Matematikai és Informatikai Intézet

Szoftverfejlesztés Java EE platformon

Készítette:

Kosárkó Ákos

Programtervező informatikus Bsc szak

Témavezető:

Tajti Tibor

tanársegéd

Eger, 2016

Tartalomjegyzék

Bevezetés

2015 elején foglalkoztatni kezdett, hogy munkába álljak és pénzt keressek. Ezen célért az interneten böngészve rátaláltam egy iskolaszövetkezet hirdetésére, amely biztos munkát ígért az általuk kínált képzés elvégzése után. A képzés tárgyaként a JAVA EE volt megjelölve, amiről első körben nem tudtam, hogy micsoda, annyit sejtettem, hogy a Java programozási nyelvhez kapcsolódhat. Második nekifutásra utánaolvastam az interneten, az jött le, hogy ez a Java programozási nyelv rengeteg külső osztálykönyvtárral kibővített "változata". Ma már tudom, hogy ez ennél jóval több. Visszatérve az iskolaszövetkezet hirdetésére, kapva az alkalmon, jelentkeztem a hirdetésükre, és a felvételi interjú után alkalmasnak találtak arra, hogy részt vegyek a képzésükön.

Így kerültem kapcsolatba a Java Enterprise Edition technológiával, amely, mint később világossá vált számomra, az iparban széles körben elterjedt és számos helyen használt eszköz. A szakdolgozat elkészítésével és az ehhez írt alkalmazás elkészítésével célom a technológia általam már ismert részének bemutatása, a technológia által nyújtott és számomra új megoldások megtalálása, és a gyakorlás. Mindezeket egy fiktív telekommunikációs cég számára készített alkalmazás, amellyel online ügyintézést tesznek lehetővé a régi és új ügyfeleik számára.

1. fejezet

Specifikáció

1.1. Az alkalmazás célja

Az alkalmazás célja lehetővé tenni a szolgáltató ügyfelei számára, hogy online intézhessék az ügyeiket. Ezen ügyintézésbe a cég szolgáltatásaira való előfizetés, szolgáltatáslemondás, számlabefizetés és hibabejelentés tartozik, emellett a szolgáltató az oldalán az új akciókról és a szolgáltatással kapcsolatos hírekről tájékoztathatja a leendő és már meglévő ügyfeleit.

1.2. Képernyők

Az alkalmazással végezhető műveletek természetesen láthatóak kell legyenek a használóik számára, erre a képernyők szolgálnak. Az alkalmazás webes böngészőkön keresztül érhető el a felhasználók számára, a képernyők is abban fognak megjelenni nekik. Az alábbiakban a felhasználók számára megjelenő képernyők leírását adom meg. A képernyők két csoportba bonthatók:

- Az egyik csoportba azok a képernyők tartoznak, amelyek bárki számára(azaz a nem bejelentkezett felhasználók számára is) jelennek meg. Az ezeken a képernyőkön elérhető információk és műveletek nincsenek egyénekhez kötve, egyedi információkat nem tartalmaznak. Ezzel együtt vannak olyan bárki számára elérhető oldalak, amelyek bejelentkezés után bővebb tartalommal rendelkeznek.
- Ennek ellenkezője áll fenn a bejelentkezett felhasználók számára elérhető oldalak esetén. Ezeken az oldalakon olyan információk és műveletek is elérhetőek, amelyek egyéniek, azaz bejelentkezett felhasználónként különböznek. Ilyen például a számlák képernyő, ahol nyilvánvalóan a felhasználó saját számlai kerülnek megjelenítésre. Ennek adatvédelmi okai (csak az előfizető láthassa a számláit, befizetésekkel kapcsolatos pénzügyeit) és praktikussági okai (egy felhasználót nem érdekel, hogy mások mely számlái várnak épp befizetésre) vannak.

