XVIII. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)
Kolozsvár, 2015. május 21-24.

Szerver oldali Java technológiák vállalati rendszerek fejlesztéséhez.

Esettanulmány: a SkillMaster projekt

Szerző:
Ilonka Csaba
Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, informatika szak, III. év

Témavezető:
Dr. Simon Károly
Egyetemi adjunktus, Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, Magyar Matematika és Informatika Intézet
Projektmenedzser, Codespring Kft.

Kivonat

A dolgozat témája a SkillMaster projekt, valamint a hozzá tartozó webes kezelő-, illetve adminisztrációs felület.

A SkillMaster projekt célja egy webes alkalmazás megvalósítása, amely alkalmazottak szakismereteinek menedzselését teszi lehetővé vállalatok számára, továbbá lehetőséget teremt szakismeret, illetve továbbtanulási igény szerinti szűrésre, munkacsoportok összeállítására.

Az alkalmazottak önértékelés és feletteseik értékelése alapján kerülnek rangsorolásra a profiljukhoz rendelt szakterületeken belül. A vállalat menedzsmentje globális kimutatásokat kap az alkalmazottak felkészültségi szintjéről, illetve a szakemberek számáról adott témában, valamint az érdeklődési és továbbtanulási igényekről.

A rendszer SaaS (Software as a Service) megoldásként, egy központi szerveren keresztül is elérhető több vállalat, illetve intézmény számára. Emellett lehetőséget biztosít dedikált módon, saját infrastruktúrára történő telepítésre is.

Tartalom

Bevezető

A dolgozat témája a Java-alapú vállalati rendszerek fejlesztésénél alkalmazható technológiák, keretrendszerek és eszközök feltérképezése és bemutatása a SkillMaster projekt megvalósítása által.

A SkillMaster projekt célja egy webes alkalmazás megvalósítása, amely egy vállalat vagy szervezet alkalmazottai, illetve tagjai számára szakismeretek menedzsmentjét teszi lehetővé. A szakismeretek feltérképezése önértékelésen alapszik, amely kiegészíthető vezető beosztásban lévő alkalmazottak, menedzserek által adott értékelésekkel. A szakismeretek nyilvántartása mellett a rendszer lehetőséget nyújt különböző szakterületek iránti érdeklődés és továbbtanulási igény felmérésére is. Az intézmény vezetőségének az így kapott globális kimutatások alapján lehetősége nyílik az alkalmazottak szakismereteinek ismeretében döntéseket hozni, célzott munkacsoportokat összeállítani, továbbá az érdeklődési köröknek leginkább megfelelő képzéseket, illetve továbbképzéseket szervezni.

A SkillMaster projekt alapötlete nem új keletű, léteznek hasonló funkcionalitást biztosító eszközök. Ezeknek a piacon lévő eszközöknek a képességeit továbbgondolva, a SkillMaster az említett rendszerek bizonyos hiányosságait igyekszik pótolni. A SkillMaster szempontjából az említett eszközök közül talán leginkább a SkillsBase[1] rendszert érdemes kiemelni, amely egy szintén web platformon futó, funkcionalitásainak szempontjából nagyon hasonló eszköz. A SkillsBase egyetlen központi szerveren keresztül biztosítja szolgáltatását minden ügyfele számára. Ez a tulajdonság kis cégek, non-profit szervezetek számára nem feltétlenül jelent gondot, de nagy vállalatok számára elképzelhetetlen lenne, hogy az alkalmazottaik adatait és az azok szaktudásáról szóló információkat egy harmadik fél számára kiadják. A SkillMaster ezt az esetet figyelembe véve nem csak SaaS (Software as a Service) megoldásként elérhető, hanem lehetőséget biztosít dedikált módon, saját infrastruktúrára történő telepítésre is.

A dolgozat öt fejezetből áll. Az első fejezet röviden bemutatja a fejlesztés során alkalmazott módszereket és segédeszközöket.

A második fejezet a rendszer követelményeit, használati eseteit és architektúráját mutatja be.

