**SZÉCHENYI ISTVÁN Egyetem  
INFORMATIKAI KAR**

**SZAKDOLGOZAT**

**Pintér Gábor U7MU92**

mérnökinformatikus BSc szak

2023

1. SZÉCHENYI ISTVÁN EGYETEM
2. INFORMATIKAI KAR
3. Gépészmérnöki, Informatikai és Villamosmérnöki Kar

**SZAKDOLGOZAT**

AUTOKER

Pintér Gábor

mérnökinformatikus BSc szak

**Győr, 2023**

# Feladatkiírás

# Értékelő lap

1. **Konzultációs lap szakdolgozathoz és diplomamunkához**

# Nyilatkozat

# Összefoglaló

A szakdolgozat témájaként egy autókereskedéses web-alkalmazásnak a megvalósítását választottam. mind a backend-et és frontend-et leprogramozva. Az kódsorok elsődleges célja az volt, hogy az alkalmazást éles környezetben is megfelelően lehessen használni és fellelhetőek legyenek benne azok a funkciók, illetve lehetőségek, amik egy ilyen alkalmazástól elvárhatóak.

A belső működés a Java programozási nyelv és a Spring, Spring Boot keretrendszerek segítségével valósulhatott meg, míg az alkalmazást használó személy által látott tartalmak elkészítésében a HTML, CSS programozási nyelvek és Thymeleaf keretrendszer voltak a segítségemre. A megfelelő http kérés elküldése után egy szintén http válasszal tér vissza az alkalmazásunk az annotációkban definiált módon. Erre külön osztályok és metódusok lettek létrehozva annak érdekében, hogy a különböző szinteket szét tudjuk szedni. Ezzel azt segítjük elő, hogy a program átláthatóbb lesz és amennyiben további változtatásra van szükség azt a dinamikusságnak köszönhetően egyszerűbben tudjuk megvalósítani.

Annak érdekében, hogy a hozzáadott járműveket és elvégzett módosításokat ne csak ideiglenesen tudjuk tárolni, MySQL adatbázissal került összekapcsolásra a Spring Boot alkalmazásunk és így minden felvett értéket biztonságosan és perzisztensen mentünk el. Ezt megelőzően H2-es adatbázissal működött a program, ami csak a futási időben őrizte meg a bevitt adatokat, de itt a cél csupán a megfelelő és logikus működés kialakítása volt.

Az elkészült alkalmazásból a Maven Build parancs segítségével egy futtatható .jar kiterjesztésű állomány is létrehozásra került, aminek köszönhetően a program könnyedén felhasználhatóvá válik más platformokon is, ugyanis ez biztosítja a hordozhatóságot. Erre a .jar típusú fájlra a Docker és Jenkins alkalmazásokban is hivatkozunk. Az építés során pedig a tesztesetek megfelelő lefutása garantálja azt, hogy minden rendben van.

A Docker segítségével érjük el azt, hogy a meglévő projektünket ne csak a fejlesztői környezeten belül lehessen futtatni, hanem például egy Docker Desktop alkalmazással vagy parancssorból is. Ez még dinamikusabbá teszi az alkalmazást és egy docker vagy docker-compose fájlnak köszönhetően könnyedén össze tudjuk fogni a különböző részeket, mint például a backend, frontend és adatbázis egységeket.

A folyamatos fejlesztés és tesztelés elősegítése érdekében a Jenkins is helyett kapott a felhasznált alkalmazások listájában. Itt a csővezeték létrehozása után egy külön erre szolgáló fájlban definiáltuk a lépéseket, amelyeket a Jenkisben végre akarunk hajtani. Ez a fájl a Github-on található meg és a megfelelő URL cím segítségével keresi meg a Jenkins. Amennyiben minden sikeres volt, akkor ezt zöld üzenetek jelzik és a DockerHub-ra is feltöltjük az aktuális Docker képünket.

# Summary

As the topic of my thesis, I chose to implement a car dealership web application, programming both the backend and frontend. The primary goal of the code was to ensure that the application could be used in a live environment and that it contained the functionalities and features expected from such an application. The internal functioning was achieved using the Java programming language and Spring and Spring Boot frameworks, while HTML, CSS programming languages, and Thymeleaf framework were used to create the content visible to the user of the application. After sending the appropriate HTTP request, the application returns with an HTTP response defined in the annotations. To achieve this, separate classes and methods were created to break down the different levels, making the program more transparent and allowing for easier implementation of any necessary changes due to its dynamic nature.

To store the added vehicles and performed modifications permanently, our Spring Boot application was connected to a MySQL database, safely and persistently saving every recorded value. Previously, the program used an H2 database that only preserved entered data at runtime, but here the goal was to create the appropriate and logical functioning. With the Maven Build command, a runnable .jar file was also created from the completed application, which allows for easy use on other platforms due to its portability. We also reference this .jar file in Docker and Jenkins applications. During the building process, the appropriate running of the test cases ensures that everything is in order.

With Docker, we can run our existing project not only within the development environment but also, for example, with a Docker Desktop application or from the command line. This makes the application even more dynamic, and with a docker or docker-compose file, we can easily combine different parts, such as backend, frontend, and database units.

To promote continuous development and testing, Jenkins is also included in the list of used applications. After creating the pipeline, we defined the steps we want to execute in a separate file, which the Jenkins finds using the appropriate URL address on Github. If everything was successful, this is indicated by green messages, and we also upload our current Docker image to DockerHub.

# Tartalomjegyzék

Bevezetés 5

1. A szakirodalom feldolgozása 4

1.1 Alcím, második címszinten 4

1.1.1 Fejezetrész címe harmadik címszinten 5

1.1.2 Következő fejezetrész címe harmadik címszinten 5

1.1.3 Újabb fejezetrész címe harmadik címszinten 5

1.2 Következő alfejezet, második címszinten 6

1.2.1 Fejezetrész címe harmadik címszinten 6

1.2.2 Újabb fejezetrész címe harmadik címszinten 7

2. Gyakorlati rész 7

2.1 A tématerület ismertetése 7

2.1.1 Fejezetrész címe harmadik címszinten 7

2.1.2 Következő fejezetrész címe harmadik címszinten 8

2.2 Elemzések 8

2.2.1 Az elemzések első alfejezete – például: Felmérések 8

2.2.2 Az elemzések második alfejezete, például: Elemzési módszerek 9

2.2.3 Következő alcím − például: Megállapítások 10

2.2.4 Következő alcím − például: Az elemzések értékelése 10

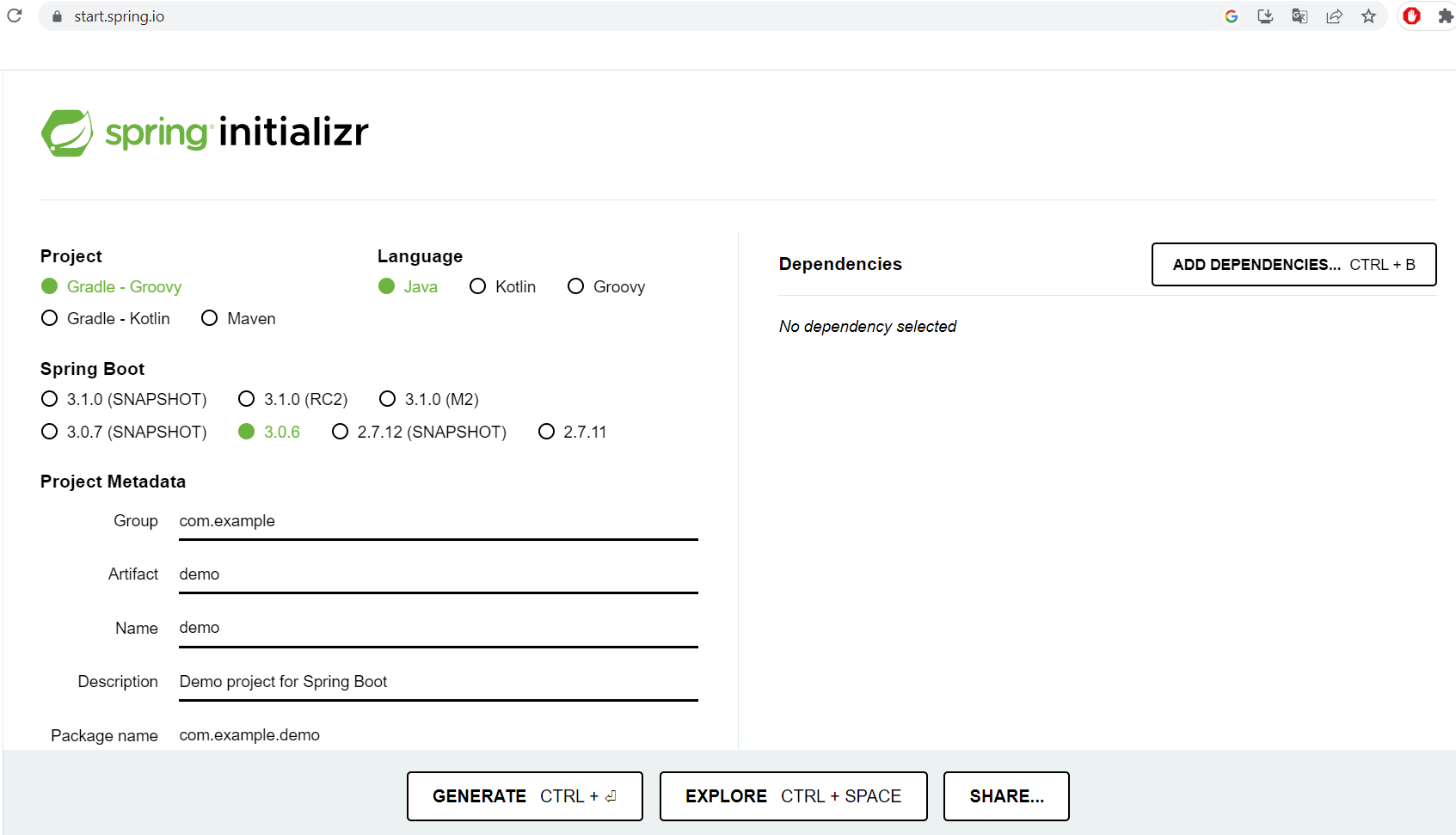
2.3 Következtetések, javaslatok 11

Összegzés 11

Irodalomjegyzék 11

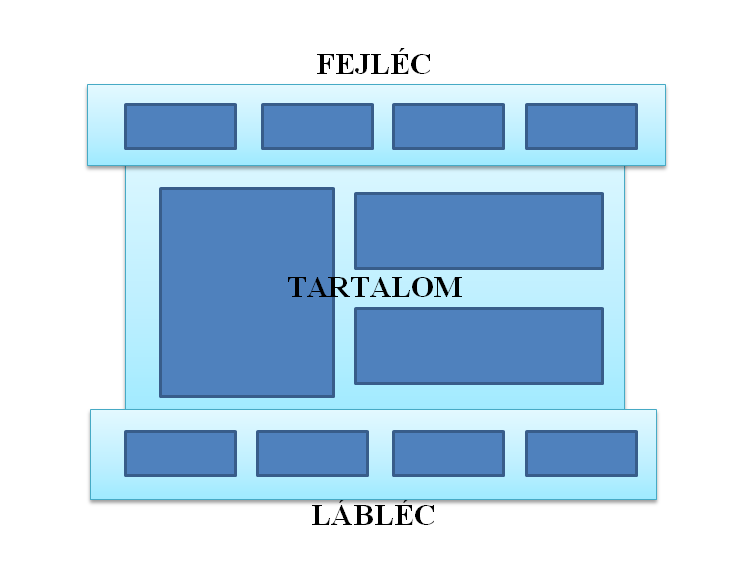
Mellékletek 14

# Bevezető

Az alkalmazás Spring Boot keretrendszerrel megvalósított, autókereskedésnek fejlesztett webalkalmazás, amely a Spring Initializr oldal segítségével lett előkészítve Maven projektként, Java nyelven. Az oldal számos lehetőséget biztosít, amely meg tudja könnyíteni a fejlesztő dolgát, hiszen az alap adatok, pl. osztály és csomag nevek megadása mellett lehetőség van a verzió és magának a nyelvnek a definiálására is. Ezeken felül úgynevezett függőségeket is hozzá tudunk adni a projektünkhöz. Ezek elengedhetetlenek ahhoz, hogy a program a későbbiekben megfelelően működjön, ugyanis egyes annotációkat, osztályokat vagy metódusokat nem ismeri fel a fordító program a hozzájuk tartozó .jar kiterjesztésű file-ok hiányában. Én személy szerint az alapértelmezett alkalmazás legenerálását is .jar kiterjesztéssel szoktam kezdeményezni, de arra is van lehetőségünk, hogy helyette .war fájlt kérjünk.

