Vizsgaremek dokumentáció

Készítették:

Réti-Nagy Tamás

Handbauer Péter

Felhasználói dokumentáció

Az általunk készített szoftver egy általános raktárkészlet kezelő alkalmazás, amelyben egyszerűen megtalálhatóak a termékek raktárokon átívelően.

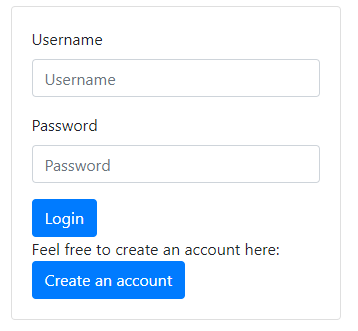
Az alkalmazásban lehetőségünk van raktárakat, polcrendszereket, polcokat, illetve termékeket kezelni, ezeket a termékeket polcokra helyezni, levenni a polcokról, illetve keresni a termékek között, megnézni a termékek elérhetőségét.

Felhasználói felület

Raktárkészlet kezelő alkalmazás

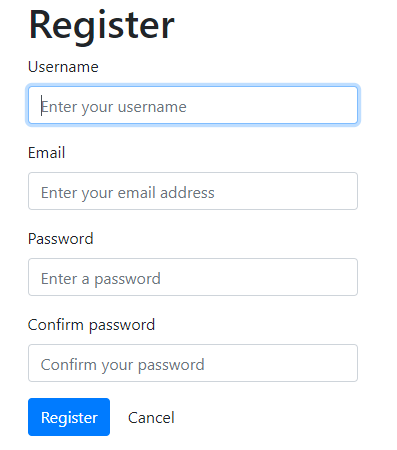
Bejelentkezés/Regisztráció/Kijelentkezés

Az alkalmazás használatához minden esetben szükség van felhasználói fiókra, illetve bejelentkezésre. Az alkalmazás megnyitásakor automatikusan a bejelentkezési felületre kerülünk.



Ezen a felületen a megfelelő bejelentkezési adatok megadása után a „Login” gombra kattintva ezek ellenőrzésre kerülnek, és ha sikeres az azonosítás, akkor visszakerülünk az alkalmazás felületére.

Ha nem rendelkezünk felhasználói fiókkal, akkor a „Create an account” gombra kattintva létrehozhatunk egyet:



A regisztrációhoz tartozó mezők követelményei:

Email:

– Szintaktikailag érvényes e-mail címet kell megadni.

Jelszó:

– Legalább 6 karakter.

– Legalább egy alfanumerikus karakter (A-Z, 0-9).

– Legalább egy nem alfanumerikus karakter/szimbólum (például , %).

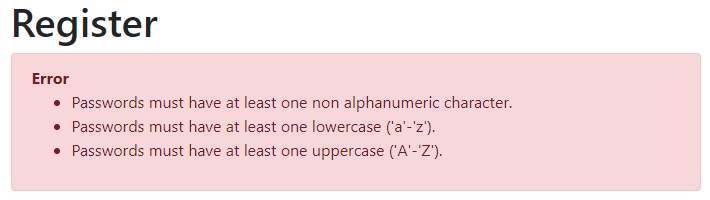
– Legalább egy kisbetűs karakter.

– Legalább egy nagybetűs karakter.

A mezők kitöltése után kattintsunk a "Register" gombra.

• Sikeres regisztráció esetén a bejelentkezési felületre kerülünk, itt már be tudunk jelentkezni a frissen készített felhasználónkkal.

• Sikertelen regisztráció esetén visszakerülünk a regisztrációs felületre, ahol megtudhatjuk a sikertelenség okát:



A Cancel gombra kattintás esetén visszatérünk az alkalmazásunkhoz.

Alkalmazott technológiák

Frontend

* Angular

Backend

A felhasználókezelést végző szerver:

* .NET 6
* Entity Framework Core: A backend alkalmazás és az adatbázis közötti kapcsolat megvalósítására.
* IdentityServer4

Az alkalmazást kiszolgáló WebAPI:

* .NET 6
* Entity Framework Core: A backend alkalmazás és az adatbázis közötti kapcsolat megvalósítására.

Adatbázis:

* MSSQL Server 2022
* SQL Server Management Studio 19 - Az adatbázisok kezelését/monitorozását segítő eszköz.

