Matematikai és Informatikai Intézet

Szoftverfejlesztés Java Enterprise platformon

Készítette:

Kosárkó Ákos

Programtervező informatikus Bsc szak

Témavezető:

Tajti Tibor

tanársegéd

Eger, 2016

Tartalomjegyzék

1. Az alkalmazás bemutatása **4**

1.1. Az alkalmazás működtetésének célja 4

1.2. Képernyők 4

1.2.1. Főoldal 5

1.2.2. Regisztrációs oldal 5

1.2.3. Bemutatkozás 5

1.2.4. Szolgáltatások 5

1.2.5. Kapcsolat 6

1.2.6. Számlák 6

1.2.7. Hibabejelentés 6

1.2.8. Bejelentett hibák 6

1.3. A szoftver elkészítésére használt technológiák, és az azt támgató eszközök 6

1.3.1. Java Enterpreise Edition 7

1.3.2. WildFly Application Server 7

1.3.3. Oracle Database 11g Express Edition 7

1.3.4. IntelliJ IDEA 8

1.3.5. Vaadin Framework 8

1.3.6. Maven 8

1.3.7. Git 8

2. A Java Enterprise Edition **10**

2.1. Szakasz címe 10

2.1.1. Alszakasz címe 10

3. Az alkalmazás **11**

3.1. Az adatbázismodell 11

3.2. A projekt összeállítása 11

3.2.1. Alszakasz címe 13

4. Összegzés **14**

4.1. Szakasz címe 14

4.1.1. Alszakasz címe 14

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

Az alkalmazás bemutatása

1.1. Az alkalmazás működtetésének célja

Az alkalmazás működtetésének célja lehetővé tenni a szolgáltató ügyfelei számára, hogy online intézhessék az ügyeiket. Ezen ügyintézésbe a cég szolgáltatásaira való előfizetés, számlabefizetés és hibabejelentés tartozik, emellett az alkalmazást a szolgáltató ügyintézői szintén használhatják, akik az ügyintézői jogosultságuk miatt más tartalomhoz férnek hozzá, mint az ügyfelek. Ők látják az összes bejelentett hibajegyet, amikhez a hiba elhárításának megoldását fűzik hozzá a hibabejelentés lezárásaként.

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) megjelennek. 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 vannak: egy előfizető számláit csakis ő maga

láthassa, az alkalmazás többi felhasználója ne. A bejelentkezett felhasználók típusoktól függően (ügyfél illetve ügyintéző) más felületeket érnek el: egy ügyfél hibabejelentő oldalt ér el, míg az ügyintéző a bejelentett hibákat listáztathatja ki, és kezelheti.

1.2.1. Főoldal

Az alkalmazás kezdőoldala a főoldal, amely a bárki számára megjelenő oldalakhoz közé tartozik. Ez az oldal egy rövid üdvözlő szöveget jelenít meg, a bárki számára elérhető menüpontok gombjaival együtt.

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, amelyre a képernyő jobb felső sarkában lévő login-box "Regisztráció" linkre kattintással jut el. A fiók létrehozásához személyes adatainak megadására van szükség, amely a form mezőinek kitöltésével végzendő el. A megadott adatok a regisztráció véglegesítése előtt egy validáción kell hogy keresztül menjenek. A validáció célja jelen esetben az, hogy elkerüljük olyan értékek megadását, amik nem lehetnek valósak, például egy számjegy a névben.

1.2.3. Bemutatkozás

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

1.2.4. 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 azok jellemzőit, 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. Éppen ezért a szolgáltatások listája szolgáltatástípusonként megjelenítve áll rendelkezésre az ügyfelek számára. A bejelentkezett felhasználók itt adhatják hozzá a bevásárlókosarukhoz a szolgáltatásokat, amelyekre elő kívánnak fizetni. A bevásárlókosárba tett szolgáltatások megrendelését a "Megrendel" gomb megnyomásával eszközölheti az ügyfél, ami után a szolgáltatási végpont adatait kell hogy megadja. A megrendelt szolgáltatások az itt megadott címre kerülnek bekötésre.

