AUTOKER

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 valósít meg a webalkalmazás, ami a megfelelő felhasználónévvel és jelszóval történő autentikációt igényel. 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.

Az alkalmazás működése:

A bejelentkezés megfelelő működéséért a Spring Security felelős ami egy hatékony és nagymértékben testreszabható 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. 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.

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. 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. 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.

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. A megvásárlás opcióra kattintva egy nyugtázó oldal tárulkozik elénk, amely azt biztosítja, hogy a vásárlásunk sikeres volt. Ekkor a megvásárolt autó adatai újból feltűntetésre kerülnek és egy nyugta formájában arról is kapunk információt, hogy mi az az összeg, amelyet be kell fizetnünk az autó után adó és egyéb költségekként. A megvásárolt jármű eltűnik a kínálatból és a továbbiakban a többi vásárló már nem fogja azt látni. Ezt követően nincs más teendőnk csak élvezni a megvásárolt jármű nyújtotta szolgáltatásokat és figyelni a saját magunk és környezetünk épségére.

Amennyiben admin jogosultsággal férünk hozzá az alkalmazáshoz 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.

JUnit tesztek:

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>

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.

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)>