**A program célja:**

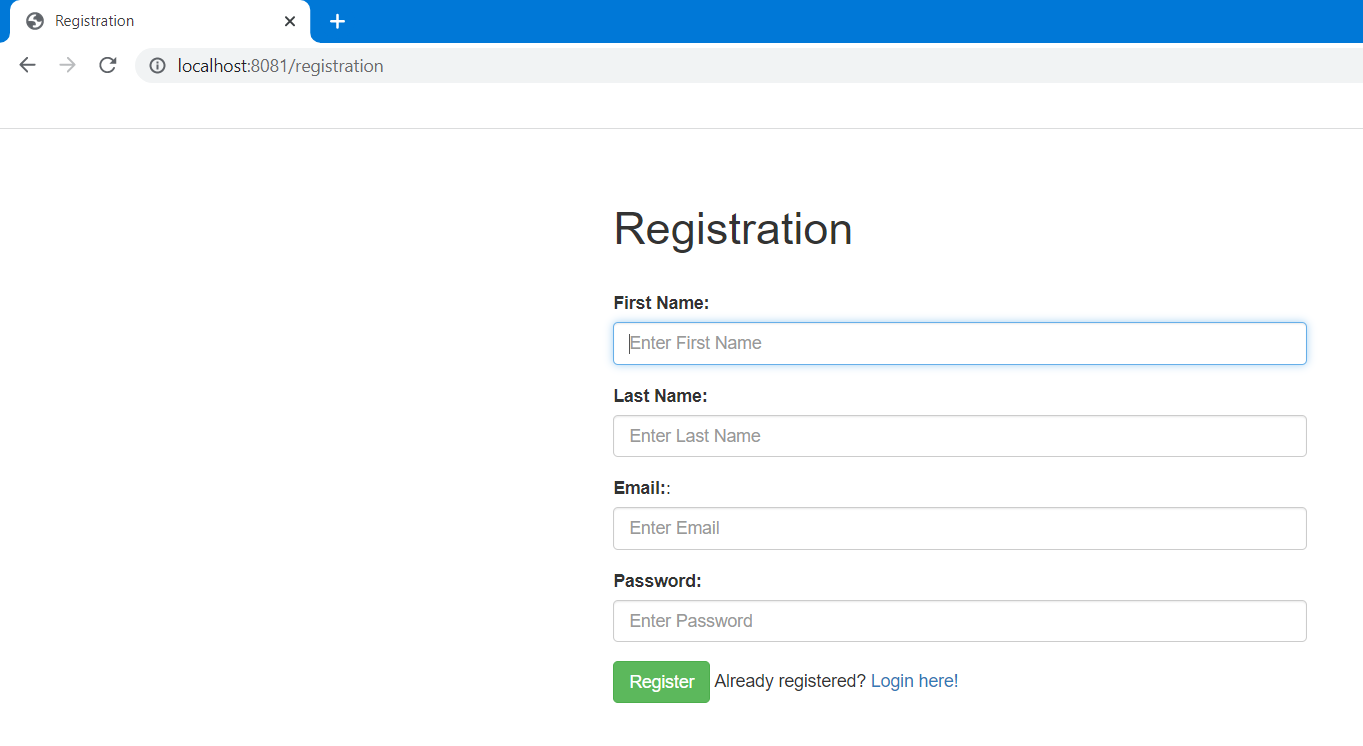
Egy autókereskedés részére fejlesztett webalkalmazást valósítanak meg a kódsorok, ami a megfelelő felhasználónévvel és jelszóval történő autentikációt igényel. A biztonság érdekében a felhasználó nem tudja elérni az autókereskedéshez tartozó oldalakat a bejelentkezés nélkül. Amennyiben rendelkezik már profillal, akkor az ehhez tartozó email címet és jelszót kell beírnia a belépő oldalon megjelenő szövegdobozokba. Ha megfelelően gépelte be az adatokat, illetve az adatbázisban ténylegesen megtalálhatóak ezek az értékek, akkor tovább tud haladni az autókereskedés nyitó oldalára. Hogyha nem rendelkezik még profillal, akkor be kell regisztrálnia egyet az adatbázisba. Miután ez sikeresen megtörtént, elénk tárul a weboldal nyitó felülete, amely tartalmaz néhány képet. Ezekre kattintva különböző funkciókat vehet igénybe a felhasználó a megfelelő jogosultság birtokában, mint pl. az autó hozzáadása, garázsok megtekintése, illetve egy adott autó keresése. A garázsok megtekintése során lehetőségünk van egy adott autót leinformálni, megvásárolni, szerkeszteni vagy törölni és ezek az aktuális bejelentkezett felhasználó jogosultságától függenek. Ezek mellett a további információszerzés érdekében van egy kapcsolat opció is a fejlécben, amire rákattintva emailt lehet küldeni az alkalmazást üzemeltető adminnak. A telephelyek és a központ könnyebb megtalálhatósága érdekében egy térkép opció is feltűntetésre került, amelyre kattintva egy Google Maps jellegű térkép tárul elénk a címekkel és a nyitvatartási idővel. Végül van egy kijelentkezési funkció is, amelyre kattintva újra a bejelentkezési felületen találhatjuk magunkat.

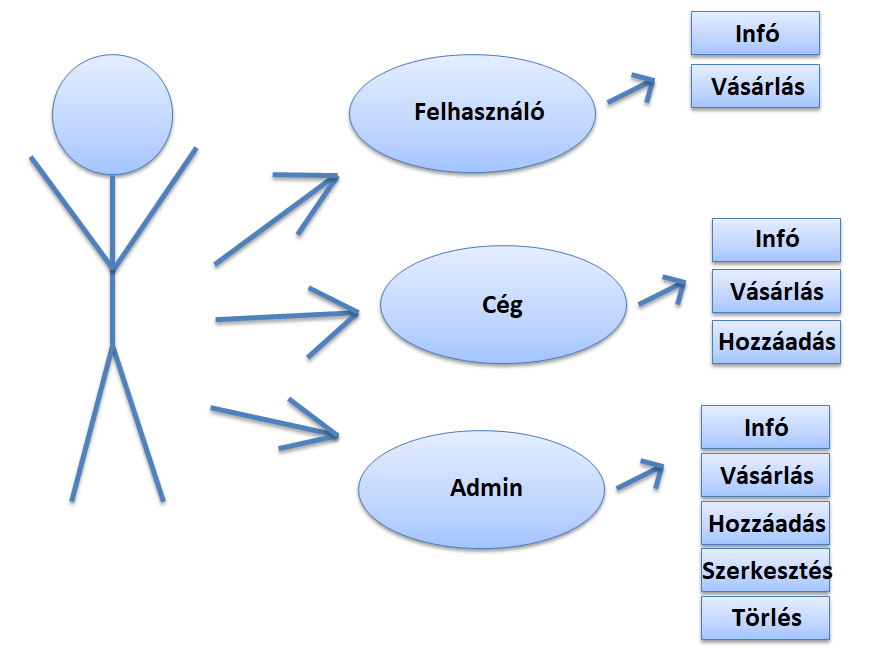
**Tervezés:**

A tervezés során figyelembe vettem minden olyan tényezőt, amely egy autókereskedésnél szerepet játszhat és ezeknek megfelelően kezdtem el leprogramozni az alkalmazásomat. Az alap elképzelés az volt, hogy legyen egy fejléc ahol az alkalmazás által biztosított lehetőségek vannak, legyen egy lábléc ahol szintén hasznos opciókat tudunk felvenni és középen helyezkedjen el a tartalom rész. Ihletként már meglévő autókereskedések weboldalait néztem meg és ezek voltak segítségemre a fejlesztés megkezdésében.

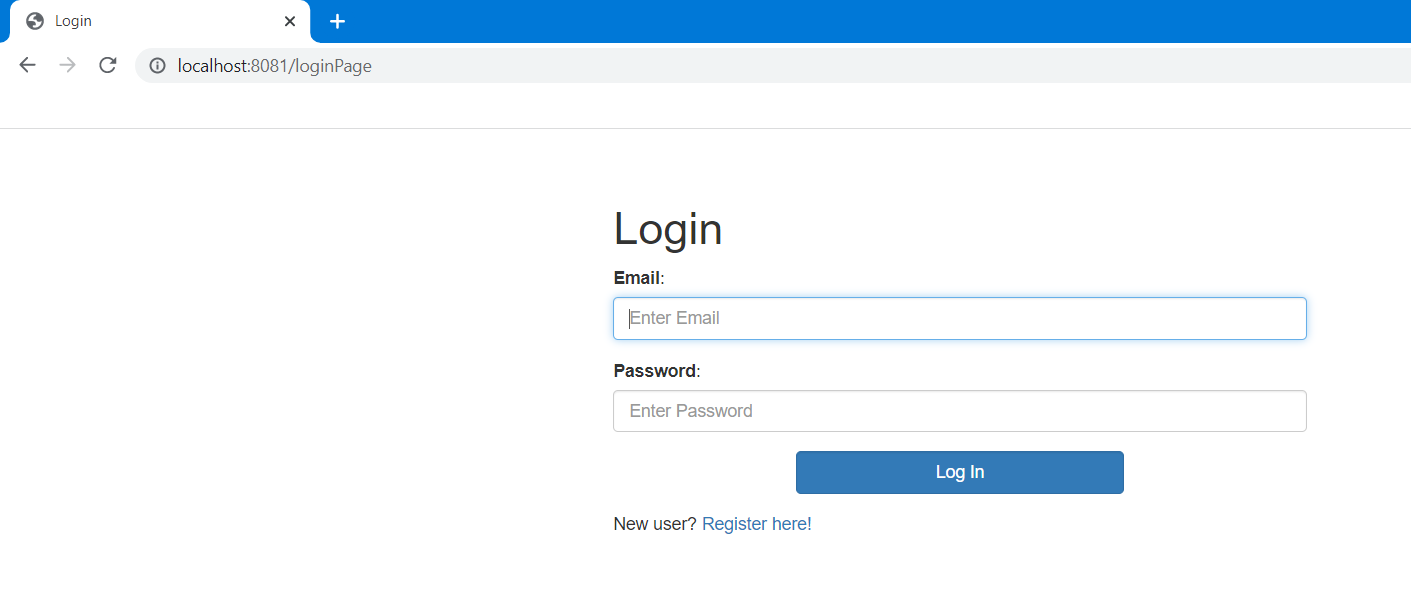
**Az alkalmazás működése**

**Regisztráció:**

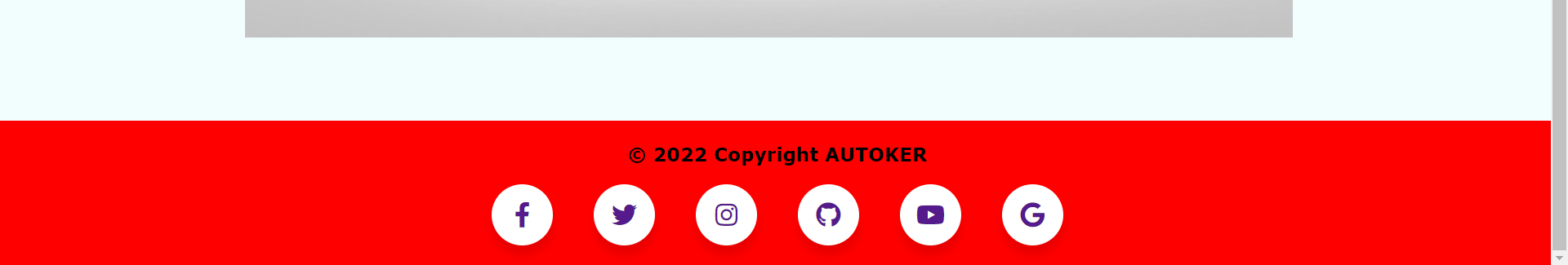
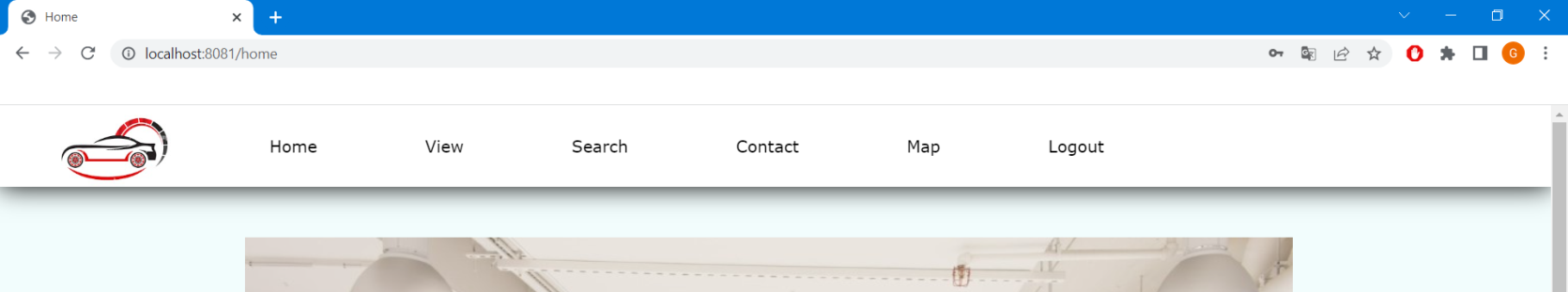
 Alapértelmezett esetben feltételezzük, hogy aki használni szeretné az alkalmazásunkat még nem rendelkezik felhasználói profillal és ilyenkor kell a regisztrációs oldalra menni. Itt az illetőnek egy vezeték és keresztnevet, egy emailt és egy jelszót kell megadnia, majd a regisztráció gombra kattintania. Ki van kötve, hogy a vezeték és kereszt névnek betűket kell tartalmaznia és minimum két darabot a sikeres továbbhaladás érdekében. Email cím megadása során egy minta van segítségünkre és olyan értéket kell beírnunk, ami tartalmaz egy @ színbólumot és egy pont karaktert. Emellett a domain nem tartalmazhat számot és végig kis betűket tartalmazhat csak az email. Jelszó megadása során csak annyi kikötés van, hogy legalább lét karakterből kell állnia. Ha minden sikeres volt, akkor az adatbázisba elmentődnek az értékeink és egy zöld üzenetet kapunk vissza. Ha az általunk megadott email az adatbázisban már fellelhető, akkor ezt egy piros hibaüzenet jelzi és más email címmel kell regisztrálnunk. Az oldal alján a Login here! szövegre kattintva át tudunk lépni a bejelentkező felületre. Regisztráció során három szintet különböztetünk meg, ami a felhasználó, a cég és az admin és ettől fog függeni a későbbiekben az, hogy milyen jogosultságokkal rendelkezünk az alkalmazás használata során.