1.2.1. Főoldal

Az alkalmazás kezdőoldala a főoldal, amely a bárki számára megjelenő oldalakhoz az elérést a menüpontokon keresztül. Az alkalmazás további funkciónak elérését lehetővé tévő "Bejelentkezési oldal" és "Regisztrációs oldal" elérését lehetővé tévő linkek szintén ezen az oldalon kapnak helyet. A szolgáltatott információkat tekintve ez az oldal egy rövid áttekintést nyújt a szolgáltató cég tevékenységéről miután üdvözlőszöveget jelenít meg a látogató számára.

1.2.2. Regisztrációs oldal

Az oldalon történő ügyintézéshez egy saját felhasználói fiókra szükséges. Ezt a fiókot a "Regisztrációs oldal"-on hozhatja létre a leendő ügyfél. Ehhez személyes adatainak megadására van szükség. A felhasználó az oldalon lévő form kitöltésével és beküldésével teheti ezt meg. A megadott adatok a regisztráció véglegesítése előtt egy validáción kell hogy keresztül menjenek. A validáció célja jelen esetben az, hogy elkerüljük egy előfizető többszöri regisztrációját a rendszerben. Ez a gyakorlatban azt jelenti, hogy például Nagy Ferenc, Eger, József Attila 125 szám alatt lakó ügyfél csak egyetlen fiókkal rendelkezzen. A validáció során, amennyiben ő már regisztrált felhasználója a rendszernek, ahelyett, hogy egy új fiókot kapna, a már meglévő fiókjához tartozó bejelentkezési adatokat kapja meg, amivel bejelentkezhet a "Bejelentkezési oldal"-on.

1.2.3. Bejelentkezési oldal

A "Bejelentkezési oldal" egy nagyon egyszerű funkciót ad, a már korábbiakban regisztrált ügyfél a fiókadatainak, felhasználónevének és jelszavának megadásával beléphet a fiókjába. Amennyiben nem ismeri a jelszavát, úgy jelszó-emlékeztetőt kérhet a regisztrációja során használt e-mail cím megadásával.

1.2.4. Bemutatkozó oldal

A "Bemutatkozó oldal"-on a szolgáltató hosszabb bemutatkozása olvasható, amely bővebb információkat ad a működéséről, felépítéséről és az általa nyújtott szolgáltatásokról.

1.2.5. Szolgáltatások

A "Szolgáltatások" felület egy fontos felület a képernyők között. Ezen a felületen találja meg az oldal látogatója a cég által nyújtott szolgáltatások listáját és jellemzőiket, úgy mint az ár, internetszolgáltatás esetén a sebesség, stb. Mivel a cég többféle szolgáltatást is nyújt(telefon, internet, kábeltv) ezért az oldal áttekinthetősége és a szolgáltatások

megjelenítésének struktúrája kiemelkedő fontosságú. Ezt úgy fogom kivitelezni, hogy a szolgáltatásokat kategóriák szerint listázhatja ki a látogató. A kategória elemei közül kiválasztott elem részletes leírása egy direkt ezen célt szolgáló felületrészen fog megjelenni.

A bejelentkezett felhasználók itt adhatják hozzá a bevásárlókosarukhoz a szolgáltatásokat, amelyekre elő kívánnak fizetni. A szolgáltatások megrendelését a bevásárlókosár tartalmának véglegesítése a "Megrendel" gomb megnyomása érvényesíti.

1.2.6. Akciók

Az "Akciók" felületen kap tájékoztatás az ügyfél az éppen futó akciókról. Ilyen akció például, hogy ha több szolgáltatásra is előfizet, akkor azt kedvezményesen teheti meg, ha meghosszabbítja az előfizetését, akkor egy havi szolgáltatás árát nem kell befizetni, vagy éppen kap egy egeret ajándékba.

1.2.7. Kapcsolat

A "Kapcsolat" oldalon a cég elérhetőségei olvashatóak, úgy mint a székhely, ügyfélszolgálati telefonszám és levelezési cím.

