
}

/* A linkre vitt egér hover hatás */
a:hover {
  color: darkgreen; /* Sötétebb zöld lesz a link
   egérrel ráállva */
}
```

A CSS használatának előnyei

Segítségével elválasztjuk egymástól a megjelenítést és a tartalmat leíró jelöléseket: A HTML a tartalomra, a CSS pedig a megjelenítésre fókuszál. Ez a szétválasztás javítja a kód olvashatóságát és karbantarthatóságát a korábban alkalmazott, a formátumot xml tulajdonságként leíró megoldáshoz képest. Mindazonáltal, továbbra is alkalmazhatóak önállóan is formátumleíró tulajdonságok a html elemekre.

Újrafelhasználható stílusok: A CSS stílusokat több HTML dokumentumhoz is csatolhatjuk, így időt takaríthatunk meg, és egységes megjelenítést biztosíthatunk.

Rugalmasság: A CSS számos tulajdonságot kínál a szöveg, a háttér, a szegélyek, a margók, a betűtípusok és még sok más formázásához.

A CSS hátrányai

Bonyolult szelektorok: A bonyolult szelektorok nehezen olvashatóvá és karbantarthatóvá tehetik a CSS kódot.

Böngésző-inkompatibilitások: Bár a CSS szabványosított nyelv, előfordulhatnak kisebb különbségek a különböző böngészők megvalósításában.

bootstrap

7. ábra: bootstrap logo

A Bootstrap[11] egy nyílt forráskódú CSS keretrendszer, amelyet a weboldalak és webes alkalmazások gyors és egyszerű fejlesztésére használhatunk. A legtöbb általános esetre tartalmaz megfelelő css osztályokat, amik vizuális megjelenésükben jól illeszkednek egymáshoz.

A Bootstrap főbb jellemzői

Responzív kialakítás: A Bootstrap automatikusan responzívvá teheti a weboldalakat, így azok minden képernyőméreten (asztali gépek, táblagépek, mobiltelefonok) optimálisan jeleníthetőek meg.

Elsősorban mobil eszközökre optimalizált: A Bootstrap feljesztésében tehát a mobil nézet a elsődleges, de az asztali gépeken további stílusokkal finomhangolható.

Könnyen használható: A Bootstrap könnyen tanulható és használható, még a kezdők számára is, hiszen lényegében css osztályok gyűjteménye, amik önállóan is alkalmazhatóak, vagy kombinálhatóak a feladatnak megfelelően.

Nyílt forráskódú: A Bootstrap nyílt forráskódú, így ingyenesen használhatja és módosítható.

A Bootstrap használatának előnyei

Gyorsabb fejlesztés: A Bootstrap előre elkészített komponensei segítségével gyorsabban hozhat létre weboldalakat és webes alkalmazásokat.

Konzisztens megjelenés: A Bootstrap egységes megjelenítést biztosít a weboldalakon.

Responzív weboldalak: A Bootstrap megfelelő osztályok használata esetén responzívvá teheti a weboldalakat, így minden képernyőméreten jól néznek ki.

Könnyen karbantartható: A Bootstrap kódja könnyen olvasható és karbantartható.

Nagy közösség: A Bootstrapnak nagy és aktív közössége van, így sok segítséget és támogatást kaphat.

[11] https://getbootstrap.com/

A Bootstrap hátrányai

Kevésbé testreszabható: A Bootstrap előre elkészített komponensei nem mindig illeszkednek tökéletesen az egyéni igényekhez, ugyanakkor könnyedén kombinálható más css könyvtárakkal, egyéni stílusokkal.

Néhány fontosabb bootstrap osztály

container: Középre igazítja a tartalmat.
row: Sorokat hoz létre.
col-*: Oszlopokat hoz létre (például: col-md-6 hat oszlopot hoz létre közepes méretű képernyőkön).
offset-*: Eltolja a tartalmat az oszlopokban (például: offset-md-3 3 oszloppal tolja el a tartalmat jobbra).
margin-*: Külső margókat ad hozzá az elemekhez.
padding-*: Belső kitöltést ad hozzá az elemekhez.
text-*: Beállítja a szöveg színét (például: text-primary).
bg-*: Beállítja a háttér színét (például: bg-primary).
btn: Gombstílusokat alkalmaz.
img-fluid: Automatikusan átméretezi a képet a konténerhez.

Entity framework

Az Entity Framework[12] (EF) core egy Object Relational Mapper (ORM) keretrendszer, amely a Microsoft .NET platformhoz készült. Az ORM lényege, hogy a relációs adatbázisok és az objektumorientált programozási nyelvek (mint például a C#) közötti kommunikációt egyszerűsítse. Teszi ezt oly módon, hogy az adott nyelven használható struktúrák alapján készíti el az sql nyelvű lekérdezéseket, így az adatok tárolása és lekérdezése a programozó számára részleteiben rejtve marad.

Az EF főbb jellemzői

Objektum-relációs leképezés: Az EF lehetővé teszi, hogy az adatbázis tábláit és oszlopait objektumokként és tulajdonságokként kezeljük a programunkban.
Adatlekérdezés LINQ segítségével: Az EF a LINQ (Language Integrated Query) technológiát használja az adatlekérdezések megfogalmazására. A LINQ segítségével a programozási nyelv szintaxisát használhatja az adatbázis lekérdezések írására, ami egyszerűbbé teszi a folyamatot.
Adatbázis-migráció: Az EF segít az adatbázis séma módosításainak kezelésében. Az adatséma módosításait (például új táblák hozzáadása, oszlopok módosítása) kódba lehet írni, és az EF ezeket a módosításokat alkalmazza az adatbázison.

[12] https://learn.microsoft.com/en-us/ef/core/

Az Entity Framework előnyei