****

**Bejelentkezés:**

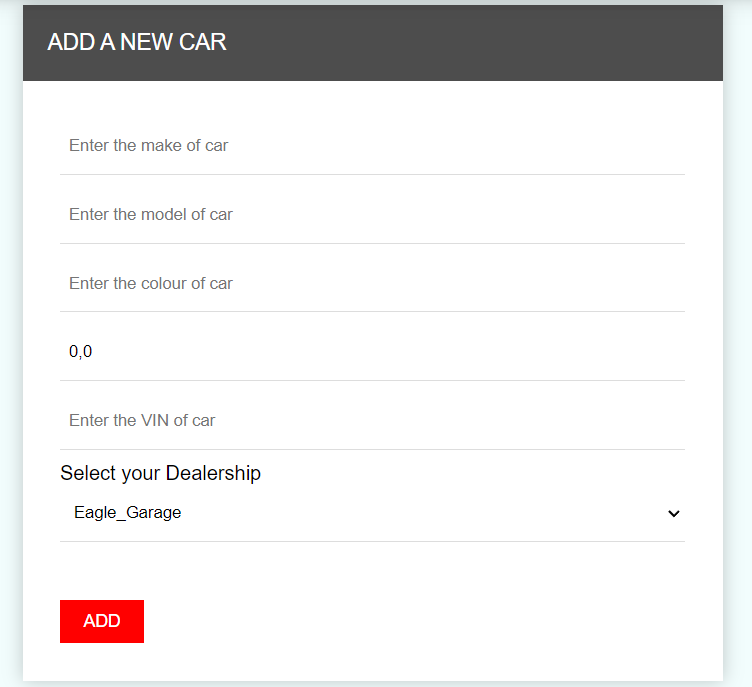
A bejelentkezés megfelelő működéséért a Spring Security felelős, ami egy hatékony és nagymértékben testre szabható hitelesítési és hozzáférés felügyeleti keretrendszer. Ahhoz, hogy elérhessük bármelyik oldalt a webalkalmazáson belül, a bejelentkezési fázison kell sikeresen túljutnunk és ezt a megfelelő jelszóval és felhasználó névvel tudjuk elérni. Amennyiben valamelyik megadott paraméter nem volt helyes, akkor nem tudunk tovább lépni és újra kell próbálkoznunk, ezúttal lehetőleg helyes adatok megadásával. Hibás érték megadásánál egy piros üzenet jelzi nekünk, hogy valamit elgépeltünk vagy még nem is létezik olyan felhasználó, ami rendelkezik a megadott paraméterekkel. Kijelentkezés után újra ezen az oldalon találjuk magunkat és egy kék üzenet jelzi, hogy kijelentkeztünk az alkalmazásból. A bejelentkezési felület legalján található egy Register here! szöveg, amire rákattintva át tudunk lépni a regisztrációs felületre. Ha a bejelentkezés sikeres volt, akkor a nyitó oldalon találhatjuk magunkat, ahol három kép tárul elénk. A képekre kattintva egy új autót tudunk hozzáadni valamelyik garázsunkhoz vagy meg tudjuk tekinteni a garázsainkban álló autókat vagy egy konkrét autóra tudunk rákeresni keresési feltételek alapján. Ezek a funkciók a képeken feltűntetésre kerültek és ezeknek megfelelően viselkedik a program. Az egyszerűség érdekében a program rendelkezik egy fejléccel is ahol ezek a lehetőségek szintén elérhetőek.

**Főoldal:**

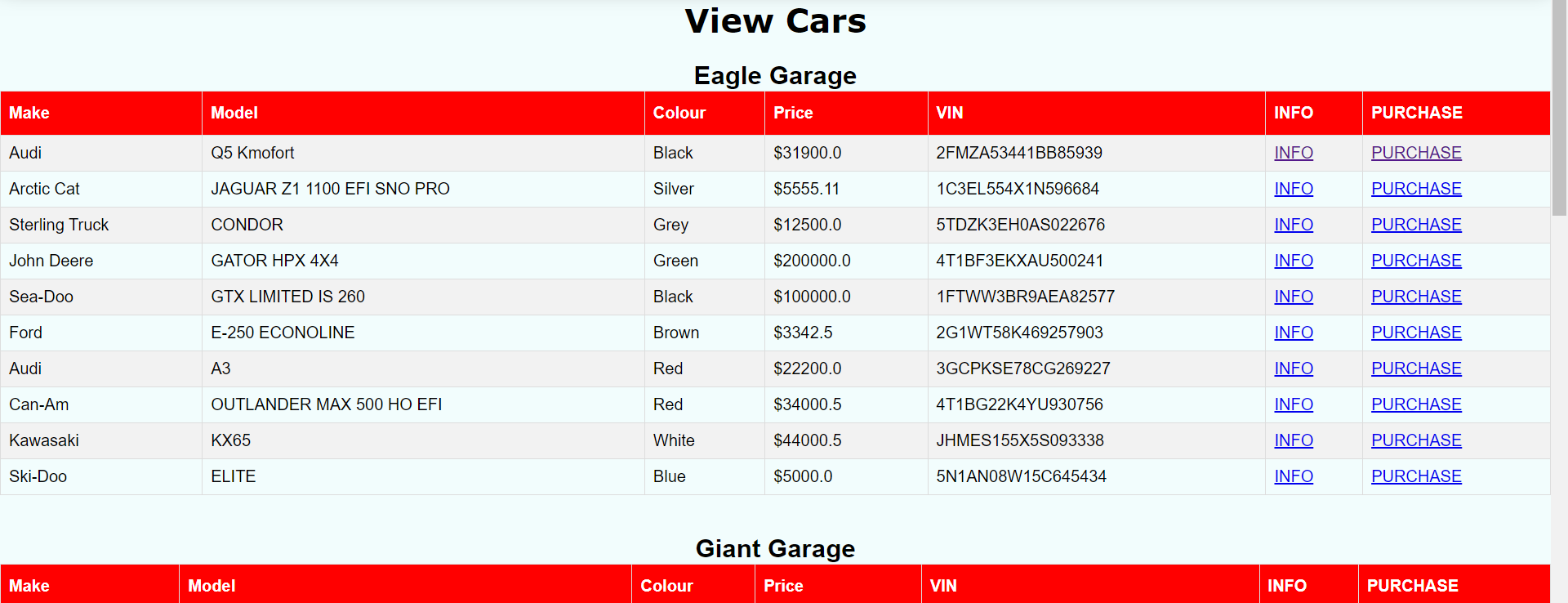
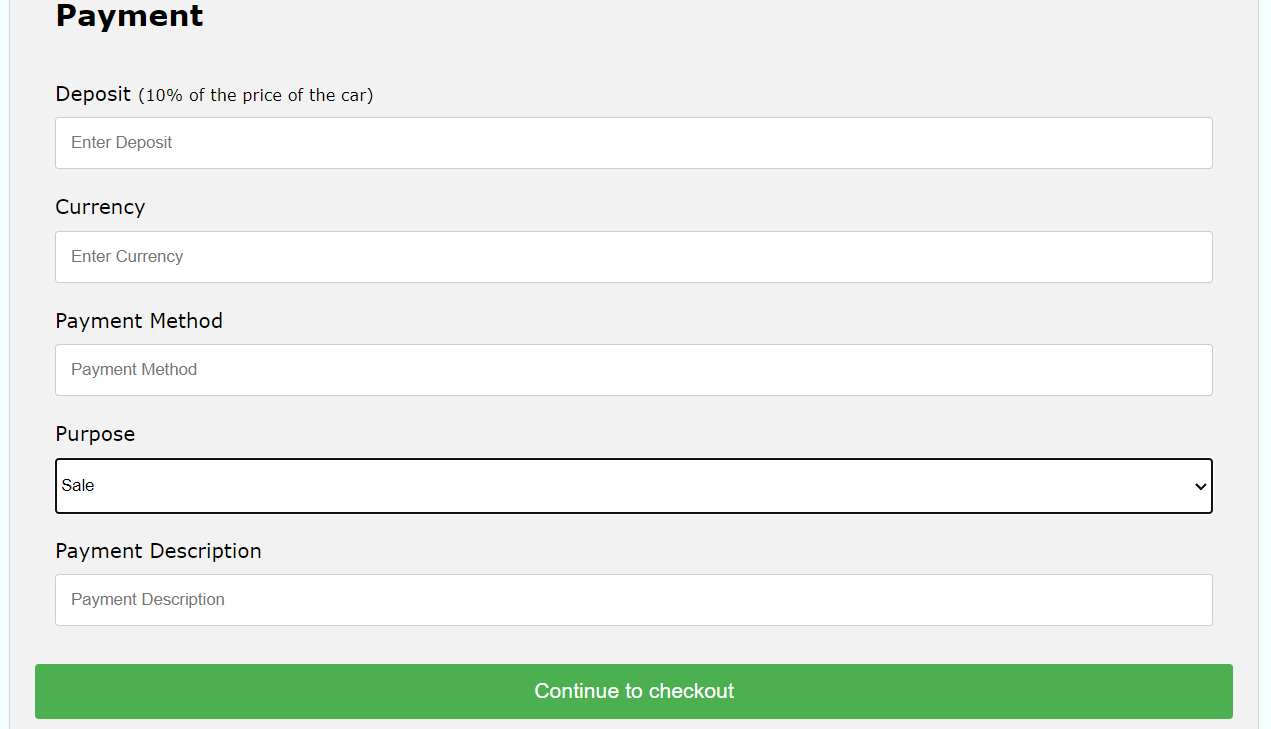
****Sikeres bejelentkezést követően tudunk ide eljutni. Az oldal rendelkezik egy fejléccel és egy lábléccel a könnyen kezelhetőség érdekében, illetve középen helyezkednek el a képek. A fejlécben a hozzáadás, a megtekintés, a keresés, a kapcsolat, a térkép és a kijelentkezés lehetőségek találhatóak meg. A főoldalon lévő képekre kattintva is át tudunk navigálni a hozzáadás, a megtekintés vagy a keresés lapokra. Fontos kiemelni, hogy autót hozzáadni a garázsainkhoz csak cég és admin szintű jogosultságokkal tudunk. A további egyértelműség érdekében a képeken fel van tűntetve az is, hogy a rákattintást követően milyen oldalra fog minket irányítani. A láblécben pedig a szerzői jog feltűntetése mellett a leggyakrabban használt közösségi médiás ikonok és kapcsolattartásra szolgáló alkalmazások is el lettek helyezve. Ezek név szerint a Facebook, a Twitter, az Instagram, a Github, a YouTube és a Gmail.

**Új autó hozzáadása:**

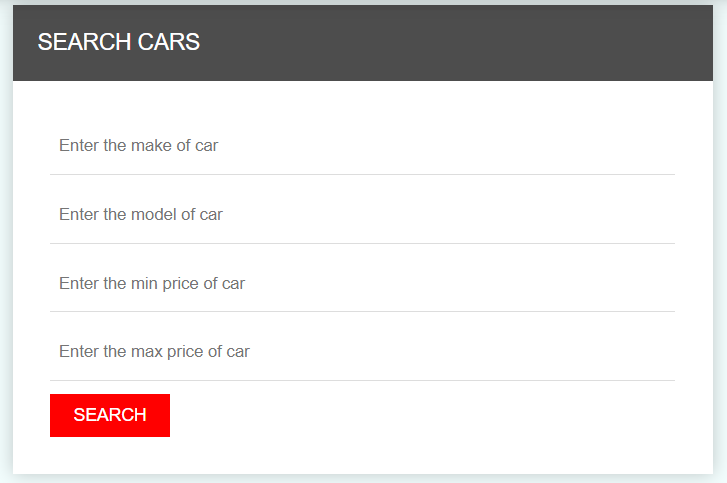
Ezt a lehetőséget kiválasztva tudjuk valamelyik garázsunkat bővíteni egy általunk megadott autóval. Ilyenkor szükséges, hogy megadjuk azt, hogy a hozzá adni kívánt autót ki gyártotta, mi a típusa, milyen színű és mi az alvázszáma. Az alvázszámnak 17 karakterből kell állnia, ahogy az elő van írva életszerű körülmények között is. Ezeken felül értelem szerűen meg kell adni az autónak az árát is, hiszen egy autókereskedésnek az oldaláról beszélünk, ahol lehetőség van az elérhető kínálatból vásárolni is. Az autó ára dollárban értendő. Az ár megadása során az a kitétel, hogy az értéknek 100 és 999999 között kell lennie. Majd amikor az említett tulajdonságokat megadtuk, akkor azt kell kiválasztanunk, hogy melyik az a kereskedés, amelynek a kínálatát bővíteni szeretnénk az aktuális autóval. Jelenleg három közül tudunk választani, amelyek a Sas, Óriás és Szuper Garázs. Végül a hozzáadás gombra kattintva az autó fellelhető lesz az adott kereskedésben és a vásárlók számára is elérhetővé válik. A hozzáadás funkció csak cég és admin szintű jogosultsággal elérhető.



**Megtekintés:**

Ezt az opciót kiválasztva elénk tárul mind a három garázs kínálata a már jelen lévő és az esetleg újonnan hozzáadott autókkal együtt. Az átláthatóság érdekében az adatok táblázatban kerültek megjelenítésre, amelyben egymás alatt listázódnak ki a különböző tulajdonságokkal eltárolt kocsik. Az első oszlopban a gyártó található. Ezt követi a típus, aztán a szín, majd az ár és alvázszám. Egy átlag felhasználó a kínálat megtekintése során ezeket látja, illetve még lehetősége van a számára szimpatikus autó megvásárlására és az Info opcióra kattintva még több tájékoztatást kaphatunk az adott járműről. Az Info lehetőséget kiválasztva egymás alatt megjelennek az autó adatai beleértve az árat és az ebből kiszámolt adókat is. Ezek mellett az is láthatóvá válik, hogy melyik garázsban lelhető fel a gépjármű és egy kép is megjelenik a bal oldalon. A megvásárlás opcióra kattintva egy új oldalra kerülhetünk, ahol először egy letét formájában kell jeleznünk, hogy az autót le akarjuk foglalni. Itt meg kell adnunk az autó árának 10%-át, illetve azt, hogy milyen pénznemben kívánunk fizetni. A letétnél természetesen a vásárlónak szám karaktereket kell begépelnie, ami nem kezdődhet 0-ás értékkel és nagyobbnak kell lennie mint 0, különben a program jelezni fogja neki, hogy nem megfelelő a megadott érték és ekkor ezt a tovább haladás érdekében kénytelen javítani. A fizetési opció megadásánál jelenleg csak a PayPal elérhető és elfogadott, ezért ennek megfelelően kell megadni itt is a begépelt bemenetet. A vásárlás céljánál három lehetőség közül tudunk választani, amik az eladás, a megbízás és a rendelés. Ezeknek komolyabb szerepe nincs, csupán a számlázásnál lehetnek érdekesek, hogy magánszemélyként vesz-e valaki autót vagy cégként és ennek megfelelően kerülnek feltűntetésre az adatok a nyugtánál. A leírásba pedig bármilyen olyan tényezőt, körülményt vagy megállapodást fel lehet tűntetni, ami a későbbiekben fontos lehet vagy a vásárlás részét képezi. Ha mindent helyesen adtunk meg, akkor a zöld gombra kattintva tudunk tovább lépni és ez átirányít minket a PayPal fiókunk bejeletkező oldalára. Itt a megfelelő email címünkkel vagy mobil számunkkal és jelszavunkkal kell bejelentkeznünk, majd ezt követően meg tudjuk adni a szállítási címet és azt, hogy mivel szeretnénk fizetni, ami jelen esetben a PayPal. Amikor mindent kitöltöttünk el tudjuk dönteni, hogy a vásárlást véglegesíteni szeretnénk-e vagy vissza is léphetünk és ilyen esetben érvénytelenné válik a vásárlásunk. Mindkét esetben egy üzenetet kapunk, ahol az eredmény fel van tüntetve és innen lehetőségünk van visszatérni a főoldalra.