1.2.8. Személyes adatok

A "Személyes adatok" oldalon tekintheti meg a rendszerben szereplő adatait az ügyfél, és szükség esetén módosíthatja azokat. Az itt szereplő adatok a személyre vonatkoznak, az előfizetéseit ezen az oldalon nem menedzselheti. A személyre vonatkozó adatok alatt a következőket értem: név, születési adatok, lakcím, elérhetőségek.

1.2.9. Előfizetési oldal

Az "Előfizetési oldal" a szolgáltatások menedzselésére, azaz az aktív és már lejárt előfizetések listázására, adataik megjelenítésére szolgál. Itt tudja a felhasználó a futó szolgáltatás időtartamának meghosszabbítási igényét a szolgáltató felé jelezni, és a még aktív szolgáltatásait lemondani. Ez esetben ennek következményeiről egy felugró ablakban kap tájékoztatást. A könnyebb navigálás érdekében a szolgáltatásokhoz kapcsolódó számláihoz erről az oldalról is hozzáfér.

1.2.10. Számlabefizetés

Ez az oldal az előfizetések megléte miatt kiállított számlákat listázza ki. Minden egyes számláról megjelennek az információk, mint például a megrendelés, amiről a számla ki lett állítva, az összeg, ami a megrendelés értéke és a befizetési határidő, amellyel

bezárólag a számlát az ügyfél köteles befizetni anélkül, hogy "büntetést" kapna késedelmi díj formájában. A számla befizetésére bankkártyás fizetéssel van mód, amelyhez a bankkártyán szereplő adatokat a számlához tartozó befizetés gomb megnyomásával betöltődő oldalon kell megadni. Ezen az oldalon - természetesen egy kitalált, nem létező - pénzügyi szolgáltató felé történő kérés fut le, amely a bankkártyás fizetést szimulálja. Ez a fizetési kísérlet kétféle eredménnyel záródhat: Az ügyfél bankszámláján van elegendő pénz a számla kiegyenlítésére, és ez esetben - ismét hangsúlyozom egy nem létező - bankszámláról levonódik az összeg, és a számla kiegyenlített státuszba kerül. A másik eset az, amikor az ügyfél bankszámláján nem áll rendelkezésre elegendő pénz a számla befizetésére, így az befizetésre váró számlaként marad a számlák között. Mivel nincs valós pénzügyi szolgáltató a befizetések mögött, ezért a befizetés sikerességét vagy sikertelenségét egy véletlen szám generálással határozom meg, amelynek értéke dönti el, hogy sikeres vagy sikertelen legyen a befizetés. Mindkét esetben újra a "Számlabefizetés" oldal töltődik be újra.

1.2.11. Hibabejelentés

Az aktív szolgáltatással rendelkező felhasználóknak lehetőségük nyílik a szolgáltatással kapcsolatos hibák bejelentésére. Ezt a "Hibabejelentés" oldalon tehetik meg. Ez egy form kitöltésével és beküldésével történik meg, amely form tartalmazza a hiba leírását az szolgáltatás megnevezésétől a hiba észlelésének időpontján át jellegének bemutatásáig.

1.3. A szoftver elkészítésére használt technológiák, és az azt támgató eszközök

Egy alkalmazás elkészítése összetett folyamat. Ennek során nem elég a specifikációt ismerni, és az alkalmazott programozási nyelv használatában járatosnak lenni, ezek mellet számos olyan eszközt érdemes igénybe venni, amely gyorsítja, hatékonyabbá teszi, vagy éppen "biztonságosabbá" teszi az alkalmazás fejlesztését azáltal, hogy a szoftverkészítés folyamat egyes állomásain a forráskód állapotát eltárolja.

1.3.1. Java Enterpreise Edition

A szakdolgozatom témája a Java EE technológia bemutatása, így ez az a technológia, amiről részletesen írni fogok. Bevezetésül pár dolgot írok itt a technológiáról.