[1] http://www.skills-base.com/

A harmadik fejezet a SkillMaster rendszer alapjául szolgáló adatmodellt, szerver oldali architektúrát, illetve a háttérszolgáltatásokat (backend) ismerteti. A felhasznált módszerek, eszközök és technológiák bemutatása mellett megemlít más Java technológiákat és keretrendszereket is. Ezek a rövid összehasonlítások az alternatív megoldások feltérképezését szolgálják. Mivel az esetek túlnyomó részében több eszköz is rendelkezésre áll egy adott feladat vagy követelmény megvalósítására, a kiválasztott eszköz vagy keretrendszer bemutatása mellett a dolgozat röviden kitér arra is, hogy miért esett az adott eszközre a választás.

A negyedik fejezet a webes felület felépítését, különböző elemeit és a megvalósításhoz felhasznált technológiákat tárgyalja (frontend), az első fejezethez hasonlóan kitérve az alternatív megoldásokra is. Mobil és egyéb kliens-oldali technológiák nem képezik részét a dolgozat tematikájának.

Az ötödik fejezet a rendszer használatát mutatja be fiktív, de valós alapú adatok felhasználásával. A fejezet részletesen bemutatja a rendszer nyújtotta funkciókat, ezért a fejezetben foglaltak felhasználói dokumentációnak is tekinthetőek.

A SkillMaster projekt megvalósításához szükséges, jelen dolgozatban bemutatott eszközökkel és technológiákkal a szerző a Codespring Kft. Mentorprogramjának keretein belül ismerkedett meg. Továbbá lehetősége volt a Mentorprogram során elsajátított elméleti tudást az említett cégnél töltött szakmai gyakorlat során gyakorlatba ültetni. A gyakorlat a cég alkalmazottainak szakmai irányítása alatt zajlott, a Codespring Kft. által biztosított infrastruktúra igénybevételével. A SkillMaster rendszer prototípusának létrejöttéért köszönet illeti a dolgozat szakmai irányítója mellett a Codespring Kft-t is.

1. Alkalmazott módszerek és eszközök

Vállalati (Enterprise) rendszernek minősülnek azok a szoftver termékek, illetve szolgáltatások, amelyek egy bizonyos vállalat vagy szervezet konkrét igényeinek kielégítését célozzák. A fogalmat tipikusan nagyobb méretű, összetett, több részből álló, sok funkcionalitást biztosító rendszerek esetében használjuk.

A vállalati rendszerek piacának növekedése megköveteli olyan módszerek és eszközök alkalmazását, amelyek elősegítik, átláthatóvá és megoszthatóvá teszik a fejlesztői munkát, továbbá folyamatosan biztosítják a szoftvertermék ellenőrizhetőségét és karbantarthatóságát.

A SkillMaster projekt kivitelezése során alkalmazott fejlesztési módszertan a Scrum [1], amely az Agilis módszerek közé tartozik. Ellentétben a klasszikus vízesés vagy V modellekkel, a Scrum egyesíti az iteratív és inkrementáló fejlesztési modellek előnyeit, ezért kiváltképp alkalmas a kutatás-fejlesztés típusú projektek kivitelezésére. A feladatok követése a SkillMaster projekt fejlesztése során a Trello [2] webes projektmenedzsment eszköz segítségével történt. A kapcsolódó dokumentumok megosztását egy XWiki [3] rendszer biztosította.

A vállalati rendszerek fejlesztésének egy másik alapkövetelménye a forráskód megfelelő módon történő tárolása, változásainak nyomon követése. A SkillMaster kódállománya Mercurial [4] verziókövető rendszer alatt van nyomon követve, amely egy ingyenes, nyílt forráskódú osztott verziókövető rendszer. A központi tároló menedzsmentje RhodeCode [5] segítségével történt, Mercurial asztali kliensalkalmazásként a TortoiseHg [6] szolgált.

Build és függőségmenedzsment rendszerek közül a SkillMaster projekt esetében alkalmazott rendszer a Maven [7], amely XML alapú, hierarchiába rendezhető POM (Project Object Model) konfigurációs állományokon alapul. Jelen dolgozat összeállításakor a Maven rendelkezik a legjelentősebb felhasználóbázissal, stabil, továbbá jelentős támogatottsággal bír az integrált fejlesztői környezetek (IDE) részéről. A projekt külső függőségeinek menedzsmentjében egy Artifactory [8] repository-management eszköz segített.