A WebAPI kód felépítése:

Controllers

A WebAPI Endpointjainak gyűjtője, ez a belépési pont az Angular alkalmazás felől

Dtos

Az endpointokon érkező és kimenő adatok adatszerkezetei.

Entities

Az adatbázis entitásainak leképezése a WebAPI oldalán.

Data

Az adatbázis felépítése, ileltve entitásai kapcsolatainak leképezése a WebAPI oldalán

Az adatbázis kezdeti adatainak feltöltését segítő DatabaseSeed megvalósítása.

– Ha nem létezik az adatbázis, akkor létrehozza azt, és migrálja is a legfrissebb verzióra.

– Feltölti az adatbázist minta adatokkal.

Repositories

Repository Design Patternt megvalósító osztályok, amelyek az adatbázisban lévő adatok eléréséhez nyújtanak segítséget.

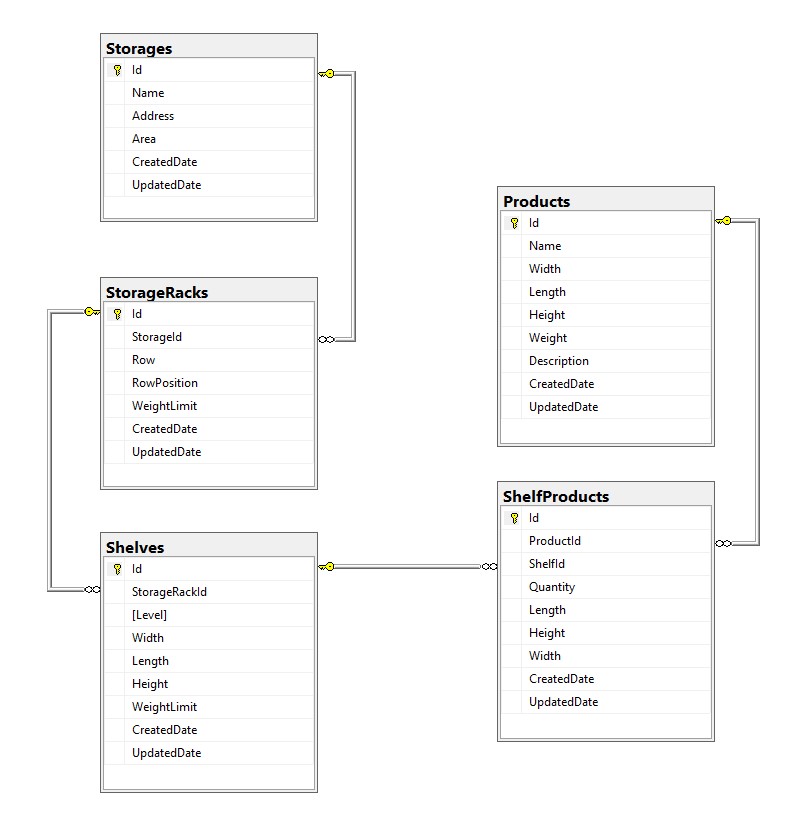
Services

A Repository-k és a Controller-ek közötti réteg, amely a modellek átalakítását, üzleti logikát tartalmaz.