Fejlesztés egyszerűsítése: Az ORM megközelítés felgyorsíthatja a fejlesztést, mivel kevesebb kód szükséges az adatbázis eléréséhez és kezeléséhez.

Egyszerűbb kód: Az objektumorientált megközelítés tisztább és karbantarthatóbb kódot eredményezhet.

Kevesebb hiba: Az EF automatizálja az SQL lekérdezések létrehozását, ami csökkenti a kézi kódolásból eredő hibák lehetőségét. Ugyanakkor jellegéből adódóan más hibákkal találkozhatunk, ha nem pontosan azt csinálja a kód, amit gondolunk róla.

Az Entity Framework hátrányai

Teljesítmény: Általában az ORM -ek lassabbak, mint a közvetlen sql lekérdezések, hiszen előre meghatározott szabályok mentén először fel kell hogy építsék a lekérdezéseket, ami plussz erőforrást igényelnek.

Rugalmasság: Az EF és más hasonló ORM -ek egyik problémája, hogy míg az egyszerű lekérdezések létrehozását megkönnyítik, addig komplex feladatok elvégzését akár meg is nehezíthetik, pl mikor többszörösen kapcsolt adatszerkezetet szeretnénk szűrni és létrehozni, akkor nem biztos hogy hamarabb célba érünk az EF -el, mint ha megírjuk a lekérdezést kézzel. De az is jelenthet gondot, ha az adatbázis szerver által kínált egyedi funkciókat szeretnénk használni. Az ilyen esetekre a legtöbb ORM, így az ef is kínál lehetőséget arra, hogy saját api -ján keresztül adjunk meg neki kézzel írt sql lekérdezéseket.[13]

Identity Framework

Az Identity Framework (ASP.NET Identity) a Microsoft ASP.NET Core keretrendszerhez használható felhasználókezelési szolgáltatásokat nyújtani képes framework, melyek webes alkalmazásokhoz lehetnek hasznosak. Segítségével kezelheti a felhasználók regisztrációját, bejelentkezését, szerepköreit, és jogosultságait.

Az Identity Framework főbb jellemzői

Felhasználókezelés: Lehetővé teszi felhasználói fiókok létrehozását, törlését, és módosítását.

Hitelesítés: Támogatja a különböző hitelesítési mechanizmusokat, mint például a helyi fiókokkal, külső szolgáltatókkal (pl. Google, Facebook) történő bejelentkezést, vagy kétfaktoros hitelesítést.

Szerepkör-alapú hozzáférés-vezérlés: Segít meghatározni a felhasználói szerepköröket és a hozzájuk tartozó engedélyeket, ezáltal biztosítva, hogy a felhasználók csak a számukra engedélyezett funkciókat és adatokat érhessék el.

[13] https://learn.microsoft.com/en-us/ef/core/querying/sql-queries

Jelszókezelés: Biztonságos jelszókezelési mechanizmusokat biztosít, beleértve a jelszó erősségének ellenőrzését és a jelszó-visszaállítási funkciót.

Token alapú hitelesítés: Támogatja a token alapú hitelesítést, amely API-k és mobilalkalmazások biztonságos hitelesítéséhez használható.

Az Identity Framework előnyei:

Beépített funkciók: Az Identity Framework számos beépített funkciót kínál a felhasználókezeléshez, így nem kell külön könyvtárakat hozzáadnia a projekthez, leszámítva persze magát az Identity keretrendszert.

Rugalmasság: Az Identity Framework testre szabható, így az alkalmazás egyedi igényeihez igazítható.

Biztonság: Az Identity Framework biztonságos mechanizmusokat kínál a felhasználói adatok kezeléséhez és a jogosulatlan hozzáférés megakadályozásához.

Az Identity Framework hátrányai:

Tanulás: Az Identity Framework használatához meg kell tanulni a keretrendszer működését és a konfigurációs lehetőségeket, melyekből rendkívül sok van, és verziónként is vannak hatalmas eltérések. A rendszer az idők során egyszerűsödött valamelyest.

bonyolultabb alkalmazásokhoz: Nagyobb, összetettebb alkalmazások esetében az Identity Framework konfigurációja bonyolultabbá válhat.

curl[14]

8. ábra: curl logo

A curl egy rendkívül hasznos eszköz, képességei közé tartozik a rendkívül nagy számú hálózati protocol kezelésének képessége. Jelen dokumentáció azért említi, mert webes api végpont teszteléshez is ez az egyik legkiválóbb eszköz. Segítségével automatizálható a tesztelési folyamat egy része. Néhány shell script segítségével jelentősen könnyebbé válhat a végpontok tesztelése.

[14] https://curl.se/

git[15]

9. ábra: git logo

A git egy verziókezelő rendszer. Erre a feladatra számtalan programot készítettek már a git előtt is, és azóta is készülnek újabb és újabb próbálkozások. Ezeknek a programoknak az a céljuk, hogy - leginkább a programkódok esetében - követni tudjuk a munkát amit végzünk, könnyebben fel tudjuk használni, és ki tudjuk javítani a mások által írt kódokat, és adott esetben vissza tudjunk térni bármelyik korábbi, működő változathoz.

Ezek a programok természetesen nem csak programkódokhoz használhatóak, ám a leghatékonyabban, minden funkciójukat csak akkor tudjuk kihasználni, hogy ha leginkább text file -okat tárolunk bennük, mert ezek esetében tudjuk megfelelően hasznosítani a verziók közti különbségek követését, és ezek esetében a leghatékonyabbak a tároló formátumok is, amiket ezek a programok használnak.

A git a példának okáért - mint oly sok más esetben - ezen dokumentum elkészítésében is hasznomra volt.