A fejlesztés a folytonos integráció (Continuous Integration) elvét követte [9], ezt a Jenkins [10] rendszer támogatta. Statikus kódelemző eszközként a SonarQube [11] szolgált. A fejlesztés Eclipse [12] fejlesztői környezetben történt.

2. A SkillMaster projekt

A SkillMaster rendszerben az adatokhoz és műveletekhez történő hozzáférés korlátozása szerepkörök segítségével van megvalósítva. A rendszerben alkalmazott szerepkörök:

- Adminisztrátor
- Menedzser
- Projektmenedzser
- Felhasználó

Minden szerepkörhöz egyedi műveletek és külön felület tartozik. A szerepköröknek megfelelő jogosultságok ellenőrzése kliens és szerver oldalon egyaránt megtörténik.

2.1. Követelmények

Az alábbiakban a SkillMaster rendszer fontosabb funkcionális követelményei lesznek felsorolva, alrendszerek szerint kategorizálva.

2.1.1. SkillMaster Szerver

A SkillMaster szerver biztosítja a rendszer különböző moduljai számára a működéshez szükséges környezetet. Megteremti az adathozzáférési réteg és az adatbázis szerver közti kapcsolatot, gondoskodik a perzisztencia műveletek megvalósításáról és a kliensalkalmazások kiszolgálásáról a szolgáltatási rétegen keresztül. Biztosítja a felhasználók hitelesítését és jogosultságaik ellenőrzését.

2.1.2. Adminisztrációs felület

Az adminisztrációs felület biztosítja az adminisztrátor hatókörhöz tartozó funkciókat és a szükséges műveletek elvégzésére szolgáló grafikus felületet.

Az adminisztrációs felület csak bejelentkezés után, a megfelelő szerepkörbe tartozó felhasználók számára érhető el. Biztosítja a felhasználók és a felhasználókhoz rendelt szerepkörök menedzsmentjét, a szakismeret kategóriák és alkategóriák meghatározását, szakismeretek bevezetését és kategóriákhoz rendelését.

2.1.3. Menedzsment felület

A menedzsment felületen keresztül érhetőek el az intézmény-szintű kimutatások. A felület lehetőséget biztosít a felhasználók listázására, szűrésére kategóriák és szakismeretek alapján. Bejelentkezést követően, menedzser hatókörből érhető csak el.

2.1.4. Projektmenedzsment felület

A projektmenedzsment felület betekintést enged a felhasználók önértékeléseibe. Lehetőséget biztosít a projektmenedzser szerepkörben lévő felhasználók számára a

felhasználók profiljainak böngészésére, továbbá azok szakismeretének értékelésére. Eléréséhez projektmenedzser szerepkör és bejelentkezés szükséges.