Migrations

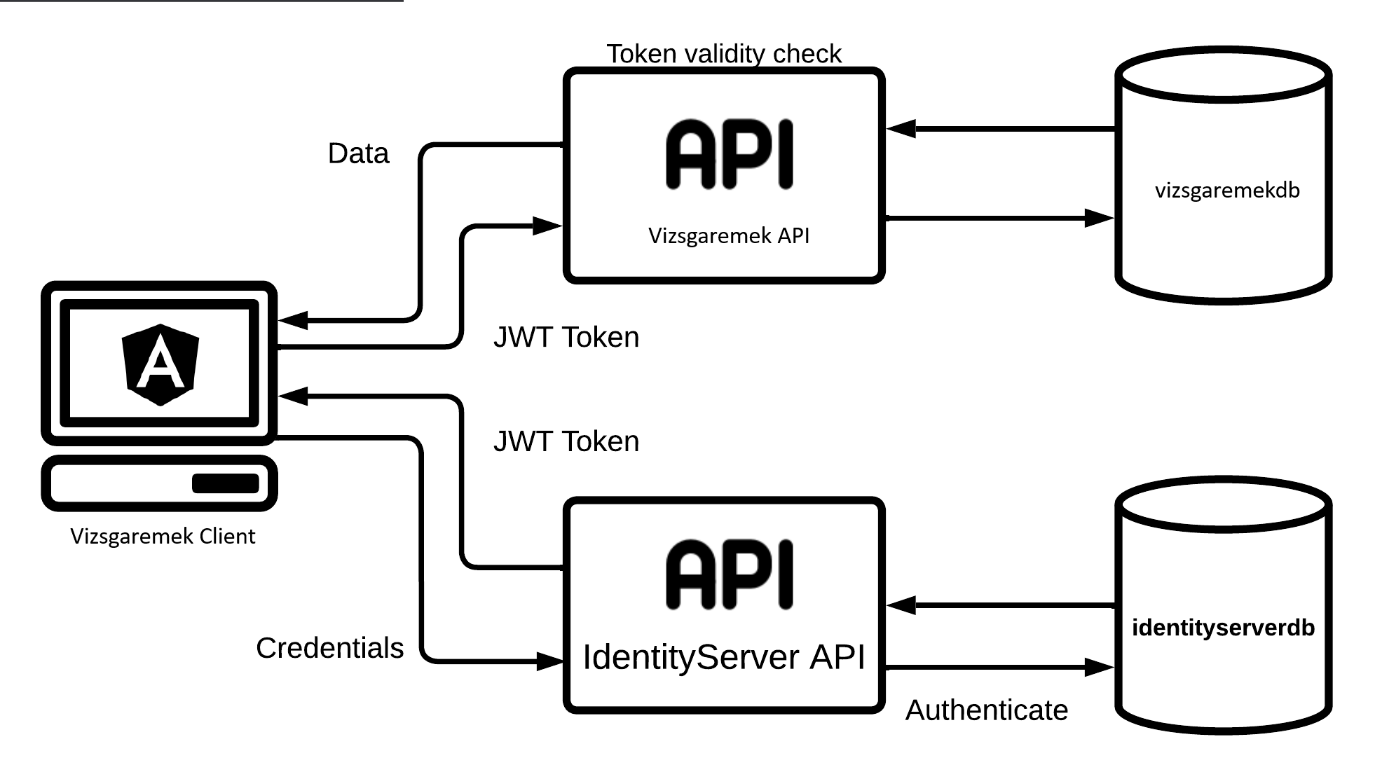
Az adatbázist felépítő migrációkat tartalmazza

Az adatbázis felépítése:



Az adatbázis modelljeinek kapcsolata.

Szoftver architectúra:



Az architectúra tervből látszik az adatok áramlása. A felhasználói felületen először azonosítani kell magunkat a bejelentkezéssel. Ez a kérés az IdentityServer felé irányul, ami ellenőrzi, hogy hitelesek-e a belépési adataink. Ha sikeres a bejelentkezés, akkor egy tokent ad vissza az Angular alkalmazásnak, és eltárolásra kerül.

Amikor az Angular alkalmazás a Vizsgaremek API felé intéz hívást, akkor a request headerjében küldi a tokent is. Az API, ha az endpoint védve van az [Authorize] attribute-tal, ellenőrzi a token hitelességét.

• Hiteles token esetén a WebAPI biztosítja az adatok elérését.

• Hibás token esetén a WebAPI 401 - Unauthorized hibakóddal válaszol.

WebAPI által biztosított végpontok:

ProductController: A termékeket kezelő végpontok

GET /api/Product – Az összes Product lekérése

POST /api/Product – Új Product hozzáadása

Minta request body:

**{**

**"id": 0,**

**"name": "string",**

**"width": 0,**

**"length": 0,**

**"height": 0,**

**"weight": 0,**

**"description": "string"**

**}**

GET /api/Product/{id} – Egy Product lekérdezése „id” alapján

PUT /api/Product/{id} – Egy Product módosítása „id” alapján. A request bodyban szükséges megadni a frissítendő adatokat:

**{**

**"id": 0,**

**"name": "string",**

**"width": 0,**

**"length": 0,**

**"height": 0,**

**"weight": 0,**

**"description": "string"**

**}**

DELETE /api/Product/{id} – Egy Product törlése „id” alapján

GET /api/Product/{id}/ProductShelves – Megadja, hogy a Product mely polcokon van jelen

GET /api/Product/Search – Megadott string alapján keres a termékek között, és visszaadja azokat, amelyek megfelelnek a keresési feltételnek.