1.2.5. Kapcsolat

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

1.2.6. Számlák

Ez az oldal az előfizetések megléte miatt kiállított számlákat listázza ki. Az ügyfél minden egyes befizetetlen számláját kilistázásra kerül ezen a felületen, számlanévvel, összeggel és befizetési határidővel együtt. 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 a számla befizetett 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.

1.2.7. Hibabejelentés

Az aktív szolgáltatással rendelkező felhasználóknak lehetőségük nyílik a szolgáltatással kapcsolatos hibák bejelentésére. Ezt a "Hibabejelentés" oldalon tehetik meg. Az észlelt hiba részleteit egy form kitöltésével adhatják meg, amely form beküldésével az rögzítésre kerül, és az ügyintézők számára láthatóvá válik.

1.2.8. Bejelentett hibák

Ez egy az ügyintézők számára elérhető felület, amin a még ki nem javított hibákról szóló bejelentések érhetőek el. Amennyiben egy hiba kijavításra került, azt az ügyintézők a megoldást eredményező munkálatok rögzítésével nyugtázhatják.

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, 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 az alkalmazásszerverek. 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 annak összes standardját.

1.3.3. Oracle Database 11g Express Edition

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

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 az egyik legjobb. Ezen választás hozománya, hogy egy újabb fejlesztői környezetet ismerek meg, a főiskolai tanulmányaim során használt Microsoft Visual Studio és NetBeans mellett.

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, elérhetővé kell tennünk ezeket a projektünk számára, mivel az függ azoktól. Ez esetben jön jól a Maven, amellyel a projektünk függőségei könnyen letölthetők az internetről és hozzáadhatók a projektünkhöz. Ehhez mindössze egy megfelelő módon összeállított, projektfüggőségeket leíró fájlra van szükségünk.(pl pom.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 az esetben, ha az egyik példány elveszne, úgy tényleges veszteség nem ér minket, mivel a projekt kódja máshol is tárolva van.

Én egy a github-on létrehozott repositoryt és a Git nevű verziókezelő rendszert használom az alkalmazás elkészítése során.

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[1], 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 rendelkezésünkre áll egy kiindulási projekttartalom, ami mindössze egy UI java objektumot, egy gwt.xml-t és néhány a kinézetre(forntend) vonatkozó .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 mavennel könnyen hozzáadható az alkalmazásunkhoz. A

3.1. kód. WildFly plugin

```
1 <plugin>
2         <groupId>org.wildfly.plugins</groupId>
```

[1] `https://vaadin.com/maven#dependencies`

```
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, amelyet az src mappa alatt néhány 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 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. Abban az esetben pedig, amikor egy entitásosztály tagváltozóinak típusa és/vagy a tagváltozókhoz beállított adatbázisbeli jellemzők az adatbázisbeli táblafelépítéssel nincsenek szinkronban, úgy a tábla módosításra kerül, hogy annak jellemzői az entitásosztályban beállított jellemzőket tükrözzék. Például, ha egy Java osztály, ami egy entitást ír le rendelkezik egy int típusú és mennyiség nevű tagváltozóval, aminek az entitáshoz tartozó táblában nincs megfelelő mező, úgy a tábla egy "ALTER TABLE ..." DDL utasítás automatikus lefutásával módosul, és bővül egy mennyiség nevű mezővel.

Az ORM műveleteket a Hibernate, egy JPA implementáció fogja végezni. Ez az entitymanager-en keresztül goldozik. Ahhoz hogy ezt használni lehessen, függőségként hozzá kel adnunk a projektünkhoz. 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 Hibernate-t, amivel az objektumorientált programunk entitásainak világa és a "rekord orientált" adatbázisunk rekordjainak világa közötti átjárást tesszük lehetővé.

3.2.1. Alszakasz címe

4. fejezet

Összegzés

4.1. Szakasz címe

4.1.1. Alszakasz címe

Irodalomjegyzék

[1] Szerző: Cím, Kiadó, Hely, évszám.