A Java Enterprise Edition API-k egy halmaza, amely API-k lehetővé teszik nagy skálázhatósággal rendelkező üzleti alkalmazások elkészítését. 13 API van, amelyek a technológia részeiül szolgálnak. Egy alkalmazás elkészítéséhez nem szükséges minden

API-t felhasználni. Például abban az esetben, amikor az alkalmazásunkban nincs szükség a kliensek közötti üzenetküldésre, abban az esetben a JMS -Java Message Service - használata nélkülözhető.

Természetesen vannak olyan API-k amelyek használata szinte nélkülözhetetlen a JEE technológia felhasználásával készített alkalmazások során. Ezen API-k az EJB és a JPA API-k, amik az üzleti logika és az adatbázisműveletek implementálására alkalmasak.

Az általam készített alkalmazás egy olyan alkalmazás, amelyik az API-k egy részét fogja használni.

1.3.2. WildFly Application Server

Az elkészült alkalmazások futtatásához egy olyan környezet szükséges, amely ismeri és kezeli az alkalmazás által használt API-kat. Ezeket a környezeteket alkalmazásszervereknek nevezzük. Feladatuk az alkalmazás számára biztosítani az általa használt API-kat, és erőforrás kezelést, erőforrás (ki)osztást végezni. Ilyen erőforrás (ki)osztás alatt értem például az alkalmazásszerver azon műveleteit, amelynek során az erőforrás konténerekbe annyi EJB példányt tárol, amennyi a kliensoldalról érkező kéréseket (optimálisan) képes kiszolgálni.

Én a WildFly alkalmazásszervert fogom használni, amely a JBoss alkalmazásszerver community verziójának új neve a 8.0-s verzió óta. Ez az alkalmazásszerver alkalmas a Java EE technológiával készült alkalmazások futtatására, mert implementálja a annak összes standardját.

1.3.3. Oracle Database 11g Express Edition

Az alkalmazás a felhasználókról, szolgáltatásokról, stb adatokat fog tárolni. Erre az Oracle ingyenesen használható adatbázisát, az Oracle Database 11g Express Edition-t fogom használni.

1.3.4. IntelliJ IDEA

A fejlesztői környezetnek az IntelliJ IDEA nevű eszközét választottam. Ezen választásom oka az, hogy tapasztalt programozóktól azt hallottam, hogy a forgalomban lévő fejlesztői környezetek közül ez a tapasztalataik alapján a legjobb. Na, majd kiderül, hogy tényleg így van-e.

1.3.5. Vaadin Framework

Az alkalmazás frontend részét Vaadin Framework használatával fogom elkészíteni. Ennek oka az, hogy Java-alapú frontend készítést tesz lehetővé. Ez azt jelenti, hogy a

böngészőben megjelenő oldalakat nem HTML használatával kell összeállítani, hanem a framework által nyújtott Java objektumokkal, és az így összeállított oldalak HTML5-ként generálódnak ki, amelyet a böngészők képesek megjeleníteni.

1.3.6. Maven

A szoftverfejlesztési iparban jelenleg sok programozó és sok cég van jelen. Előfordul, hogy az egyikük elkészít egy funkcióhalmazt implementáló kódot, és azt mások által felhasználhatóvá teszi. Az is előfordulhat, hogy egy specifikációt, elvbeli működést (pl JPA) több cég is implementál, méghozzá másként, majd kiadja azt másoknak felhasználásra. Ezen esetekben ezeken az implementációkat osztálykönyvtárak formájában érjük el egy központi repositoryban. Amennyiben használni szeretnénk ezen osztálykönyvtárakat, úgy azokat hozzá kell kapcsolnunk a projektünkhöz, mivel az függ azoktól. Ez esetben jön jól a Maven, amellyel a projektünk függőségei könnyen letölthetők az internetről és belefordíthatóak a projektünkbe. Ehhez mindössze egy megfelelő módon összeállított, projektfüggőségeket leíró fájlra van szükségünk.(pl pon.xml) Ez a fájl tartalmazza a projektben használt függőségek elérési helyét és azonosítóját, így azok egy áttekinthető, struktúrált és kényelmes módon integrálhatóak a projektünkbe.