ShelfController: A Polcokat kezelő végpontok

GET /api/Shelf – Az összes Shelf lekérdezése

POST /api/Shelf – Új Shelf hozzáadása

Minta request body:

**{**

**"id": 0,**

**"storageRackId": 0,**

**"level": 0,**

**"width": 0,**

**"length": 0,**

**"height": 0,**

**"weightLimit": 0,**

**"totalProducts": 0,**

**"shelfProducts": []**

**}**

GET /api/Shelf/{id} – Egy Shelf lekérdezése „id” alapján.

PUT /api/Shelf/{id} – Egy Shelf módosítása „id” alapján. A request bodyban szükséges megadni a frissítendő adatokat:

**{**

**"id": 0,**

**"storageRackId": 0,**

**"level": 0,**

**"width": 0,**

**"length": 0,**

**"height": 0,**

**"weightLimit": 0,**

**"totalProducts": 0,**

**"shelfProducts": []**

**}**

DELETE /api/Shelf/{id} – Egy Shelf törlése „id” alapján.

ShelfProductController:

Ezek az adatok mutatják, hogy mely polcokon mely termékek vannak.

GET /api/ShelfProduct – Az összes ShelfProduct lekérdezése

POST /api/ShelfProduct – Új ShelfProduct hozzáadása

Minta request body:

**{**

**"id": 0,**

**"name": "string",**

**"weight": 0,**

**"productId": 0,**

**"shelfId": 0,**

**"quantity": 0,**

**"totalWeight": 0,**

**"width": 0,**

**"length": 0,**

**"height": 0**

**}**

GET /api/ShelfProduct/{id} – Egy ShelfProduct lekérdezése „id” alapján.

PUT /api/ShelfProduct/{id} – Egy ShelfProduct módosítása „id” alapján. A request bodyban szükséges megadni a frissítendő adatokat:

**{**

**"id": 0,**

**"name": "string",**

**"weight": 0,**

**"productId": 0,**

**"shelfId": 0,**

**"quantity": 0,**

**"totalWeight": 0,**

**"width": 0,**

**"length": 0,**

**"height": 0**

**}**

DELETE /api/ShelfProduct/{id} – Egy ShelfProduct törlése „id” alapján.

StorageController: A raktárakat kezelő végpontok

GET /api/Storage – Az összes Storage lekérdezése

POST /api/Storage – Új Storage hozzáadása

Minta request body:

**{**

**"id": 0,**

**"name": "string",**

**"address": "string",**

**"area": 0,**

**"totalStorageRacks": 0,**

**"totalProducts": 0**

**}**

GET /api/Storage/{id} – Egy Storage lekérdezése „id” alapján.

PUT /api/Storage/{id} – Egy Storage módosítása „id” alapján. A request bodyban szükséges megadni a frissítendő adatokat:

**{**

**"id": 0,**

**"name": "string",**

**"address": "string",**

**"area": 0,**

**"totalStorageRacks": 0,**

**"totalProducts": 0**

**}**

DELETE /api/Storage/{id} – Egy Storage törlése „id” alapján.

StorageRackController: A polcszerelvényeket kezelő végpontok

GET /api/StorageRack – Az összes StorageRack lekérdezése

POST /api/StorageRack – Új StorageRack hozzáadása

Minta request body:

**{**

**"id": 0,**

**"storageId": 0,**

**"row": 0,**

**"rowPosition": 0,**

**"weightLimit": 0,**

**"shelves": 0**

**}**

GET /api/StorageRack/{id} – Egy StorageRack lekérdezése „id” alapján.

PUT /api/StorageRack/{id} – Egy StorageRack módosítása „id” alapján. A request bodyban szükséges megadni a frissítendő adatokat:

**{**

**"id": 0,**

**"storageId": 0,**

**"row": 0,**

**"rowPosition": 0,**

**"weightLimit": 0,**

**"shelves": 0**

**}**

DELETE /api/StorageRack/{id} – Egy StorageRack törlése „id” alapján.

GET /api/StorageRack/{id}/Shelves – Egy StorageRack Shelf-jeinek lekérése.