2.1.5. Felhasználói felület

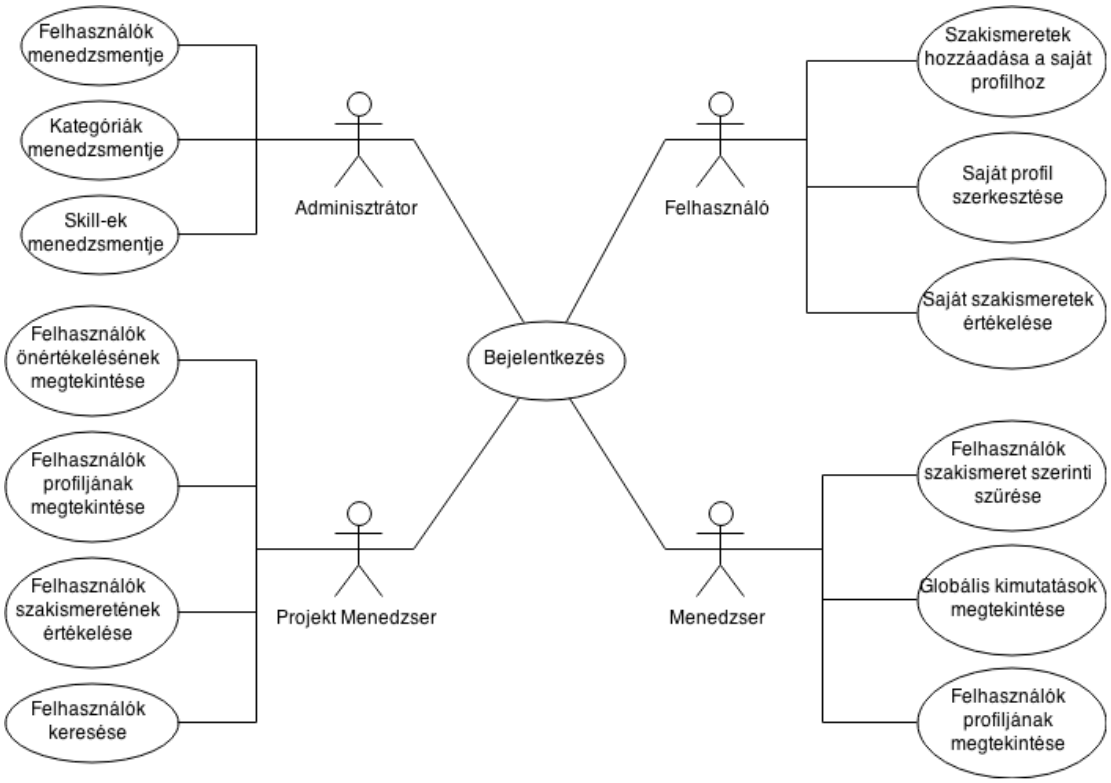
A felhasználói felület lehetőséget biztosít a rendszerben tárolt személyes adatok szerkesztésére, továbbá a rendszerben szereplő kategóriák és szakismeretek listázására és a szakterületek felhasználói profilhoz való hozzárendelésére.

A profilhoz rendelt szakismeretekhez önértékelés adható meg, amely kiegészíthető az adott téma iránti érdeklődési szint, illetve a továbbtanulási igény megadásával. A felhasználói felület biztosítja az adott szakterülethez tartozó önértékelés módosítását, valamint a projektmenedzserek értékeléseinek megtekintését.

A felhasználói felület felhasználó szerepkörből, bejelentkezést követően érhető el.

2.2. Használati esetek

Az 1. ábra a SkillMaster rendszer használati eseteit szemlélteti, szerepkörök szerint csoportosítva.