1.3.7. Git

A program elkészítése során nagy segítséget nyújt egy verziókezelő rendszer használata, amely lehetővé teszi a fejlesztés során az alkalmazás aktuális állapotának mentését. Ennek több előnye is van. Egyik előnye az, hogy a fejlesztés egy pontjából több irányba is elágaztathatjuk a a fejlesztés további menetét. Ez amiatt hasznos, mert ha két megrendelő hasonló alkalmazást kér, akkor az alkalmazások funkciói közötti átfedéseket egy fejlesztési vonalon kezelhetjük és implementálhatjuk, a különbségeket pedig egy-egy külön ágon "adhatjuk" hozzá az átfedéseket tartalmazó részhez. Másik előnye a verziókezelésnek, hogy amennyiben azt találjuk, hogy az alkalmazás jelenlegi állapota nem a kívánt állapottal egyenlő, akkor könnyen átállhatunk egy olyan mentett állapotra, amelyik egy kívánt állapot. Harmadik előnye, hogy amennyiben osztott verziókezelő rendszert használunk, úgy a forráskódból több példány is van, így amennyiben az egyik példány elveszne, úgy tényleges veszteség nem ér minket, mivel van belőle másik.

Én egy a github-on létrehozott repositoryt és a Git nevű verziókezelő rendszert fogom az alkalmazás megírása során használni

2. fejezet

A Java Enterprise Edition

2.1. Szakasz címe

A Java EE API-k egy halmaza.

2.1.1. Alszakasz címe

3. fejezet

Az alkalmazás

3.1. Az adatbázismodell

3.2. A projekt összeállítása

A projekt vázának egy Vaadin archetípust, nevezetesen a vaadin-archetype-application archetípust használom. Ez az archetípus egy kiindulási alapot szolgáltat, amelyhez az EJB-k, entitások és egyéb Java objektumok, illetve a konfigurációs fájlok, mint például a persistence.xml, hozzáadása az alkalmazás elkészüléséhez vezet. Ahhoz, hogy ezen archetype alapján egyszerűen létrehozzam a projektet, egy új projektet hozok létre, amely egy maven-es projekt, és a projekt létrehozása során megadom, hogy archetype alapján jöjjön létre a projekt. Ehhez meg kell adni az archetype jellemzőit: groupID, artifactId, és a verzió. Mindezek a Vaadin honlapján elérhetőek. . Jelen esetben ezek a következőek: groupId: com.vaadin, artifactId:vaadin-archetype-application version: 7.6.2 . Az archetype kiválasztása után a saját alkalmazásunk groupId-ját és artifactID-ját kell megadnunk. Én groupdId-nak a com.szakdolgozat, artifactId-nak pedig a szer-t választottam. Ezen adatok megadása után az IDE letölti a maven központi repository-jából a archetype-ot, mint függőséget, és ez alapján legenerálja a projektet. Ezután kézhez kapunk egy alap projektet, ami mindössze egy UI java objektumok, egy gwt.xml-t és néhány kinézetet leíró .scss-t tartalmaz, egy struktúrált mappaszerkezetben.