A git születése: A git -et eredetileg Linus Torvalds készítette saját projectje, a linux kernel fejlesztésének megkönnyítésére. Ezt megelőzően egy BitKeeper néven ismert, kereskedelmi programot alkalmazott erre a célja, melynek fejlesztője megengedte hogy a linux kernelhez ingyenesen használhassák a programot a fejlesztők. Később egy hosszadalmas vita kapcsán ezt a lehetőséget elvesztették, és szükség volt valamire ami képes pótolni ezt a programot. Linus nem rajongott az ingyenesen elérhető régebbi programokért, ezért úgy döntött készít egy sajátot megoldást. A kezdetben rendkívül körülményesen használható program később hatalmas népszerűségre tett szert, idő közben maga alá temetve sok korábban ismert és kedvelt scm programot, köztük magát a BitKeeper -t is, amin nem segített már az sem, hogy évekkel később nyílt forrású programként tették közzé.

[15] https://git-scm.com/

A git néhány versenytársa:

Mercurial[16]: A git mellett érdemes megemlíteni egy másik programot, amely szintén a BitKeeper pótlására született, ez a mai napig fejlesztett, és számos nagy cég álltal alkalmazott Mercurial nevű scm volt. Ez utóbbi is el tudta volna látni a feladatot amire készült, ám Linus úgy gondolta, hogy túlságosan hasonló a BitKeeper -hez, így végül a választás a git -re esett.

Fossil[17]: A fossil -t a jól ismert és kedvelt file alapú adatbázis, az sqlite vezető fejlesztője hozta létre, hogy hogy megkönnyítse annak fejlesztését. A fossil nem csak az - elosztott - verziókezelést kívánja megoldani, hanem a project management, a bug tracker, és a hosting feladatokat is, egyetlen binálisba fordított c program formájában. Free, és opensource akár csak a git.

Pijul[18]: A Pijul a Darcs verziókezelő utódjának tekinthető, célja hogy annak hibáit, leginkább a teljesítményproblémákat orvosolja. Nem branch -ekkel dolgozik, hanem csak módosításokkal. Nagyon leegyszerűsítve, kicsit hasonlít ahhoz, mint ha git -nél minden esetben csak cherry pick -et használhatnánk. Git -hez hasonló branch rendszert nem alkalmaz.

A git előnyei:

Teljesítmény: A git gyors. Nagyon gyors. Ez volt az egyik legfontosabb követelmény a fejlesztés során. Így a commit, és merge folyamata nem hátráltatja a fejlesztőt.

Branching: A git branch rendszere nagyon flexibilis, bármilyen munkafolyamathoz hozzá lehet igazítani. Lehetnek helyi és távoli branch -ek, mindig mi döntjük el hogy melyik fejlesztési folyamatot szeretnénk megosztani másokkal, és melyiket szeretnénk megtartani magunknak, például mert az adott megoldás még nincs megfelelő minőségi állapotban.

Rugalmas: Egy repository tartalmával lényegében azt teszünk, amit csak akarunk. a commit fát átírhatjuk, javíthatjuk, módosíthatjuk, több commit -ot egybe rakhatunk, szét szedhetünk stb. Egészen addig, amíg nem osztottuk meg a branch -ünket senkivel, minden fajta negatív következmény nélkül csinálhatunk vele lényegében bármit. A git pedig aktívan támogat minket ebben. Ezért - talán kis túlzással - mondhatjuk azt is, hogy a git nem csak egy verziókezelő rendszer, hanem egy hasznos fejlesztőeszköz is.

Staging: A Staging azt jelenti, hogy a git lényegében engedi hogy elkészítsük a commitunk -at, mielőtt az bekerülne a commit fába. Minden módosításunk először ide kerül. Ezt sokan negatívumként említik, ám ez teremti meg a lehetőséget az olyan funkciókhoz, mint például a részmódosítások hozzáadása, külön commitként elkészítve azokat, ahogy a logika épp megköveteli.

Mindent helyre lehet hozni: Mivel a git működéséből adódóan egyetlen parancs kivételével mindig csak hozzáad a repo -hoz, ezért bármit is teszünk, egy ideig még biztosan megtaláljuk minden commitunkat a reflog -ban, ahonnan vissza is tudjuk hozni azt egy új branch -be, vagy akár a régibe.

A git hátrányai:

Könnyen tönkre lehet tenni: Mivel a git szinte végtelen szabadságot biztosít a repository tartalmával kapcsolatban, ha valaki nem tudja mit csinál, jobb ha megmarad a legalapvetőbb utasításoknál.

Felhasználói felület: Nagyon sokáig jogos érv volt a git ellen, hogy a kezelése rendkívül bonyolult volt. Az egyes kapcsolók és lehetőségek nem voltak teljesen egyértelműek, így sokan úgy élték meg az első találkozást, hogy a git egy végtelenül túlbonyolított, haszontalan eszköz. Manapság ezeknek az ellenérveknek már csak kis része védhető érvekkel.

Egyszerű példa a git használatára A következő nagyon egyszerű példában létrehozunk egy repository -t, majd egy aa.txt nevű állományt. Ezek után két sort írunk bele, mindkét módosítást egy egy commit -ban rögzítve. Ezután a git log -ban megnézzük mi történt eddig a repository -ban. Létrehozunk egy új branch -et teszt néven, két új sort két commitként rögzítünk. Visszaváltunk a master branch -re, és beolvasztjuk a teszt -ben található módosításokat. Ezután töröljük a teszt branch -et. Ebben az esetben nem jött létre merge commit, mert közvetlenül levezethetőek voltak a módosítások a master és a teszt branch változásai alapján.