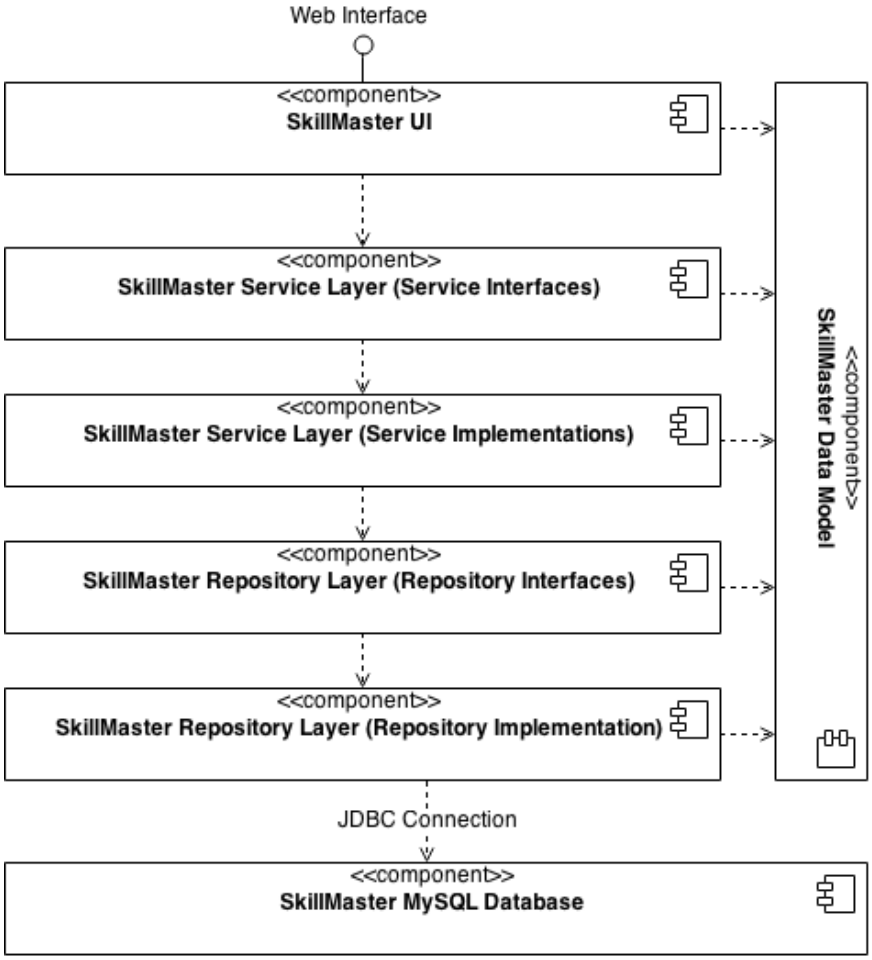
1. ábra A SkillMaster használati esetei

2.3. Architektúra

A SkillMaster rendszer architektúrája a többrétegű architektúra-modellt (Multi-tier architecture) követi. Minden köztes réteg interfészeken keresztül teszi elérhetővé a szolgáltatásait.

Minden réteg a közös adatmodellt (központi entitások) használja. Az adathozzáférési (Repository) rétegen belül vannak implementálva az entitásokkal kapcsolatos perzisztencia műveletek. A konkrét alkalmazáslogikával kapcsolatos műveletek a szolgáltatási (Service) rétegen belül vannak megvalósítva. Ezeket a műveleteket veszik igénybe a szolgáltatási réteg interfészein keresztül a kliensek kéréseit fogadó és kiszolgáló webes komponensek.

Az architektúrát a 2. ábra szemlélteti.



2. ábra A SkillMaster komponens diagramja

3. A szerver megvalósítása és a felhasznált szerver-oldali technológiák

A Java platform bő választékkal rendelkezik szerver oldali keretrendszerek és technológiák terén. Vállalati rendszerek esetében a projekt sikeres kivitelezése nagyban függ a felhasznált keretrendszerektől, amelyek nagymértékben egyszerűsítik és rövidítik a fejlesztés folyamatát.

3.1. Adatmodell

A SkillMaster alkalmazás adatmodelljének alapját képező entitások a „convention over configuration” elvet szem előtt tartva Java Bean-ként vannak reprezentálva. A Java Bean-ek olyan általános Java osztályok (Plain Old Java Object, POJO), amelyek privát adattagokkal, az adattagokhoz tartozó publikus hozzáférő, illetve beállító metódusokkal (getter/setter), valamint publikus, paraméter nélküli konstruktorral rendelkeznek és szerializálhatóak.