IdentityServer

A szoftver felhasználókért felelős részét az IdentityServer4 könyvtárral valósítottam meg. Találtam hozzá egy érthető, használható útmutatót, ez alapján, építettem meg a saját IdentityServert. Az IdentityServer egy alap verziója elérhető githubon. Ezt a verziót használtam kiindulópontnak. Szerencsére a license-e ezt lehetővé tette. Pár apró módosítást kellett végezni rajta, például a regisztráció lehetősétét biztosítani a bejelentkezési oldalról, illetve hogy a bejelentkezés ellenőrzéséhez ne teszt adatokat használjon, hanem az adatbázisban lévő adatokat.

Használati útmutató: <https://code-maze.com/identityserver-4-series/>

Quickstart verzió: <https://github.com/IdentityServer/IdentityServer4.Quickstart.UI>

Quickstart license:

<https://github.com/IdentityServer/IdentityServer4.Quickstart.UI/blob/main/LICENSE>

A program indítása

A program indításához szükséges összetevők:

* Visual Studio Code
* Node.js
* Angular CLI: Node.js telepítése után terminálban: "npm install -g @angular/cli
* MSSQL Server 2022
* SQL Server Management Studio 19

Ha a backendet is fordítókörnyezetből szeretnénk indítani:

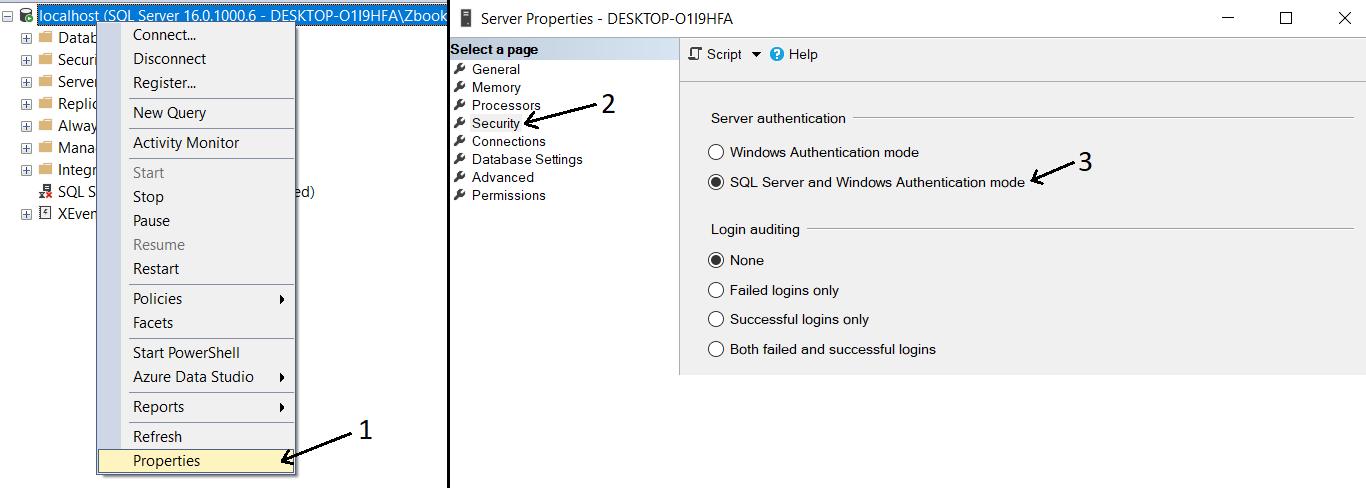
* Visual Studio 2022

Mivel .NET 6 keretrendszert használtam a szoftverhez, ezért a szoftver az alábbi operációs rendszer verziókkal kompatibilis:

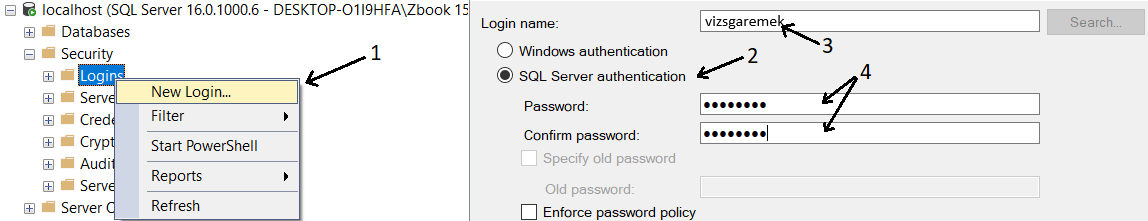
<https://github.com/dotnet/core/blob/main/release-notes/6.0/supported-os.md>