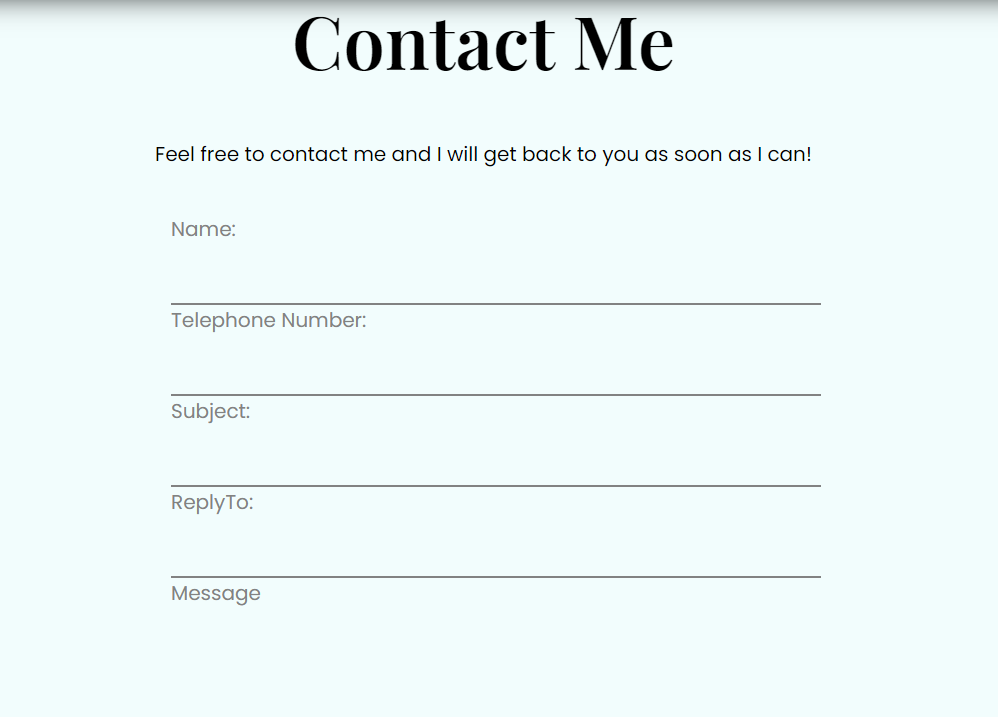
Amennyiben admin jogosultsággal lépünk be az alkalmazásba, akkor lehetőségünk van a garázsonként táblázatokban kilistázott autók törlésére és szerkesztésére is. Törlés során az adott jármű és az ahhoz tartozó adatok eltűnnek az aktuális garázsból, így a vásárlók számára az már nem lesz elérhető. Ez mellet lehetőség van a szerkesztésre is, amely azért hasznos, mert ha egy autó tulajdonságai megváltoznak, akkor azt nem kell kitörölni teljesen és újból felvinni az elérhető gépkocsik listájába, hanem elég csak a megváltozott adatot felülírni. Ennek köszönhetően sok időt spórolhatunk meg és kezelhetőbb lesz a webalkalmazás.



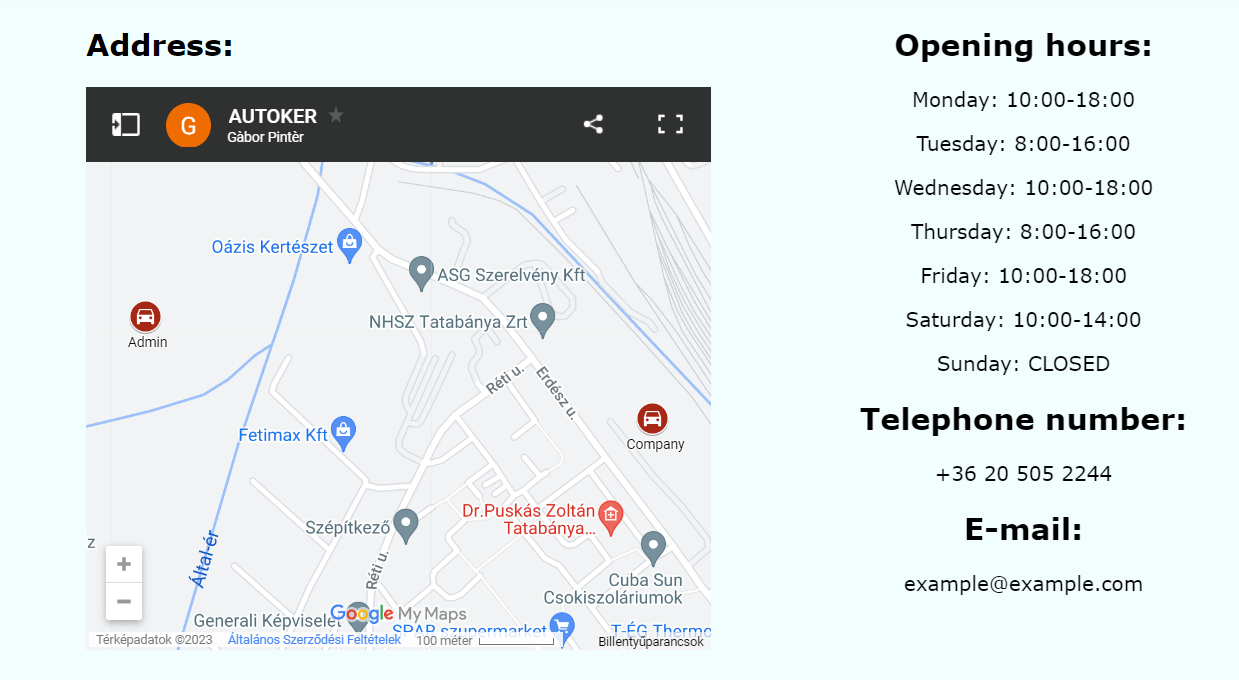
**Keresés:**

Ez szintén egy igen hasznos funkció, amely lehetőséget ad a felhasználók egyedi igényeinek a kielégítésére. Ha valaki nem kívánja végigböngészni az összes garázs összes elérhető járművét, akkor a feltűntetett szűrők segítségével megszabhatja azt, hogy milyen tulajdonsággal rendelkező gépkocsik jelenjenek csak meg. Lehetőség van a gyártó, a típus a minimum és a maximum ár szerint keresni és ezeknek megfelelően azok az autók fognak megjelenni, amelyek eleget tesznek az általunk megadott feltételeknek. Fonton megemlíteni, hogy amennyiben valaki több szempontot is szem előtt tart akkor van lehetőség arra, hogy a keresés során több paramétert is megadjon és az ezekkel a tulajdonságokkal rendelkező járművek fognak megjelenni számára.

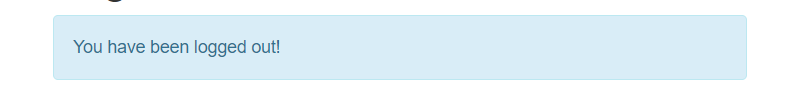
**Kapcsolat:**

Ennek az oldalnak köszönhetően tudunk egyszerűen kapcsolatba lépni az oldal üzemeltetőjével, aki segítségünkre lehet, ha valami nem világos vagy ha bármilyen kérdésünk merül fel. A beviteli mezőknél meg tudjuk adni a nevünket, a telefonszámunkat, a tárgyat és az üzenetünket. A sikeres email küldés érdekében minden mezőbe kell valamilyen értéket írnunk. Email címet azért nem kell megadnunk, mert az application.properties fájlban definiált email cím érték lesz hozzánk rendelve és így tudunk emailt küldeni.  A telefonszám megadása során egy minta alapján kell begépelnünk a karaktereket és a kezdő értéknek a plusz jelnek kell lennie majd 11 darab számnak kell követnie. Jelenleg az elfogadott értékek halmaza Magyarországra van szűkítve ezért az első két számnak 06-nak vagy 36-nak kell lennie míg a következő két érték 20 30 vagy 70 lehet.

**Térkép:**

Ezt a lehetőséget kiválasztva a fejlécben, képet kapunk a cégek és a központ elhelyezkedéséről a Google Maps-ben az újonnan megjelenő lap bal oldalán. A térképben van lehetőségünk nagyítani, kicsinyíteni illetve teljes képernyőre kirakni a megjelenő térkép részletet. A megosztás gomb segítségével egyszerűen el is lehet tárolni és tovább lehet osztani a megjelent címeket annak érdekében, ha a későbbiekben ezekkel rendelkezni szeretnénk vagy ha hivatkozni kívánunk rájuk. Emellett feltűntetésre kerültek a nyitvatartási órák is, így aki személyesen kívánja meglátogatni valamelyik telephelyet vagy a központot meg tudja tenni a megadott intervallumon belül. A gyorsabb kapcsolatfelvétel céljából a telefonszám és az email cím is kiírásra kerültek ezen az oldalon is a többi információ mellett.

**Kijelentkezés:**

Erre a gombra kattintva kijelentkezünk az autókereskedéses alkalmazásból és a bejelentkező oldalon találhatjuk magunkat. Innen vissza lépni nem tudunk a Spring Security miatt, tehát kénytelenek vagyunk újra begépelni az email címünket és jelszavunkat ha ismét használni kívánjuk az alkalmazás biztosította funkciókat. Kijelentkezést követően egy kék üzenet jelzi nekünk, hogy ezt valóban megtettük és újból a bejelentkező felületen találhatjuk magunkat.

Backend

A webalkalmazás belső működésének megvalósítása során számos technológia került felhasználásra, amelyek arra szolgálnak, hogy a program minél kiszámíthatóbb és egyszerűbben kezelhető legyen. Ilyen technológia például maga a Java, mint programozási nyelv, amelynek segítségével az alkalmazás megírásra került. Emellett a Spring és Spring Boot keretrendszer volt az, ami a .jar fájlok egyszerűbb kezelhetősége érdekében elérhetővé tesz egy úgynevezett pom.xml fájlt ahol a függőségek könnyedén tárolhatóak és módosíthatóak. A fejlesztői környezet ahol a kódsorok íródtak az Eclipse volt, ugyanis amióta programozással foglalkozom, ezt használom és számomra ez is legszimpatikusabb az átláthatósága és a billentyűkombinációk nyújtotta kényelem miatt. A fejlesztés során felhasznált építési automatizálási szoftver a Maven volt amit első sorban Java projektekhez használnak de lehetőség van más nyelven íródott projektek készítésére és kezelésére is például C#, Ruby, Scala stb. Az autók adatbázisban való eltárolása érdekében az src/main/resources mappában egy .sql kiterjesztésű fájlban lett a három különböző autókereskedés garázsa létrehozva. Ezek a garázsok feltöltésre kerültek néhány járművel, amik már megtalálhatóak a kínálatban és a továbbiakban ezekhez lehet hozzáadni vagy ezeknek az adatait lehet módosítani az elérhető kínálat alapján. A konkrét Sql lekérdezések és műveletek egy beépített osztály segítségével valósulnak meg, amelyben az alapértelmezett parancsok már definiálva vannak és nekünk csak ezeket kell kiegészítenünk, amennyiben változtatni szeretnénk rajtuk vagy egyedi lekérdezéseket kívánunk megvalósítani. Ezeken kívül, amit még fontosnak tartok megemlíteni az a csomagfájl formátuma. Mielőtt legeneráljuk az alkalmazásunkat a Spring Initializr odal segítségével két opció közül tudunk választani, amelyek a JAR és WAR fájlformátumok. Ezek azért hasznosak, mert ilyen módon egyszerűen tudjuk terjeszteni a kódunkat, hiszen minden egy fájlban van elhelyezve és a fejlesztői környezetben is csak egész egyszerűen importálnunk kell azt a fájlt amivel dolgozni szeretnénk. Végül megfigyelhető az is, hogy a webalkalmazásban fellelhetőek képfájlok is, amelyek arra szolgálnak, hogy felhasználóbaráttá és könnyen megérthetővé tegyék a programot és annak működését.

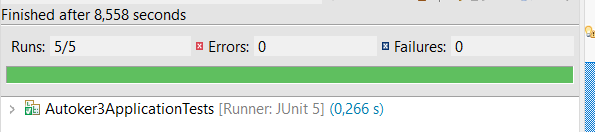
Az src/main/java mappában található az összes olyan java fájl, amelyek az alkalmazás belső működéséért felelősek. Itt különböző csomagok lettek létrehozva, amelyek segítenek elkülöníteni a más és más funkciót megvalósító kódsorokat. Az alapértelmezett csomag a com.example.AUTOKER3, amiben a futtatandó osztály található. Ebben jobb klikkel rákattintva a fájlra, majd a Run As lehetőség után a Java Application-t kell kiválasztani és így elérhető lesz az alkalmazás a megfelelő URL beírása után. A com.example.AUTOKER3.beans csomagban találhatóak az entitás fájlok, ahol egy adott entitást és annak a tulajdonságait fejtettük ki. A com.example.AUTOKER3.configuration csomagban vannak a konfigurációval kapcsolatos fájlok, amik definiálják azt, hogy az alkalmazásunk miként működjön. A com.example.AUTOKER3.controllers csomagban fellelhető fájlokban lévő metódusok segítségével van az leírva, hogy a webhelyen egy adott gombra kattintás után melyik oldalra kerüljünk át. A com.example.AUTOKER3.database és com.example.AUTOKER3.repository csomagban lévő osztályok segítségével mondjuk meg azt, hogy az adatbázisunkat hogyan érjük el és milyen módon tudjuk az oda betöltött adatokat újra felkeresni és rendelkezni velük. Végül pedig a com.example.AUTOKER3.service-ben van az összes olyan metódust leíró fájl, amik azt mondják meg, hogy egy adott lehetőségre rákattintva a webhelyen mi az aminek történnie kell.