Ezután az alkalmazás IDE-ből deployolhatóságát állíttam be. Ehhez egy plugin-t használok, amely alkalmazásszerver specifikus. Ez a plugin maven-nel könnyen hozzáadható az alkalmazásunkhoz. A

3.1. kód. WildFly plugin

```
1 <plugin>
2         <groupId>org.wildfly.plugins</groupId>
3         <artifactId>wildfly-maven-plugin</artifactId>
4         <version>1.1.0.Alpha6</version>
5         <configuration>
```

```
6                   <port>10000</port>
7                   <name>szer.war</name>
8           </configuration>
9   </plugin>
```

sorok pom.xml-hez hozzáadásával a plugin letöltésre kerül, és a projekt részévé válik. Ezután a Maven toolboxban megjelenik a pluginok között, és a wildfly:deploy opció indításával máris deployolásra kerül az alkalmazásszerverünkre. A port és name konfigurációs attribútumok beállításával finomhangolhatjuk a deployolás tulajdonságait. Jelen esetben a 10000-es porton futó administration consol-ú szerverre fog történni a deploy - melynél jelen esetben maga a szerver a 8090-es porton fut - és szer.war néven fog az alkalmazás deployálásra kerülni. Ez utóbbi azt vonja magával, hogy az alkalmazást a localhost:8090/szer URL-en fogjuk elérni.

Ezt követően az első lépés az adatbázis elérést lehetővé tévő persistence.xml fájl hozzáadása a projekthez a feladat, amelyet az src mappa alatt nehény mélységben található resources almappában létrehozandó META-INF mappában kell elhelyezni. Ez az xml file tartalmazza az adatbázis eléréséhez szükséges információkat. Jelen esetben a DataSource nevét, amely az adatbáziskapcsolat kialakításához szükséges információkat tárolja: az adatbázis címe, a bejelentkezéshez szükséges felhasználónév és jelszó. Emellett a ami számunkra fontos ebben az xml-ben, az a table generation strategy. Egy jól megválasztott stratégiával gondoskodhatunk az adatok biztonságáról. Én az update értéket választottam. Ez a stratégia az alkalmazás indulásakor megvizsgálja az adatbázis séma tartalmát, és amennyiben nem talál benne a projektben definiált entitás számára táblát, létrehozza azt.

Az ORM műveleteket a JPA fogja végezni. Ez az entitymanager-en keresztül végzi a munkáját. Ahhoz hogy ezt használni lehessen, a JPA-t függőségként hozzá kel adnunk a proketünkhoz. Ezt a pom.xml szerkesztésével tehetjük meg:

3.2. kód. Hibernate függőségek

```
1   <dependency>
2           <groupId>org.hibernate</groupId>
3           <artifactId>hibernate-core</artifactId>
4           <version>4.0.1.Final</version>
5   </dependency>
6   <dependency>
7           <groupId>org.hibernate</groupId>
8           <artifactId>hibernate-validator</artifactId>
9           <version>4.2.0.Final</version>
10  </dependency>
11  <dependency>
12          <groupId>org.hibernate.common</groupId>
13          <artifactId>hibernate-commons-annotations</
                artifactId>
```

```
14          <version>4.0.1.Final</version>
15          <classifier>tests</classifier>
16  </dependency>
17  <dependency>
18          <groupId>org.hibernate.javax.persistence</
                groupId>
19          <artifactId>hibernate-jpa-2.0-api</artifactId>
20          <version>1.0.1.Final</version>
21  </dependency>
22  <dependency>
23          <groupId>org.hibernate</groupId>
24          <artifactId>hibernate-entitymanager</
                artifactId>
25          <version>4.0.1.Final</version>
26          <exclusions>
27                  <exclusion>
28                          <groupId>dom4j</groupId>
29                          <artifactId>dom4j</artifactId>
30                  </exclusion>
31          </exclusions>
32  </dependency>
```

A fenti kód alapján a maven a központi repositoryból letölti a szükséges JAR-okat, és hozzáadja a projektünkhöz. Így már fogjuk tudni használni a JPA-t.

3.2.1. Alszakasz címe

4. fejezet

Összegzés

4.1. Szakasz címe

4.1.1. Alszakasz címe

Irodalomjegyzék

[1] Szerző: Cím, Kiadó, Hely, évszám.