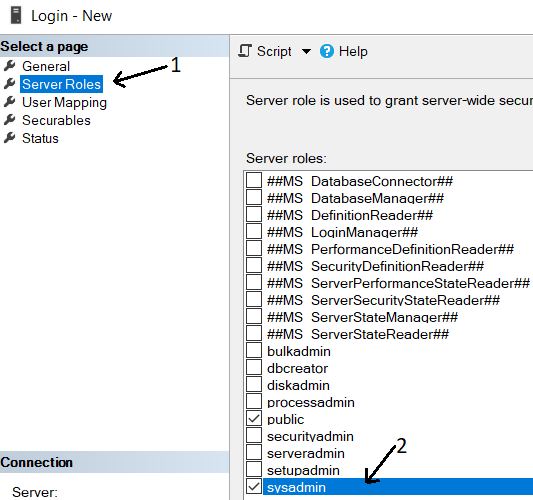
Backend indítása az alapoktól

* MSSQL Server telepítése
* SQL Server Management Studio telepítése
* Opcionális: Visual Studio telepítése
* Ha az MSSQL Server csak Windows authentication-nel lett telepítve, akkor át kell állítani "SQL Server and Windows authentication mode"-ra.



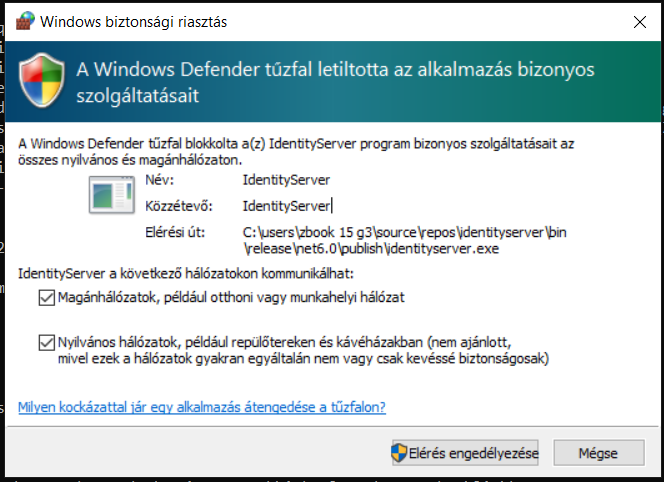
* MSSQL Server login létrehozása. A tesztelésre használt felhasználó "server\_role"-jai a következők voltak: public, sysadmin





Publisholt, egyszerűen indítható verzió indítása:

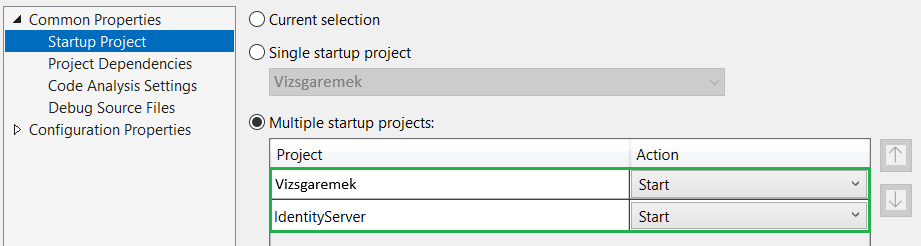
* Backend/Vizsgaremek/bin/Release/net6.0/publish/appsettings.json:
  + ConnectionString átírása az aktuális MSSQL server eléréséhez:
  + "Server=xxxxx;Initial Catalog=brickitdb;User Id=yyyyy;Password=zzzzz;TrustServerCertificate=True;"
  + Helyettesítsük be a megfelelő paramétereket:
    - ∗ xxxxx - MSSQL server elérése (pl. localhost)
    - ∗ yyyyy - MSSQL server login felhasználónév (pl. sa)
    - ∗ zzzzz - MSSQL server login jelszó
* Backend/IdentityServer/bin/Release/net6.0/publish/appsettings.json:
  + ConnectionString átírása az aktuális MSSQL server eléréséhez:
  + "Server=xxxxx;Initial Catalog=brickitdb;User Id=yyyyy;Password=zzzzz;TrustServerCertificate=True;"
  + Helyettesítsük be a megfelelő paramétereket:
    - ∗ xxxxx - MSSQL server elérése (pl. localhost)
    - ∗ yyyyy - MSSQL server login felhasználónév (pl. sa)
    - ∗ zzzzz - MSSQL server login jelszó
* IdentityServer indítása: Backend/IdentityServer/bin/Release/net6.0/publish/IdentityServer.exe futtatása
  + Első indításkor a Windows Defender engedélyt kér, hogy engedje az alkalmazás kommunikációját a hálózatokon.