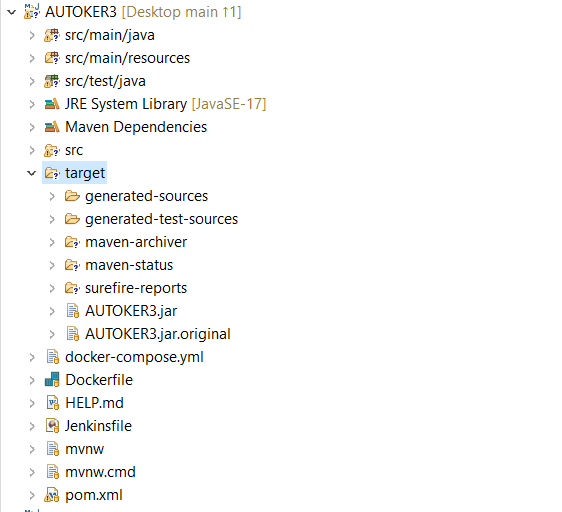
Az src/main/resources mappában vannak további olyan elemek, amik az alkalmazást kiegészítik és kezelhetőbbé, illetve felhasználóbarátabbá teszik. A static mappán beül kerültek elhelyezve az alkalmazásban fellelhető képek és itt vannak definiálva a .css fájlok is, amik azt mondják meg, hogy egy adott .html oldal miként jelenjen meg. A templates mappában van az összes .html kiterjesztésű fájl, amik a webböngészőben való megjelenítést szabályozzák. Az application.properties fájlban definiálásra kerültek olyan tényezők, amelyek az alkalmazás működését befolyásolják. Ezek a következők:

* Az adatbázis elérését meghatározó URL cím és az adatbázisunkhoz tartozó felhasználónév és jelszó
* Meg lett adva az is, hogy az alkalmazás hányas porton fusson, ami esetünkben a 8081
* Definiálva van egy frissítési művelet, ami megpróbál új oszlopokat, megszorításokat stb. hozzáadni, de soha nem távolít el olyan oszlopot vagy kényszert, amely korábban létezett
* Az is meg van határozva, hogy az alkalmazás első indításakor legyenek adatok, amik már alapból be vannak töltve annak érdekében, hogy ne egy üres adatbázist lássunk
* Az sql műveletek meg vannak jelenítve a console-on, ugyanis ez elősegíti a fejlesztést és átláthatóbbá teszi a programot fejlesztői oldalról
* A naplózási szint debug módra van állítva és ez kiírja az sql lekérdezéseket részletesebben is, amennyiben debug módban futtatjuk a programunkat
* Meghatároztuk azt is, hogy az adatbázisunk milyen nyelvjárást használjon a későbbi ebből fakadó esetleges hibák kiküszöbölésére
* Az email küldési funkció sikeres működése érdekében leírásra került az emailhez tartozó port, host, felhasználónév, jelszó és autentikációt elősegítő tulajdonságok is amellett, hogy a saját Google fiókunk kezelése során létrehoztuk a jelszót és elvégeztük a szükséges beállításokat
* A PayPal-os fizetés lehetővé tétele érdekében is felvételre került néhány tulajdonság az application.properties fájlban, amik az azonosító, titkos kulcs és működési mód

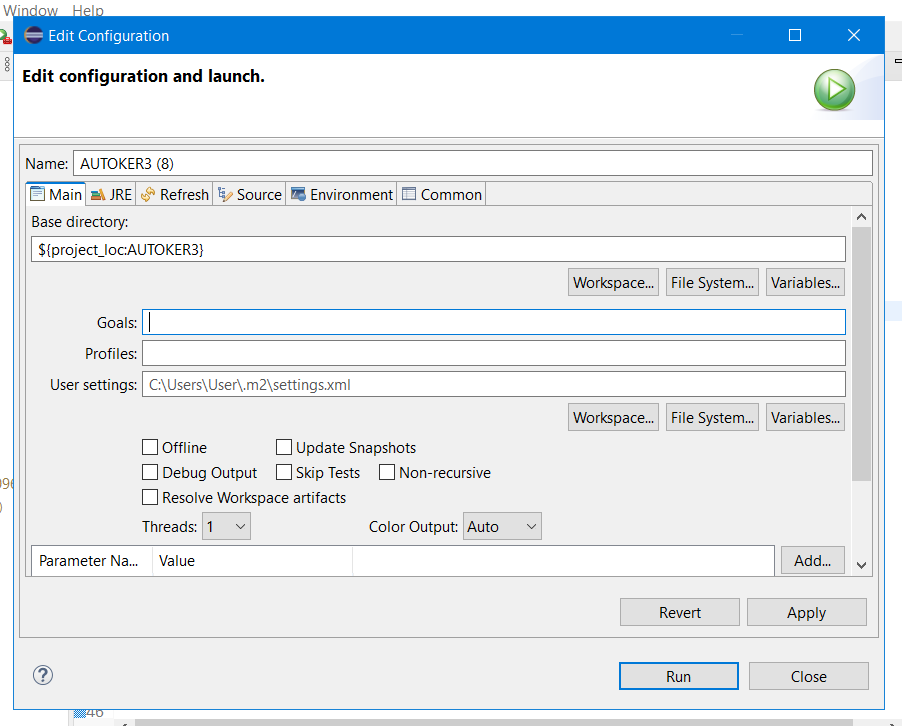
A legelső futtatás során annak érdekében, hogy a weboldalunkon ne üres garázsokat kapjunk ugyanúgy az src/main/resources mappában létrehozásra került egy .sql kiterjesztésű fájl is ahol 3 garázst vettünk fel és ezek autó adatokkal lettek feltöltve.

**JUnit tesztek:**

A JUnit egy úgynevezett teszt keretrendszer a Java programozási nyelvhez, amely lehetővé teszi az tesztek könnyű és hatékony írását és futtatását. A JUnit tesztek segítségével ellenőrizhetjük a különböző osztályok és metódusok helyes működését, így biztosítva az alkalmazásunk stabilitását és megbízhatóságát. A JUnit tesztek írása egyszerű, csak annyit kell tennünk, hogy elkészítjük a tesztosztályunkat, amely tartalmazza a tesztmetódusokat. A tesztmetódusokban ellenőrizzük az egyes metódusok helyes működését, a visszatérési értékeket és az esetleges kivételeket. A tesztek futtatása a JUnit keretrendszerrel a fejlesztési környezetben vagy a parancssorban történhet. Az ilyen jellegű tesztek írása és futtatása rendkívül fontos a Java alkalmazások fejlesztése során, mivel segítenek megbizonyosodni arról, hogy az alkalmazásunk helyesen működik, és ha változtatunk az alkalmazásunkon, a tesztek segítenek a hibák azonosításában és javításában.

****A webalkalmazás megfelelő és hatékony működése érdekében az src/test/java mappában tesztesetek találhatóak meg. Ezek sikeres lefutása azt biztosítja, hogy nem került semmilyen hiba bele a programkódba a fejlesztés során és a program az elvárt kritériumoknak megfelelően működik. Azt, hogy az adott osztályban a teszt eseteket futtatni tudjuk és tesztként is értelmezze a fordító, a @Test annotáció segítségével tudjuk elérni. Amennyiben valamilyen oknál fogva kivételek dobódnak, akkor a JUnit ezt hibaként jelenti és a teszt futása sikertelen lesz. Ha viszont nem dobódik kivétel, akkor a teszt sikeresnek mondható és egy zöld csík lesz az, ami jelzi nekünk, hogy minden rendben van. Bármilyen hiba előfordulása esetén pedig egy piros csík jelenik meg, ami felhívja a figyelmünket arra, hogy valami nincsen rendben és a kód további javításokra szorul. A JUnit teszteléssel kapcsolatban hasznos információk érhetőek el az alábbi linken keresztül: <https://en.wikipedia.org/wiki/JUnit>

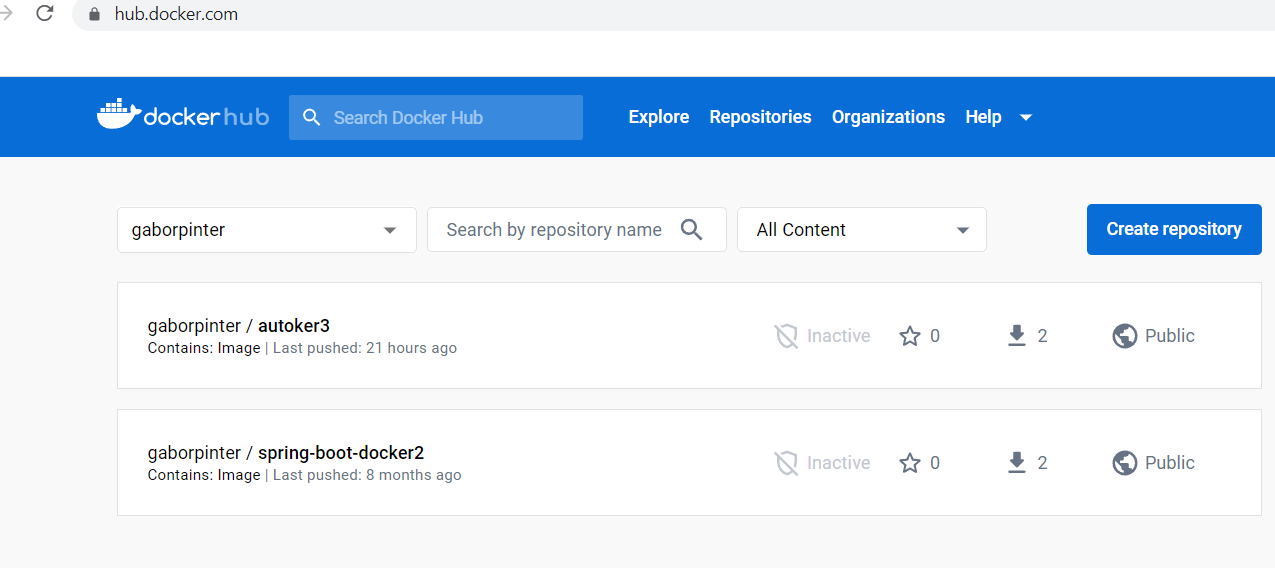
A JRE System Library és Maven Dependencies mappákban található meg az összes olyan .jar kiterjesztésű fájl amely nélkül a programunk nem tudná az elvárt működést produkálni. Ezek a fájlok teszik lehetővé hogy bizonyos annotációkat használjunk, működjenek a biztonsági funkciók, az adatbázisunkhoz hozzá tudjunk csatlakozni és alapvető nyelvi elemeket tudjunk használni követve a Java programozási nyelv szintaktikáját. Az itt jelen lévő fájlokat a pom.xml módosításával tudjuk szerkeszteni, illetve a kezdeti projekt létrehozása során határozhatjuk meg azt, hogy az alkalmazásunk hogyan jöjjön létre.

A Java projektre rákattintva jobb klikkel majd a Run As lehetőségnél a Maven Build opciót kiválasztva tudjuk felépíteni a projektünket. A fejlesztés során a Goals-hoz a clean install -X parancs segítségével történt az építés és így egy futtatható .jar fájl került elkészítésre a target mappába, ami a későbbi lépések során fel is fogunk használni. Több mappá és fájl is megtalálható a target mappán belül, de a számunkra lényeges az AUTOKER3.jar lesz, ami a Spring Boot alkalmazás nevéből és egy .jar kiterjesztésből tevődik össze.

**Dockerfile:**

A Dockerfile egy olyan fájl, amely leírja, hogy hogyan kell megépíteni egy Docker képet, amelyben az alkalmazásunk futni fog. A Dockerfile utasítások listáját tartalmazza, amelyekkel lépésről lépésre épül fel a kép. A Dockerfile soronként tartalmaz utasítást, amelyek közül az első általában egy FROM utasítás, amely meghatározza a környezetet amivel dolgozunk. Ezután definiálhatjuk a munkakönyvtárat a WORKDIR paranccsal, majd használhatjuk a COPY vagy a ADD parancsot az alkalmazásunk forrásfájljainak másolására a Docker konténerbe. A Dockerfile további utasításokat tartalmazhat a szükséges függőségek telepítéséhez, a beállításokhoz és a konfigurációhoz. Az EXPOSE paranccsal megadhatjuk az alkalmazásunk által használt portokat, majd a CMD vagy az ENTRYPOINT paranccsal indítjuk el az alkalmazást. A Dockerfile használható az alkalmazásunk teljes körű telepítési folyamatának automatizálására, ami lehetővé teszi számunkra az alkalmazásunk gyors, következetes és megbízható telepítését a különböző számítógépekre és infrastruktúrákra. A Dockerfile segítségével az alkalmazásunk egyszerűen skálázható, és könnyen kezelhető. Esetünkben a MySql adatbázis tulajdonságai is feltűntetésre kerültek a Dockerfile-ban az egyértelműség kedvéért és a későbbi hibák elkerülése érdekében. Itt az ENV parancs segítségével adjuk meg a környezeti változóinkat és ezekhez rendeljük hozzá a megfelelő értékeket. Az alkalmazásunknak a 9090-es portot, míg az adatbázisunknak a 3306-os portot adjuk meg a Docker kép létrehozása során annak érdekében hogy ez ne akadhasson össze a localhost-on futtatott egyéb dolgokkal. A Docker-ről további részletes leírás található a következő linken keresztül: <https://en.wikipedia.org/wiki/Docker_(software)>

**DockerHub:**

A DockerHub egy felhő alapú Docker konténerek regisztrációját és megosztását segítő szolgáltatás, amely még egyszerűbbé teszi a fejlesztők számára, hogy megoszthassák és használhassák Docker konténereiket. A DockerHub a Docker technológiára épít, amely egy konténereken alapúló virtualizációs technológia, így biztosítva azt, hogy alkalmazásainkat és szolgáltatásainkat gyorsan és hatékonyan csomagolhassuk és futtathassuk.

Ez gyakorlatilag egy olyan közösségi platform, ahol a fejlesztőként meg tudjuk osztani Docker konténereinket, és mások letölthetik és használhatják ezeket. A platformon található számos közösségi és hivatalos konténer, amelyek előre beállított konfigurációkkal rendelkeznek, így a fejlesztőknek csak a konténer telepítésével és konfigurálásával kell foglalkozniuk. Így érhető el például az is, hogy adatbázisokkal tudjunk dolgozni annak ellenére hogy a számítógépünkre nincsen telepítve semmilyen adatbázis jellegű alkalmazás. Könnyedén kereshetünk és találhatunk konténereket a közösségi könyvtárában. Ez egy olyan nyílt forráskódú könyvtár, amelyben bárki megoszthatja saját konténereit, és bárki letöltheti a már fent lévők közül bármelyiket.