```
user@debian:/mnt/temp$ git init
Initialized empty Git repository in /mnt/temp/.git/
user@debian:/mnt/temp$ touch aa.txt
user@debian:/mnt/temp$ echo 'első sor' >> aa.txt
user@debian:/mnt/temp$ git add .
user@debian:/mnt/temp$ git commit -m "első sor"
[master (root-commit) 8fa7b83] első sor
 1 file changed, 1 insertion(+)
 create mode 100644 aa.txt
user@debian:/mnt/temp$ echo 'második sor' >> aa.txt
user@debian:/mnt/temp$ git commit -am "második sor"
[master 008dd3d] második sor
 1 file changed, 1 insertion(+)
user@debian:/mnt/temp$ git lk
* 008dd3d (HEAD -> master) második sor
* 8fa7b83 első sor
user@debian:/mnt/temp$ git checkout -b teszt
Switched to a new branch 'teszt'
user@debian:/mnt/temp$ echo 'harmadik sor' >> aa.txt
user@debian:/mnt/temp$ git commit -am "harmadik sor"
[teszt 8004464] harmadik sor
 1 file changed, 1 insertion(+)
user@debian:/mnt/temp$ echo 'negyedik sor' >> aa.txt
user@debian:/mnt/temp$ git commit -am "negyedik sok"
[teszt 93d8491] negyedik sok
```