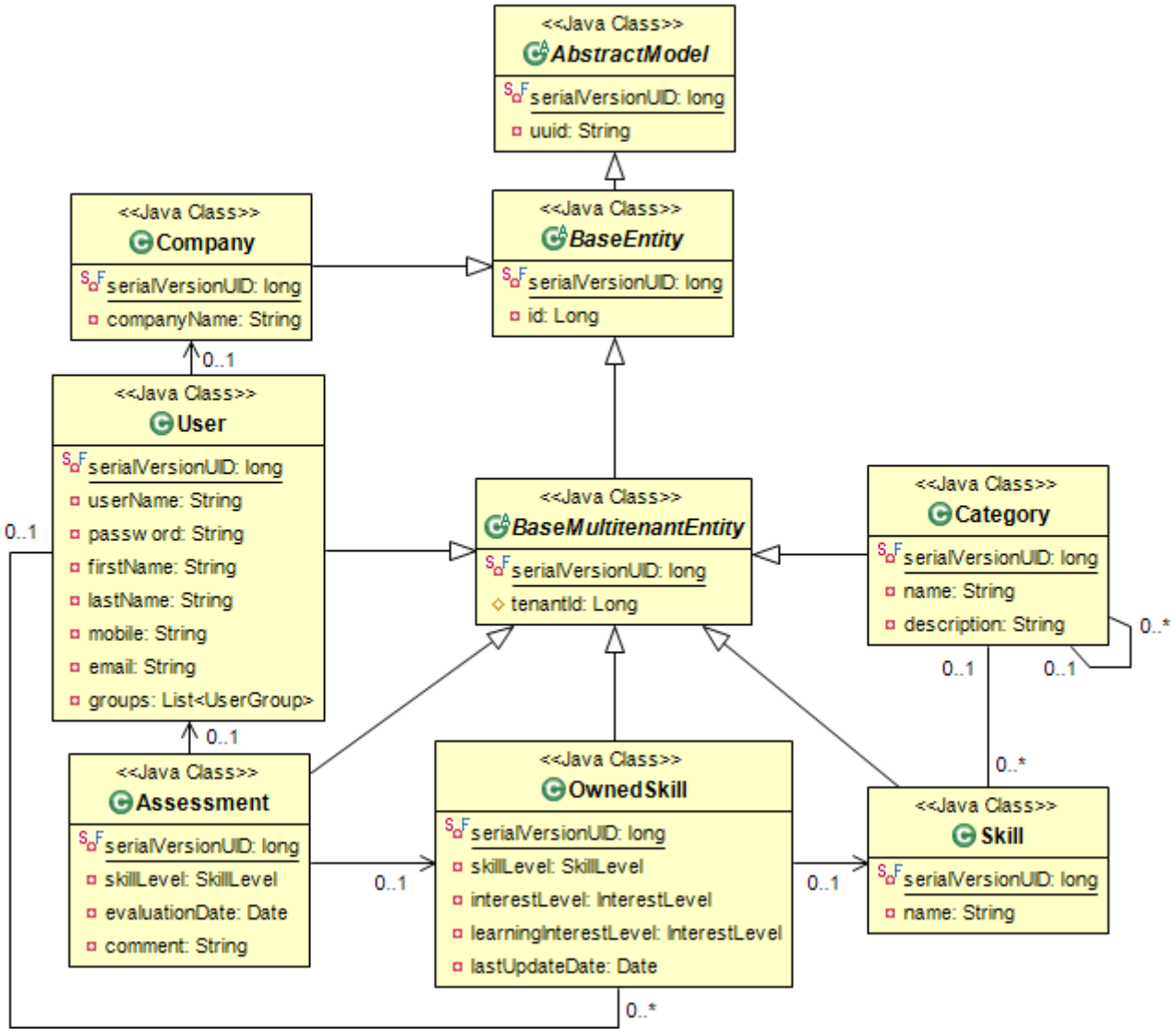
Mivel az adathozzáférési réteg a Java Persistence API (JPA) [13] specifikációt implementáló ORM (Object Relational Mapping) keretrendszert használ, az entitások JPA meta adatokat is tartalmaznak, amelyek annotációk segítségével vannak megadva. A SkillMaster adatmodelljét képező osztályok tehát JPA entitások, a perzisztenciával kapcsolatos műveletek ennek megfelelően egy JPA EntityManager szolgáltatáson keresztül valósulnak meg.

Az *edu.codespring.skillmaster.backend.model* csomag tartalmazza a SkillMaster rendszer adatmodelljének elemeit. Az adatmodell három alapvető elemre, ősosztályra épül.

Az *edu.codespring.skillmaster.backend.model.AbstractModel* absztrakt osztály áll a modellhierarchia csúcsán, a rendszerben szereplő összes entitás ősosztálya. Implementálja a *java.io.Serializable* interfészt, és emellett ellátja az entitásokat egy olyan univerzálisan egyedi azonosítóval (Universally Unique Identifier, UUID), amely lehetőséget biztosít azok megkülönböztetésére, egyedi azonosítására, már azelőtt is, mielőtt az entitások adatbázisba történő mentése megtörténik és az adatbázis-menedzsment rendszertől megkapják az elsődleges kulcsnak megfelelő egyedi azonosítójukat.