A DockerHub szolgáltatásai közé tartozik az automatikus építés és tesztelés is. A fejlesztőknek van lehetőségük beállítani azt, hogy automatikusan építsék és teszteljék a konténereiket, amikor új verziókat töltöttek fel. Ez a funkció azért hasznos, mert segíthet abban, hogy tisztában legyünk azzal, hogy a konténereink megfelelnek-e az elvárásainknak, mielőtt azokat megosztanánk másokkal.

Továbbá gyorsan és egyszerűen tudunk telepíteni és konfigurálni konténereket majd megoszthatjuk ezeket másokkal. Nekünk is van lehetőségünk arra, hogy használjuk mások konténereit, és különböző konfigurációkat kipróbálhassunk anélkül, hogy maguknak kellene ezeket beállítanunk.

**Jenkinsfile:**

A Jenkinsfile egy olyan fájl, amely leírja, hogyan kell automatizálni a Jenkins által támogatott projektet. A Jenkinsfile egy úgynevezett Jenkins csővezeték leírás, amely szkriptnyelven írható, például a Groovy vagy a Python nyelven. A Jenkinsfile lehetővé teszi a projekt folyamatának menedzelését, ahol a folyamat az általános koncepció szerint minden lépésben automatizált, és az ellenőrzések és a tesztek automatikus végrehajtásán alapul. A Jenkinsfile definiál egy szkriptet, amely futtatja a projekt építését, tesztelését, integrációját és egyéb műveleteket, amelyek a Jenkinsben definiálhatók. A folyamat során a Jenkins környezeti változókat hoz létre, amelyek a futtatási időben használhatók. Ez egy fontos eszköz az egységesített CI/CD folyamatok kialakításához, amelyek lehetővé teszik a fejlesztők számára a kódjuk gyors és hatékony kiadását. A Jenkinsfile-t úgy hozhatjuk létre, hogy a kódunk mellé külön fájlba megírjuk jellemzően a projekt mappa alá és ezt a Github-ra feltöltjük, ahonnan a Jenkins ezt automatikusan felhasználja. Ezzel lehetővé tesszük a projekt folyamatának testreszabását a különböző környezetekben, amelyekben futtatva van. Például a tesztek futtatásának vagy a konfiguráció megváltoztatásának különböző beállításai lehetnek a tesztkörnyezetben és az éles környezetben. Az általam megírt JenkinsFile több lépést is magába foglal, amelyek a következőek:

* első lépésként a kódnak az elérhetősége van megadva, ami egy Github linken keresztül történik a master ágon
* ezután történik az építés, amely során ha minden teszt is sikeresen lefutott egy Build Success üzenetet kapunk és ezáltal jön létre a .jar fájlunk a target mappában
* ezt követően hozzuk létre a Docker képünket, amelyből később egy futtatható konténert tudunk készíteni
* majd bejelentkezünk a DockerHub fiókunkba a megfelelő jelszó és felhasználónév kombinációval
* amennyiben sikeresen megtörtént a bejelentkezés feltöltjük a létrehozott képünket a DockerHub platformra
* végül kijelentkezünk a fiókunkból és amennyiben minden sikeres volt akkor csak zöld téglalapokat látunk a Jenkinsben

A Jenkins működésével kapcsolatban további lehetőségek ismerhetőek meg a feltűntetett linken keresztül: <https://en.wikipedia.org/wiki/Jenkins_(software)>

**Pom.xml:**

A pom.xml egy Maven projekt konfigurációs fájlja, amely definiálja a projekt függőségeit, verzióit és build folyamatát. Az XML formátumban íródott fájlban definiálható a projekt neve, az alapértelmezett csomagolás típusa, a forrásfájlok és a tesztfájlok helye, a projekt függőségei, az építési folyamat során futtatni kívánt bővítmények és azok beállításai. A fájl fontos elemei közé tartozik a függőségek kezelése, amelyben definiálhatjuk az összes függőség verziószámát, valamint az építést. Az építés során lehetőség van az életciklus események felüldefiniálására, a bővítmények futtatásának szabályozására, és az alkalmazás becsomagolásának konfigurálására. Ezeken felül a pom.xml fontos szerepet játszik a Maven projekt menedzselésében és a projekt életciklus eseményeinek során történő automatikus végrehajtásában. A megfelelő pom.xml fájl kialakítása elengedhetetlen az alkalmazás hatékony és stabil fejlesztéséhez és üzemeltetéséhez.

**A Java, mint programozási nyelv:**

A Java egy magas szintű, osztályalapú, objektum-orientált programozási nyelv, amelyet úgy terveztek, hogy a lehető legkevesebb megvalósítási függőséggel rendelkezzen. Ez egy általános célú programozási nyelv, amelynek célja, hogy a programozók egyszerre írhassák, bárhol lehessen futtatni, ami azt jelenti, hogy a lefordított Java kód minden Java-t támogató platformon futhat újrafordítás nélkül. A Java-alkalmazásokat általában bájtkódra fordítják, amely bármely Java virtuális gépen (JVM) futhat, függetlenül az alapul szolgáló számítógép-architektúrától.

Szintaxisa hasonló a C-hez és a C++-hoz, de kevesebb alacsony szintű szolgáltatással rendelkezik. A Java futtatókörnyezet olyan dinamikus képességeket (például tükrözést és futásidejű kódmódosítást) biztosít, amelyek általában nem állnak rendelkezésre a hagyományos fordítási nyelveken. A mai napig az egyik legnépszerűbb programozási nyelv különösen a kliens-szerver webalkalmazások esetében.

A nyelvet eredetileg tölgynek hívták egy tölgyfa után. Később a projekt a Green nevet vette fel, és végül Java névre keresztelték, a Java kávéból, egy Indonéziából származó kávéfajtából. A Java szoftverek a laptopoktól az adatközpontokon, a játékkonzolokon át a tudományos szuperszámítógépekig mindenen futnak.

A nyelv létrehozásának öt fő célja volt:

- Egyszerűnek, objektumorientáltnak és érthetőnek kell lennie.

- Kiszámíthatónak és biztonságosnak kell lennie.

- Hardver függetlennek és hordozhatónak kell lennie.

- Nagy teljesítménnyel kell végrehajtani a feladatokat.

- Többszálúnak és dinamikusnak kell lennie.

Négy Java-kiadás definiált és támogatott, amelyek különböző alkalmazási környezeteket céloznak meg, és számos API-t szegmentált úgy, hogy azok valamelyik platformhoz tartozzanak. A Java API-k osztályai külön csoportokba vannak szervezve, amelyeket csomagoknak nevezünk. Minden csomag tartalmaz egy sor kapcsolódó interfészt, osztályt, alcsomagot és kivételeket.

A Java egyik tervezési célja a hordozhatóság, ami azt jelenti, hogy a Java platformra írt programoknak hasonlóan kell futniuk a hardver és az operációs rendszer bármely kombinációján, megfelelő futásidejű támogatással. Ez úgy érhető el hogy a Java nyelvi kód bájtkódként fordul nem pedig gépi kódként. A végfelhasználók általában a készülékükre telepített Java Runtime Environment-et (JRE) használják önálló Java-alkalmazások futtatására.

Ez a programozási nyelv automatikus szemétgyűjtőt használ a memória kezelésére az objektum életciklusában. A programozó határozza meg, hogy mikor jönnek létre az objektumok, és a Java futási környezet felelős a memória helyreállításáért, ha az objektumok már nincsenek használatban. Ha nem marad hivatkozás egy objektumra, az elérhetetlen memória jogosulttá válik arra, hogy a szemétgyűjtő automatikusan felszabadítsa. Ha egy nem létező objektum metódusait hívjuk meg, null pointer kivételt dobódik.

Végül, de nem utolsó sorban fontos megemlíteni a szintaxisát a nyelvnek, amelyre az objektum-orientáltság a jellemző. Ez azt jelenti, hogy minden kód osztályokon belül van írva, és minden adatelem objektum. Kivételt képeznek ez a szabály alól a primitív adattípusok, azaz egész számok, lebegőpontos számok, logikai értékek és karakterek, amelyek teljesítmény okokból nem objektumok. A nyelv részletesebb megismerése érdekében a következő link segítséget tud nyújtani: <https://en.wikipedia.org/wiki/Java_(programming_language)>

**Az Eclipse mint szoftver:**

Az Eclipse a számítógépes programozásban használt integrált fejlesztői környezet (IDE). Tartalmaz egy alap munkaterületet és egy bővíthető beépülő rendszert a környezet testre szabásához. A mai napig ez az egyik legnépszerűbb IDE a java fejlesztések megvalósításához. Többnyire Java nyelven íródott, és elsődleges felhasználása Java alkalmazások fejlesztésére szolgál, de más programozási nyelveken is használható alkalmazások fejlesztésére beépülő modulokon keresztül, például C++, Python, stb. Mindezek mellett dokumentumok írására is szintén használható. A felhasználók bővíthetik lehetőségeiket az Eclipse Platformhoz írt beépülő modulok telepítésével, például fejlesztői eszközkészletekkel más programozási nyelvekhez és saját beépülő moduljaikat is megírhatják, amelyekkel ilyen módon hozzájárulhatnak a cél orientáltabb és hatékonyabb fejlesztésnek.

Az Eclipse különböző verziói különböző tudományos vonatkozású neveket kaptak és ezeket jellemzően bolygókról nevezték el. Beépülő modulokat használ az összes funkció biztosításához a futásidejű rendszeren belül. Ezek lehetővé teszik az Eclipse Platform kiterjesztését más programozási nyelvekkel, például C és Python használatával, illetve a beépülő keretrendszer lehetőséget ad arra is, hogy együttműködjön hálózati alkalmazásokkal, mint például adatbázis-kezelő rendszerekkel. További információk lelhetőek fel a szoftverrel kapcsolatban az alábbi oldalon: <https://en.wikipedia.org/wiki/Eclipse_(software)>

**A JAR mint fájlformátum:**

A JAR (Java ARchive) fájlformátum a Java nyelvhez kapcsolódó archiválási formátum. Az ilyen típusú fájlok több Java osztályfájlt és metaadatot tartalmaznak. A JAR fájlok a Java platformon futtatva egy különálló Java Virtual Machine (JVM) környezetben futnak. Tartalmazhatnak Java osztályfájlokat, külső könyvtárakat, képeket, hangokat, konfigurációs fájlokat és más adatokat. Többnyire tömörített formátumban vannak, amely lehetővé teszi a fájlok kisebb helyigényét és egyszerűbb hordozhatóságát. Könnyedén telepíthetőek és futtathatóak az összes Java platformon.

A JAR fájlokat tartalmazó alkalmazások a Java Virtual Machine által kerülnek értelmezésre, így a fájlok függetlenek az operációs rendszertől. A fájlok létrehozása a Java archiváló (jar) segédprogram segítségével történik. Ez lehetőséget biztosít a fájlok összegyűjtésére és tömörítésére egy JAR formátumba, valamint az összes metaadat hozzáadására, amely az alkalmazás futtatásához nélkülözhetetlen. A fájlformátum lehetővé teszi a kód, a konfiguráció és az erőforrások egységes kezelését, valamint az alkalmazások függetlenségét az operációs rendszertől.

Bővebb információk olvashatóak a fájlformátumról a linkre kattintva: [https://en.wikipedia.org/wiki/JAR\_(file\_format)#:~:text=A%20JAR%20(%22Java%20archive%22,into%20one%20file%20for%20distribution.](https://en.wikipedia.org/wiki/JAR_(file_format)%23:~:text=A%20JAR%20(%22Java%20archive%22,into%20one%20file%20for%20distribution.)

**Apache Maven:**

Az Apache Maven egy szoftver projektmenedzsment eszköz, amely a Java platformra készült. Célja az, hogy megkönnyítse a fejlesztők számára a Java alkalmazások fejlesztését, építését, tesztelését és megosztását. A folyamatos egységesülés és a folyamatos szállítás eszközeinek alapvető eleme, és segít a projektmenedzsment és az erőforrások automatizálásában.

A Maven a szabványos projektstruktúrára épül, amely segít a fejlesztőknek a projektjük felépítésében és az erőforrások helyes konfigurálásában. A Maven által használt pom.xml fájl tartalmazza a projekt leírását, beleértve a függőségeket, az építési beállításokat és a tesztek futtatásához szükséges konfigurációt. Ezzel lehetővé teszi a fejlesztők számára, hogy könnyedén importálják és használják az előre elkészített függőségeket, például a Spring keretrendszert vagy a Hibernate-et, és egyetlen parancs segítségével elvégezzék az összes szükséges építési és tesztelési folyamatot.

Az Apache Maven az Apache Software Foundation által fejlesztett nyílt forráskódú projekt, amely aktívan karbantartott és támogató közösséggel rendelkezik. A Maven az egyik legnépszerűbb projektmenedzsment eszköz a Java platformon, és széles körben használják a fejlesztők és az iparágban dolgozók. A fejlesztőknek teret ad arra, hogy egyszerűen elkészítsék az alkalmazásukat, az erőforrásaikat és a függőségeket, majd ezeket beépítsék az alkalmazásukba. Az eredményként létrejött JAR (Java Archive) fájlok könnyen telepíthetők és futtathatók, és az összes függőség automatikusan letöltődik az internetről. Ilyen módon számos előnyt kínál a fejlesztőknek, például az automatikus függőségkezelést, a belső ellentmondásoktól mentes projektstruktúrát és az egységes építési és tesztelési folyamatot. Az Apache Maven kiterjedt dokumentációt, közösségi támogatást és fejlett funkciókat kínál, amelyek segítik a fejlesztőket a gyorsabb és hatékonyabb Java alkalmazások fejlesztésében és karbantartásában. Részletesebb leírás a következő linken található: <https://en.wikipedia.org/wiki/Apache_Maven>

**Spring keretrendszer:**

A Spring keretrendszer egy nyílt forráskódú, Java alapú keretrendszer, amely segít az alkalmazások fejlesztésében és tesztelésében. Az egyik legnépszerűbb Java keretrendszer, és széles körben használják a vállalati alkalmazásokban. Azért népszerű mert egy rugalmas és moduláris megközelítést kínál az alkalmazások fejlesztéséhez. A keretrendszer számos különböző modult tartalmaz, amelyek segítenek az alkalmazások építésében és tesztelésében. A Spring keretrendszer főbb moduljai közé tartozik például a Spring MVC, a Spring Security és a Spring Boot stb.