```
 1 file changed, 1 insertion(+)
user@debian:/mnt/temp$ git lk
* 93d8491 (HEAD -> teszt) negyedik sok
* 8004464 harmadik sor
* 008dd3d (master) második sor
* 8fa7b83 első sor
user@debian:/mnt/temp$ git checkout master
Switched to branch 'master'
user@debian:/mnt/temp$ git merge teszt
Updating 008dd3d..93d8491
Fast-forward
 aa.txt | 2 ++
 1 file changed, 2 insertions(+)
user@debian:/mnt/temp$ git branch -d teszt
Deleted branch teszt (was 93d8491).
user@debian:/mnt/temp$ git lk
* 93d8491 (HEAD -> master) negyedik sok
* 8004464 harmadik sor
* 008dd3d második sor
* 8fa7b83 első sor
user@debian:/mnt/temp$
```

Backend api végpontok

Auth

Az Auth végpont a felhasználók regisztrációjára, azonosítására, és beléptetésére szolgálnak. A program jelenlegi változatában az Identity framework végzi el a feladatot, melynek adattábláit külön adatbázisban helyeztem el.

/auth/register A felhasználó regisztrációja a következő adatok megadásával lehetséges. Jelenleg semmilyen nem szükséges semmilyen egyéb azonosítási forma az egyszerűség kedvéért, de valós körülmények között természetesen email azonosításra volna szükség, esetleg captcha, vagy más biztonsági megoldással kiegészítve. Ezen kívül a kiegészítő adatok is átkerülnének egy külön oldalra, mint pl a lakcímhez tartozó adatok.

```
curl -X 'POST' \
  'szerver:1234/auth/register' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "userName": "string",
```

```
  "password": "string",
  "email": "string",
  "teljesNev": "string",
  "iranyitoszam": 0,
  "varos": "string",
  "utca": "string",
  "hazszam": 0
}'
```

/auth/AssignRole

```
curl -X 'POST' \
  'szerver:1234/auth/AssignRole' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "role": "string",
  "email": "string"
}'
```

Jogkörök felhasználóhoz rendelésére használható végpont. Minden felhasználó kap a regisztráció alkalmával egy “USER” role -t, ami a felhasználáshoz szükséges legfontosabb végpontok elérését biztosítja. Ezen kívül a jelenlegi kódban használok még “ADMIN” role -t, ami az adminisztrációs feladatok elvégzéséhez szükség jogkör. A lista szükség esetén tovább bővíthető.

/auth/login

```
curl -X 'POST' \
  'szerver:1234/auth/login' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "userName": "string",
  "password": "string"
}'
```

A felhasználó beléptetésére szolgál. A beléptetés során a felhasználónév, valamint a jelszó ellenőrzésre kerül, és amennyiben a felhasználó adatai megfelelőek, úgy a beléptetés során jwt token kerül létrehozásra, amit visszaküldök a kliensnek. Hibás adat esetén üres token érkezik majd.