* + Indításkor létrehozza a ConnectionString-ben megadott adatbázist, ha az még nem lézetik
  + Adatbázis ellenőrzése/létrehozása után adatokkal tölti fel az adatbázist, létrehozva így egy tesztelésre használható admin felhasználót. (admin/Pa55w0rd!)
* Vizsgaremek API indítása: Backend/BrickIT/bin/Release/net6.0/publish/BrickIT.exe futtatása
  + Első indításkor a Windows Defender engedélyt kér, hogy engedje az alkalmazás kommunikációját a hálózatokon.
  + Indításkor létrehozza a ConnectionString-ben megadott adatbázist, ha az még nem lézetik.
  + Adatbázis ellenőrzése/létrehozása után adatokkal tölti fel az adatbázist, hozzáadva raktárakat, polcszerkezeteket, polcokat, termékeket.

Fejlesztőkörnyezetből indítás

* Nyissuk meg a Vizsgaremek.sln fájlt Visual Studio-val.
* A Solution Explorerben a solution-re jobb kattitás -> Properties -> A "Startup project" menüpontban állítsuk mindkét projektet "Start"-ra.



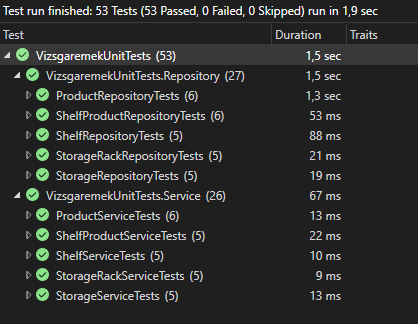
* Vizsgaremek projekten belüli appsettings.json fájlban lévő ConnectionString beállítása.
* IdentityServer projekten belüli appsettings.json fájlban lévő ConnectionString beállítása.
* Kattintsunk a Visual Studio "Start" gombjára.

Frontend indítása az alapoktól

* Visual Studio Code telepítése.
* Node.js telepítése.
* Angular CLI telepítése.
* Frontend mappa megnyitása Visual Studio Code-ban.
* Visual Studio Code terminálban: "npm install" parancs futtatása.
* Visual Studio Code terminálban: "ng serve" parancs futtatása.
* A Frontend webes alkalmazás elérhető a http://localhost:4200 címen.

Tesztelés

A Vizsgaremek solution-ben található egy UnitTest projekt VizsgaremekUnitTests néven. Ebben a projektben található 53 Unit test, amely átfogóan teszteli a Repository és a Service rétegeket.



Ezen felül sor került manuális tesztelésre is:

* Az első lépésben az alkalmazás felhasználói felületét teszteltük. Ellenőriztük a gombok és menük működését, és biztosítottuk, hogy az összes funkció és opció elérhető legyen. Különböző felhasználói interakciókat szimuláltunk, és ellenőriztük, hogy az alkalmazás megfelelően reagál-e.
* A következő lépésben a háttérlogikát és az adatkezelést teszteltük. Ellenőriztük az adatbázisba történő adatfelvitelt és az adatok lekérdezését. Különböző műveletek végrehajtása után ellenőriztük, hogy a szoftver helyesen kezeli-e az adatokat. Teszteltük az adatfeldolgozási folyamatot, és ellenőriztük, hogy a rendszer a várt eredményeket adja-e vissza.
* A harmadik lépésben a különböző komponensek közötti kommunikációt teszteltük. Ellenőriztük, hogy az adatok helyesen átadódnak-e az egyes komponensek között, és hogy a kommunikáció hibamentesen működik-e.