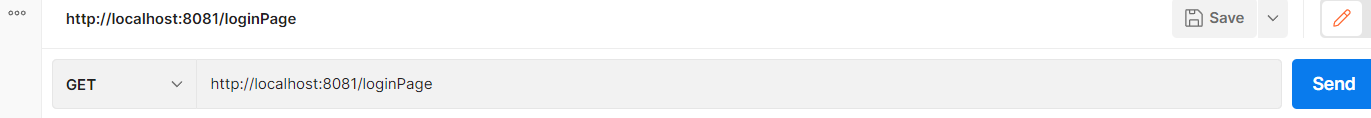
A Spring MVC (Model-View-Controller) teszi lehetővé a webalkalmazások fejlesztését. A modul a Model-View-Controller tervezési mintát követi, és lehetővé teszi az alkalmazások különválasztását az üzleti logikától és a felhasználói felülettől.

A Spring Security segítségével biztonságosabbá tehetők az alkalmazások, hiszen segítséget nyújt a felhasználók autentikációjában és az engedélyezések kezelésében.

A Spring Boot leegyszerűsíti a Spring alkalmazások építését és felgyorsítja a folyamatot. Olyan modulokat tartalmaz, amelyek előre konfigurált függőségeket és beállításokat adnak hozzá a projektünkhöz.

A Spring keretrendszer nagy előnye, hogy a moduláris felépítése miatt csak azokat a modulokat kell használni, amelyekre szükség van. Ezen kívül a keretrendszer támogatja a tesztelést, ami segíti a minőségi és megbízható alkalmazások fejlesztését. Összességében a Spring keretrendszer egy nagyon erős és rugalmas Java keretrendszer, amely széles körben használatos a vállalati alkalmazások létrehozása során. További ismeretekre tehetünk szert amennyiben a feltűntetett linkre kattintunk: <https://en.wikipedia.org/wiki/Spring_Framework>

**REST végpontok tesztelése:**

Abból a célból, hogy az alkalmazásunk kiszámítható legyen és érvényes URL címnél az ahhoz hozzárendelt oldalt kaphassuk vissza a Talend API Tester és a Postman segítségével is tesztelésre kerültek az elérési útjaink.

A Postman egy széles körben elterjedt kliens-alkalmazás, amely segíti az API-k fejlesztését, tesztelését és dokumentálását. Biztosítja a felhasználók számára, hogy HTTP kéréseket küldhessenek a szervereknek, majd megjelenítsék és feldolgozzák a válaszokat. A felhasználók felvehetik és menthetik a kéréseket, majd később újra használhatják őket. A Postman sokféle funkcióval rendelkezik, például tesztelési környezetekkel, változók használatával és automatizált tesztekkel. A program támogatja a szinkronizációt például a GitHub és más platformok használatával, így a felhasználók egyszerűen megoszthatják kéréseiket és kollegáikkal dolgozhatnak rajtuk együttműködve. Az alkalmazás felülete egyszerűen kialakított és könnyen használható, így mind a fejlesztők, mind az üzleti felhasználók számára ideális választás lehet az API-k teszteléséhez és dokumentálásához. Bővebb információkat lelhetünk fel a Postman-nal kapcsolatban az alábbi linken keresztül: <https://en.wikipedia.org/wiki/Postman_(software)>

A Talend API Tester szintén az API-k tesztelésében van segítségünkre. Az alkalmazásnak köszönhetően a felhasználók egyszerűen küldhetnek HTTP kéréseket, majd megjeleníthetik és feldolgozhatják a válaszokat. A Talend API Tester támogatja az egyéni beállításokat és a szinkronizációt is a Git és más platformok használatával. Az alkalmazással tudjuk a különböző tesztkörnyezeteket kezelni és azok automatizálására is van lehetőségünk, így jelentősen csökkenhet a tesztelési idő. A program felhasználóbarát felülete lehetővé teszi a könnyű navigációt. A Talend API Tester alkalmas lehet a fejlesztők, tesztelők és más szakemberek számára, akik szeretnék hatékonyabbá és egyszerűbbé tenni az API-k tesztelését. Az API-ok teszteléséről részletesebben olvashatunk, ha a következő linkre kattintunk: <https://en.wikipedia.org/wiki/API_testing>

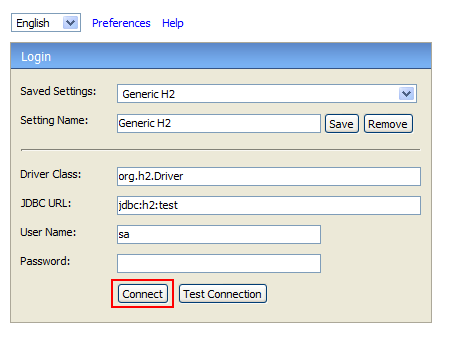
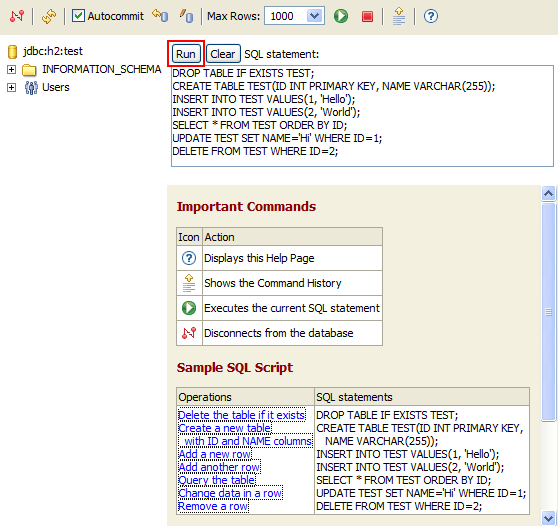
**SQL (Structured Query Language):**

Az SQL egy programozási nyelv, amelyet relációs adatbázis-kezelő rendszerekhez használnak adatok kezelésére és lekérdezésére. Képesek vagyunk vele az adatok struktúrájának definiálására, a táblák létrehozására, módosítására és törlésére. Az SQL-t használó alkalmazások könnyebbé teszik a felhasználók számára, hogy adatokat keressenek, szűrjenek, rendezzenek és csoportosítsanak, valamint különböző műveleteket végezzenek az adatokkal, mint például a csoportosítás, számítások és összekapcsolás.

Képesek vagyunk vele tranzakció kezelésre, indexelésre, adatok biztonsági mentésére és adatintegritás ellenőrzésre is. Ezek miatt nagy hatékonyságot és rugalmasságot tudunk biztosítani az adatkezelésben az alkalmazásunknak. Találkozhatunk az SQL nyelvvel üzleti alkalmazásokban, mobilalkalmazásokban és még sok más helyen is.

Adatbázis-kezelő rendszer által nyújtott különböző funkciók vannak biztosítva számunkra, mint például a SELECT, INSERT, UPDATE és DELETE parancsok. Ezekkel a kulcsszavakkal tudjuk elérni az adatok keresését, beszúrását, frissítését és törlését a táblákban. Az SQL együttműködik a relációs adatbázis-kezelő rendszerekkel, amelyek a táblákat és a kapcsolatokat használják az adatok tárolására és kezelésére. A táblák egymáshoz való kapcsolatát használva van lehetőségünk arra, hogy hatékonyan és gyorsan kérdezzünk le adatokat. További információ az alábbi linken: <https://en.wikipedia.org/wiki/SQL>

**H2 adatbázis:**

A program fejlesztése során és a tesztelési fázisban még az úgynevezett H2-es adatbázist használtam, hiszen itt még nem volt cél az adatok eltárolása véglegesen, csupán a működési mechanizmusokat vizsgáltam. A H2 adatbázis egy nyílt forráskódú, relációs adatbázis-kezelő rendszer, amely különösen alkalmas beépített alkalmazások, prototípusok és tesztelési célú projektek számára. Sok hasznos funkcióval rendelkezik, mint például azzal, hogy teljesen Java-ban íródott vagy, hogy a memóriában tárolt adatbázisok és a többi relációs adatbázis-kezelőhöz képest gyorsabb teljesítményt nyújt.

Széles körben elterjedt, mert egyszerű és könnyen kezelhető webes konzolalkalmazást kínál a felhasználóknak, amely lehetővé teszi az adatbázis-kezelő funkciók egyszerű használatát, például az adatbázis táblák létrehozását, módosítását és törlését, valamint az adatok beszúrását, frissítését és törlését. Mindezek mellett lehetőséget biztosít az adatbázis-objektumok, például táblák és indexek létrehozására és módosítására Java osztályok segítségével is, ami különösen hasznos a Java fejlesztők számára.

A H2 adatbázis támogatja a szinkronizációt és az adatok másolását is, így biztosítja a felhasználók számára, hogy az adatokat több helyszínen és eszközön is egyidejűleg használhassák és kezelhessék. Java API-t is kínál, aminek köszönhetően az adatbázis-kezelő funkciókat Java alkalmazásokba integrálhatják a Java fejlesztők.

**MySQL adatbázis:**

A Spring Boot alkalmazásunkhoz egy MySQL adatbázist kapcsoltam hozzá, annak érdekében, hogy az adatokat maradandóan és a program megállítása után is el tudjuk tárolni.

A MySQL Workbench segítségével hoztam létre az adatbázist, amely az alkalmazással össze van kapcsolva és ebben lelhetőek fel a táblák is, amelyeket az alkalmazás használ. Ez az alkalmazás grafikus felületet biztosít az adatbázisok és táblák kezeléséhez. Jelenleg hét darab tábla van használatban, amik az Eclipse-ben az AUTOKER3 projektben a beans csomagban található meg. A táblák a következők:

* car: ez tartalmazza az autók összes adatát és ez a kiinduló táblánk is
* eagle\_garage: ez egy konkrét tábla, amiben az ide felvett és elmentett autók vannak tárolva
* giant\_garage: ez egy konkrét tábla, amiben az ide felvett és elmentett autók vannak tárolva
* role: ez a tábla segít minket a bejelentkezés során a szerepek elkülönítésére, ami lehet felhasználó, cég vagy admin
* super\_garage: ez egy konkrét tábla, amiben az ide felvett és elmentett autók vannak tárolva
* user: itt vannak a beregisztrált felhasználók adatai eltárolva
* users\_roles: ez a tábla segít összekötni a felhasználókat és a szerepeket az azonosító alapján

A java alkalmazásban is jelzésre kerültek, hogy ezek az osztályok az adatbázisban tábla szerepet töltenek be az @Entity és @Table annotációk segítségével. Ezek mellett meghatározásra került az elsődleges kulcs az @Id annotációnak köszönhetően és ehhez automatikusan generált értéket rendeltünk hozzá a @GeneratedValue annotációval. Néhány oszlop külön névvel lett ellátva, ami a @Column annotáció megjelölés miatt történhetett meg. Az user és role táblák a @ManyToMany és @JoinTable annotációkkal kerültek összekapcsolásra.

*Autó tábla:*

|  |  |
| --- | --- |
| *Oszlop* | *Adattípus* |
| *id* | *int AI PK* |
| *colour* | *varchar(255)* |
| *dealership* | *varchar(255)* |
| *dealerships* | *tinyblob* |
| *make* | *varchar(255)* |
| *model* | *varchar(255)* |
| *new\_dealership* | *varchar(255)* |
| *price* | *double* |
| *vin* | *varchar(255)* |
| *photo* | *varchar(255)* |

*Sas\_garázs tábla:*

|  |  |
| --- | --- |
| *Oszlop* | *Adattípus* |
| *id* | *int AI PK* |
| *make* | *varchar(100)* |
| *model* | *varchar(100)* |
| *colour* | *varchar(100)* |
| *price* | *decimal(12,2)* |
| *vin* | *varchar(100)* |

*Óriás\_garázs tábla:*

|  |  |
| --- | --- |
| *Oszlop* | *Adattípus* |
| *id* | *int AI PK* |
| *make* | *varchar(100)* |
| *model* | *varchar(100)* |
| *colour* | *varchar(100)* |
| *price* | *decimal(12,2)* |
| *vin* | *varchar(100)* |

*Szerep tábla:*

|  |  |
| --- | --- |
| *Oszlop* | *Adattípus* |
| *id* | *bigint AI PK* |
| *name* | *varchar(255)* |

*Szuper\_garázs tábla:*

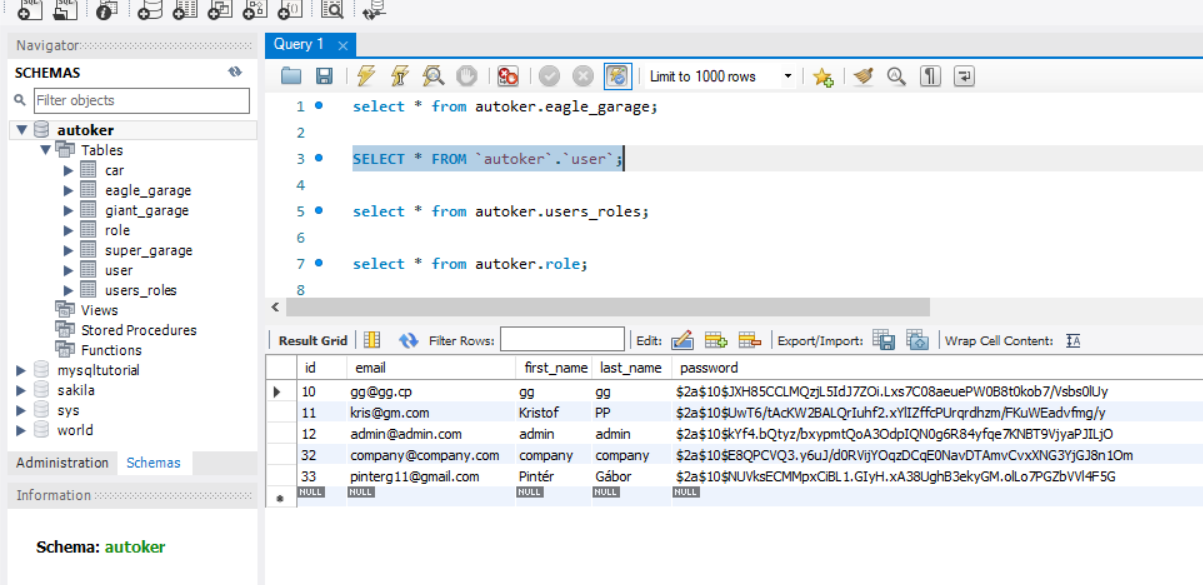
|  |  |
| --- | --- |
| *Oszlop* | *Adattípus* |
| *id* | *int AI PK* |
| *make* | *varchar(100)* |
| *model* | *varchar(100)* |
| *colour* | *varchar(100)* |
| *price* | *decimal(12,2)* |
| *vin* | *varchar(100)* |