```
{
  "token": ""
}
```

Valós körülmények között ez nem volna biztonságos megoldás, hiszen a jwt token minden adatot tartalmaz a felhasználóról, így ha ezt a kliens oldal nem kezeli körültekintően, úgy biztonsági problémákat okozhat. Sok esetben jobb megoldás lehet az egyszerű session cookie (ha nincs kezelhetetlenül nagy terhelés, akkor így tartható minden adat a server -en is, és a felhasználó csak a session cookie -t kapja), esetleg jwt -vel combinálva, vagy pl a dupla token -es megoldás, ahol az azonosításhoz használt jwt token nem tartalmaz minden adatot, azt csak azonosítás után kaphatná meg a felhasználó.

Felhasznalok

A program jelenlegi változatában az itt elérhető végpontok nincsenek használatban, csak egyszerű tesztelési célokat szolgáltak.

Kategoriak

A termékek kategóriáinak kezelésére használható végpontok. Minden terméknek kötelezően kell hogy legyen kategória besorolása is, így amennyiben a kívánt kategória nem létezik, azt szükséges először rögzíteni.

/api/Kategoriak/KategoriaId

```
curl -X 'GET' \
  'szerver:1234/api/Kategoriak/KategoriaId/1' \
  -H 'accept: */*'
```

Id alapján lehet az egyes kategóriákat lekérdezni.

/api/Kategoriak/OsszesKategoria

```
curl -X 'GET' \
  'szerver:1234/api/Kategoriak/OsszesKategoria' \
  -H 'accept: */*'
```

Ezen a végponton az összes kategória lekérdezése lehetséges.

/api/Kategoriak/kategoriaUJ

```
curl -X 'POST' \
  'szerver:1234/api/Kategoriak/kategoriaUJ' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "nev": "string"
}'
```

Új kategória rögzítése.

Rendelesek

Ezen a végponton a rendelések kezelése történik. Rendelések leadása történhet a frontenden pl egy kosár implementálásával. A rendelések módosítása a jelenlegi változatban nem lehetséges, valamint nem tárol olyan adatokat sem, amik a termékhez tartoznak. (tehát pl a jelenlegi változatban nem kezelhető vele az árak változása, hiszen az árak a termékhez tartoznak) A megrendelt termékek mennyiségét egy kapcsoló tábla tárolja, a megrendelő ID -jával, valamint a darabszámmal együtt.

/api/Rendelesek/RendelesId

```
curl -X 'GET' \
  'szerver:1234/api/Rendelesek/RendelesId/1' \
  -H 'accept: */*'
```

Rendelés lekérdezése Id alapján.

/api/Rendelesek/OsszesRendeles

```
curl -X 'GET' \
  'szerver:1234/api/Rendelesek/OsszesRendeles' \
  -H 'accept: */*'
```

Ezen a végponton az összes rendelés lekérdezése lehetséges.

/api/Rendelesek/RendelesekUj

```
curl -X 'POST' \
  'szerver:1234/api/Rendelesek/RendelesekUj' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "felhasznaloId": "string",
  "termekek": [
    {
      "id": 0,
      "mennyiseg": 0
    }
  ]
}'
```

Új rendelés leadásakor használható végpont. A backend -en több azonos termék egyidejű megrendelése esetén, a termék csak egyszer kerül tárolásra, a termékek darabszáma pedig összeadódik. Csak regisztrált felhasználó tud rendelést leadni.

Termekek

A termékek kezelésére használható végpontok. A termékek lekérdezése minden esetben a hozzá tartozó fényképek lekérdezését is jelenti, ami base64 formában érkezik a kliensre.

/api/Termekek/