*Felhasználó tábla:*

|  |  |
| --- | --- |
| *Oszlop* | *Adattípus* |
| *id* | *int AI PK* |
| *email* | *varchar(255)* |
| *first\_name* | *varchar(255)* |
| *last\_name* | *varchar(255)* |
| *password* | *varchar(255)* |

*Felhasználók-szerepek tábla:*

|  |  |
| --- | --- |
| *Oszlop* | *Adattípus* |
| *user\_id* | *bigint* |
| *role\_id* | *bigint* |

Választásom azért esett a MySQL adatbázisra, mert ez egy nyílt forrású relációs adatbázis-kezelő rendszer, amit széles körben használnak a fejlesztők és az üzleti felhasználók szerte a világon. Folyamatos fejlesztés és karbantartás alatt van, illetve támogatja az SQL nyelvet. Az alkalmazásban adatbázisokat és táblákat hozhatunk létre, amelyek tartalmazzák az adatokat. A táblák oszlopai az adattípusokat határozzák meg, amelyek között megtaláljuk az egész számokat, lebegőpontos számokat, karakterláncokat, dátumokat és időket. Az adatbázisok és táblák használatával a MySQL lehetőséget biztosít az adatok hatékony tárolására, lekérdezésére és kezelésére. Számos funkció és eszköz van biztosítva annak érdekében, hogy az adatokat megfelelően tudjuk kezelni és hibatűrő legyen az alkalmazásunk. A biztonságos hozzáférés és az adatok védelme érdekében a MySQL lehetővé teszi a felhasználói fiókok és jogosultságok kezelését. A teljesítménye és megbízhatósága miatt is nagyon népszerű, és lehetővé teszi az adatok hatékony tárolását és kezelését a legkülönbözőbb alkalmazásokban és rendszerekben. Bővebb leírást találhatunk a megadott linken: <https://en.wikipedia.org/wiki/MySQL>

**Képek:**

Abból a célból, hogy a webalkalmazásunk még jobban átlátható és felhasználó barátságos legyen képek kerültek elhelyezésre benne. Ezek a képek ábrázolják a garázst, az autókat és minden olyan szükségesnek gondolt dolgot, amik a fejlesztés során felmerültek és segítik az alkalmazást életszerűvé tenni. A képek .jpg és .jpeg formátumúak.

A JPEG (Joint Photographic Experts Group) az egyik legelterjedtebb képformátum, amely kiváló minőségű tömörítést biztosít a fotók számára. Széles körben használatos a digitális képek, fényképek, rajzok és más grafikai tervezési munkák tárolására és megosztására. A szabványos fájlkiterjesztése a .jpg vagy .jpeg, és a tömörítési arány beállításával lehetővé teszi a képméret kicsinyítését anélkül, hogy jelentősen befolyásolná a kép minőségét. Azért szeretik sokan mert kompatibilis szinte minden képnéző szoftverrel, és lehetővé teszi a képek gyors megosztását az interneten vagy a közösségi média platformokon keresztül is. Amennyiben jobban is a témába kívánjuk ásni magunkat a formátummal kapcsolatban akkor a feltűntetett linken ezt megtehetjük: <https://en.wikipedia.org/wiki/JPEG>

Front-end:

Az alkalmazást használók számára látható felületek leprogramozásában a HTML, a CSS és a Thymeleaf keretrendszer voltak a segítségemre. Az oldalak leprogramozása során arra törekedtem, hogy a színek és árnyalatok a megfelelő arányban legyenek feltűntetve és az elemek elhelyezése egy éles rendszerben is megállja a helyét. A felhasználói élmény javítása érdekében elhelyezésre került a fejléc, a lábléc és maga az oldalak tartalmi része, azaz a törzse. A fejlécen találhatóak meg azok a lehetőségek, amelyek az alkalmazásban elérhetőek, míg a láblécben a leggyakrabban használt közösségi médiás és üzenet küldési funkciót biztosító alkalmazások kerültek elhelyezésre. Ezeken felül email küldési és fizetési opciók is kialakításra kerültek és megfelelően kitöltve a szükséges beviteli mezőket ezeknek a használatára is lehetőségünk van.

**HTML (HyperText Markup Language):**

Az HTML egy olyan jelölőnyelv, amelyet a weboldalak fejlesztésére használnak. Az HTML kódok segítségével jelölik meg a weboldal tartalmát, szerkezetét és stílusát, és lehetővé teszik a böngészők számára, hogy a tartalmat megfelelően megjelenítsék. A dokumentumok tartalmazhatnak szöveget, képeket, linkeket, videókat, hangfájlokat és más multimédiás elemeket, amelyek mindegyike saját jelölőjelét használja. A kódok felépítése hierarchikus így ez lehetővé teszi a tartalom megfelelő strukturálását, például a címek, bekezdések, listák és táblázatok elhelyezését. A nyelvnek számos változata létezik és folyamatos fejlesztés alatt van annak érdekében, hogy minél szélesebb körben kiszolgálhassa a fejlesztői igényeket. A HTML az egyik olyan programozási nyelv, amely elengedhetetlen a webfejlesztéshez és számos ingyenes és fizetős szoftver áll rendelkezésre a kódok szerkesztéséhez és teszteléséhez. A nyelv részletesebb megismerésében segít a következő link: <https://en.wikipedia.org/wiki/HTML>

**CSS (Cascading Style Sheets):**

A CSS azért tud hozzájárulni az alkalmazásunk minőségéhez, mert tudunk vele stíluslapokat definiálni a weboldalak megjelenítéséhez. Két alapvető részből áll: a szelektorból és a deklarációból. A szelektorok határozzák meg azokat az elemeket, amelyekre a stíluslapot szeretnénk alkalmazni, a deklarációkkal pedig a megjelenítési tulajdonságokat tudjuk leírni. Ebből adódóan lehetőségünk van az oldalak megjelenésének egységesítésére és a tartalom elkülönítésére a megjelenéstől. A különálló stíluslapok rugalmasabbá teszik a weboldalakat, hiszen könnyen cserélhetők, módosíthatók és újrahasznosíthatóak.

A CSS használata nagyban segíti a webfejlesztőket a reszponzív weboldalak tervezésében is. Ez azt jelenti, hogy a weboldalak lehetővé teszik az oldalak megfelelő megjelenítését különböző méretű készülékeken is és nem fognak elcsúszni a használt elemek. Itt definiálhatóak többek között a betűtípusok és a méretek, a színek, a háttérképek, az elrendezési tulajdonságok és az animációk is. Biztosítva van a megjelenítési tulajdonságok öröklődésére is, ami még inkább elősegíti a stíluslapok hatékony használatát.

Használhatjuk más technológiákkal együtt, például HTML-lel így elősegítve a szép és felhasználóbarát weboldalak kialakítását vagy JavaScripttel ami pedig az interaktivitásnál és dinamizmusnál játszik szerepet. További hasznos tudnivalók találhatóak itt: <https://en.wikipedia.org/wiki/CSS>

**Thymeleaf:**

A Thymeleaf egy szerveroldali Java sablonmotor, amely képes arra, hogy a fejlesztők HTML fájlokba helyezzenek be dinamikusan generált adatokat, így készítve dinamikus webalkalmazásokat. Rendelkezésre áll egy egyszerű és magától értetődő szintaxis, aminek köszönhetően könnyedén tanulható és megértető. Számos lehetőség van biztosítva annak érdekében, hogy megkönnyítse a sablonok fejlesztését. Ilyenek például a származtatott attribútumok, amik biztosítják az attribútumok dinamikus cseréjét, a for-each ciklusok, amelyek megvalósítják a listák iterálását és az elágazások, amelyek választási lehetőséget biztosítanak az if-else feltételekkel, de további sok más funkció is helyet kapott.

Különösen előnyös a HTML és CSS nyelvek használatával, hiszen használhatjuk ezeket a nyelveket a sablonok fejlesztéséhez. A sablonmotor ugyanis a HTML kód ellenőrzését is elvégzi, így segít a hibák felkeresésében és javításában. Több módon is konfigurálható, ami azt biztosítja, hogy testreszabhassuk az alkalmazásunkat az igényeinknek megfelelően. Támogatja a Spring keretrendszert, így könnyen felhasználható a Spring alapú alkalmazásokban is. Részletesebb ismeretek megszerzése céljából érdemes meglátogatni a feltűntetett linket: <https://en.wikipedia.org/wiki/Thymeleaf>

Összefoglalás:

Összességében kimondhatom, hogy a projekt elkészítése számos előnnyel járt, mert a különböző funkciók leprogramozása során még mélyebb betekintést nyerhettem abba hogyan is épül fel egy ilyen web alkalmazás. Főként backend programozással foglalkozom, Java nyelven, így ott túlságosan sok újdonsággal nem találkoztam, mert napi szinten is használom a projektben látottakat, de mindenképpen egy érdekes feladat volt például a Paypal-os fizetés vagy az e-mail küldési lehetőség megvalósítása. A fejlesztés során voltak természetesen elakadásaim is, amikre amikor megtaláltam a megoldást még jobban rögzülni tudott az adott hiba kiküszöbölésének módja. Az egyik legnagyobb kérdést talán az jelentette, hogy pontosan mik is legyenek azok, amik az alkalmazásban helyet kapnak, de már meglévő és aktívan működő autókereskedések weboldalai jó alapot nyújtottak. A különböző keretrendszerek is nagy segítséget nyújtottak, mert a már előre megírt annotációk és metódusok közül használhattam fel azokat, amik szimpatikusak voltak vagy elég volt csak kisebb módosításokat eszközölni beléjük. A függőségek további egyszerű kezelését a Maven tette lehetővé a pom.xml segítségével, amiben elég volt csak néhány sort írni ahhoz, hogy egy .jar kiterjesztésű fájl beimportálásra kerüljön és az ebben definiáltakat már használni is tudtuk. Az előírt általános szintaxisnak megfelelően a különböző szolgáltatásokat leíró osztályok különböző csomagokba szerveztem szét és így a program még jobban elkülöníthető lett és ezáltal a statikusságot is ki tudtam küszöbölni.

A másik nagy feladat a frontend leprogramozása volt, hogy a háttérben már működő dolgok szemmel is láthatóak legyen egy programozásban nem jártas ember számára is. Ez azért okozott kicsit nagyobb problémát, mert eddigi tanulmányaim során nem ástam magam túlságosan mélyen bele a frontend-et megvalósító programozási nyelvekbe és az ezekhez kapcsolódó keretrendszerekbe. A sok újdonság ellenére úgy gondolom, hogy sikeresen meg tudtam valósítani ezt a feladatot is és ahol bármilyen fajta megakadásom volt ott utánanéztem annak, miként is kellene megvalósítani az aktuális feladatot. Ennek köszönhetően olyan tudásra tettem szert, amit már a későbbiekben is használni tudok és ez is nagy segítséget nyújtott abban, hogy még jobban megértsem egy ilyen jellegű alkalmazásnak a működését.

Végül pedig a konténerizációs és CI/CD technológiák is helyet kaptak az alkalmazásban. Ez azt jelenti, hogy az alkalmazásból készül egy egy .jar kiterjesztésű fájl is a target mappába és egy úgynevezett Docker Image is, amit használhatunk a későbbiekben. A DockerFile segítségével hivatkozunk a target mappában lévő .jar fájlunkra és itt mondjuk meg azt, hogy miként fogjuk felhasználni. Ezek a lépések a Jenkins-ben is fel vannak véve különböző lépésenként és amikor lefutottak a tesztek, elkészült a projekt építése majd az Image-ünk is létrejött akkor már nem teszünk mást, minthogy feltöltjük az elkészült képet a DockerHub platformra. Ez azért hasznos, mert így fejlesztő társaink is hozzáférhetnek és a Jenkins segítségével azt is meg tudjuk mondani például, hogy milyen gyakran akarjuk építeni automatikusan a csővezetékünket. Manuálisan bármikor tudjuk építeni és ez a megadott linken elérhető projekt állapotától függően fog sikeres vagy sikertelen lenni. Az én példámban a Github-on érhető el a projektem forráskódja és a Jenkins innen használja fel a kódsorokat.

IRODALOMJEGYZÉK

Link: url=<https://www.baeldung.com/spring-security-login>

Link: url=<https://www.baeldung.com/spring-controllers>

Link: url=<https://www.youtube.com/watch?v=PKcGy9oPVXg&ab_channel=JavaTechie>

Link: url=<https://github.com/wajzy/GKxB_INTM049/blob/master/css1/css1.pdf>

Link: url=<https://www.youtube.com/watch?v=oTJ89wcz5Ec&ab_channel=zaaimhalim>

Link: url=<https://github.com/Halim-Z/spring-boot-web-application/blob/master/src/main/java/com/halim/web/service/ProductService.java>

Link: url=<https://stackoverflow.com/questions/56721145/failed-to-convert-value-of-type-java-lang-string-to-required-type-int>

Link: url=<https://www.bezkoder.com/docker-compose-spring-boot-mysql/>

Link: url=<https://www.youtube.com/watch?v=6hMHziv0T2Y&ab_channel=B2Tech>

Link: url=<https://github.com/bezkoder/docker-compose-spring-boot-mysql>

Link: url=<https://www.youtube.com/watch?v=S2s28PCg4M4&t=423s&ab_channel=B2Tech>

Link: url=<https://charith.xyz/docker/dockerfile-environment-variables/>

Link: url=<https://www.w3schools.com/html/html_form_elements.asp>

Link: url=<https://www.w3schools.com/html/html_form_attributes.asp>

Link: url=<https://www.youtube.com/watch?v=PKcGy9oPVXg&t=276s&ab_channel=JavaTechie>

Link: url=<https://www.youtube.com/watch?v=e3YERpG2rMs&t=437s&ab_channel=JavaTechie>

# Mellékletek