```
curl -X 'GET' \
  'szerver:1234/api/Termekek/1' \
  -H 'accept: */*'
```

Termék lekérdezése ID alapján.

/api/TermekLista

```
curl -X 'GET' \
  'szerver:1234/api/Termekek/TermekLista?Rendez=nev \
  &CsokkenoSorrend=true&Keres=bor&Lapszam=2&Lapmeret=10' \
  -H 'accept: */*'
```

A termékek lekérdezését lehet elvégezni vele, számos kritérium alapján. Ezek a kritériumok mind rendelkeznek alapértékekkel, módosításuk az url -ben lehelyezett query string -el lehetséges. A szűrést befolyásoló értékek a következők:

Rendez: Kétféle elv alapján lehet rendezni a listát, név, és ár szerint. Ha a megadott érték nem ezek valamelyike, vagy egyáltalán nincs megadva rendezési elv, akkor rendezetlen marad a lista.

CsokkenoSorrend: Ha megfelelően megadott rendezési elvet határoztunk meg, akkor a lista ezen érték igazra állításával fordítható meg.

Keres: A lista szűkíthető egy kereső szöveggel, amennyiben meg van adva. A keresés a termék névben, és a leírásban (amennyiben van leírás) egyaránt megtörténik, mindkét találati esetre szűkül a lista. Ha nincs megadva keresőszó, úgy az összes termék listázásra kerül.

Lapszam: Ez az érték a tördeléshez használható fel. Alap értéke 1, azaz ha nincs megadva, akkor az első lapméretnyi termék kerül majd lekérdezésre. Ha ennek az értéknek és a lapméretnek a szorzata nagyobb mint ahány termék szerepel a listában, akkor üres lista kerül vissza.

Lapmeret: Ez az érték azt határozza meg, hogy mennyi termék kerüljön egyidejűleg lekérdezésre. Alap értéke 10, azaz ha nincs beállítva, úgy egyszerre maximum 10 termék kerül lekérdezésre.

/api/Termekek/TermekTorles

```
curl -X 'DELETE' \
  'szerver:1234/api/Termekek/TermekTorles/100' \
  -H 'accept: */*'
```

Egy adott termék törlése Id alapján, a hozzá tartozó képpel együtt. A jelenlegi változatban terméket törölni csak akkor lehetséges, ha nincs rá aktív megrendelés.

/api/Termekek/TermekUj Új termék felvételére használható végpont. Form adatokat vár. Egy egyszerű példa:

```
<form action="szerver:1234/api/Termekek/TermekUj
" method="post" enctype="multipart/form-data">
        <label for="Nev">Név:</label><br>
        <input type="text" id="Nev" name="Nev"><br>
        <label for="Ar">Ár:</label><br>
        <input type="text" id="Ar" name="Ar"><br>
        <label for="Leiras">Leírás:</label><br>
        <textarea id="Leiras" name="Leiras" rows="4"
         cols="50"></textarea><br>
        <label for="KategoriaId">Kategória ID:</label><br>
        <input type="text" id="KategoriaId"
        name="KategoriaId"><br>
        <input type="file" name="Kep"><br><br>
        <input type="submit" value="Submit">
    </form>
```

A kategória, valamint a kép megadása is kötelező. Kép csak jpg lehet, amit a backend tárolás előtt a file vizsgálatával határoz meg.

Forrásjegyzék

1. Téma: mariadb adatbázis: https://mariadb.com/ letöltés dátuma: 2024.04.26.
2. Téma: mariadb server: https://github.com/MariaDB/server letöltés dátuma: 2024.04.26.
3. Téma: dotnet keretrendszer: https://dotnet.microsoft.com/en-us/ letöltés dátuma: 2024.04.26.
4. Téma: react framework: https://reactjs.org/ letöltés dátuma: 2024.04.26.
5. Téma: html spec: https://html.spec.whatwg.org/ letöltés dátuma: 2024.04.26.
6. Téma: css spec: https://www.w3.org/TR/CSS2/ letöltés dátuma: 2024.04.26.
7. Téma: bootstrap css keretrendszer: https://getbootstrap.com/ letöltés dátuma: 2024.04.26.
8. Téma: entity framework: https://learn.microsoft.com/en-us/ef/core/ letöltés dátuma: 2024.04.26.