Az *edu.codespring.skillmaster.backend.model.BaseEntity* absztrakt osztály a hierarchia következő eleme, amely a relációs adatbázis által meghatározott numerikus azonosító nyilvántartására biztosít lehetőséget az entitások számára. A legtöbb entitás egy

Long típusú azonosítóval rendelkezik, amelyet ettől a *BaseEntity* osztálytól örököl. Ez az azonosító az adatbázis oldalon generált elsődleges kulcsnak a megfelelője, az entitások első mentésekor lesz inicializálva. A modell a kivételes helyzetek kezelésére is lehetőséget ad: amennyiben valamelyik entitás esetében indokolt más típusú (például összetett) kulcs alkalmazása, az illető entitás lehet az *AbstractModel* közvetlen leszármazottja.



3. ábra a SkillMaster adatmodelljének vázlatos diagramja

Az *edu.codespring.skillmaster.backend.model.BaseMultitenantEntity* absztrakt osztály a hierarchia soron következő eleme, amely lehetőséget biztosít a perzisztens entitások csoportosítására egy numerikus érték (tenantId) meghatározásával. Ennek az adatnak a bevezetése a multi-tenancy szempontjából fontos. SaaS rendszer lévén a SkillMaster alkalmazás esetében fontos, hogy minden felhasználó csak a saját cégéhez tartozó adatokhoz férjen hozzá. Következményként gyakorlatilag minden adatbázis lekérdezésnek ki kell egészülnie egy plusz feltétellel. A modern ORM keretrendszerek, mint például a SkillMaster

esetében használt EclipseLink [14] is, támogatják az ilyen helyzetek megoldását: a tenant azonosító alapján (amelyet a SkillMaster esetében a felhasználó intézménye határoz meg) automatikusan kiegészítik a lekérdezéseket a megfelelő feltétellel. A SkillMaster esetében a legtöbb entitás ennek a *BaseMultitenantEntity* osztálynak a leszármazottja. Azok az entitások, amelyek nem tenant-függőek (nem csak adott céghez tartoznak) lehetnek a *BaseEntity* közvetlen leszármazottjai.

Az *AbstractModel*, *BaseEntity* illetve *BaseMultitenantEntity* absztrakt osztályok el vannak látva a *javax.persistence.MappedSuperclass* annotációjával. Az annotáció jelzi a JPA implementációnak (jelen esetben EclipseLink), hogy az adott osztálynak nem felel meg külön adatbázis tábla, de az osztály tartalmaz olyan adatokat, amelyek a származtatott osztályok példányainak esetében részét fogják képezni azok perzisztens állapotának, tehát az EntityManager szolgáltatásnak menedzselnie kell ezeket az adatokat.

Az *edu.codespring.skillmaster.backend.model.User* osztály reprezentálja a rendszerben jelenlévő felhasználókat, tartalmazza a bejelentkezéshez szükséges adatokat, illetve személyes információkat. Az *edu.codespring.skillmaster.backend.model.UserGroup* enumeráció elemeinek segítségével történik a különböző szerepkörök felhasználókhoz rendelése, melyek különböző hozzáférési jogosultságokat biztosítanak.

Az *edu.codespring.skillmaster.backend.model.Category* osztály szerepe a különböző szakterületek kategóriákba szervezése, és a kategóriák közti hierarchia megvalósítása.

Az *edu.codespring.skillmaster.backend.model.Skill* osztály reprezentálja a rendszerbe bevezetett különböző szakterületeket és azok általános jellemzőit, leírását.

Az *edu.codespring.skillmaster.backend.model.OwnedSkill* osztály valósítja meg a különböző szakterületek felhasználókhoz való rendelését, valamint lehetőséget biztosít a felhasználók önértékelésének, érdeklődési szintjének és továbbtanulási igényének nyilvántartására, az *edu.codespring.skillmaster.backend.model.SkillLevel*, illetve az *edu.codespring.skillmaster.backend.model.InterestLevel* enumerációk elemeinek segítségével.

Az *edu.codespring.skillmaster.backend.model.Assessment* osztály reprezentálja a menedzser pozícióban lévő felhasználók által meghatározott értékeléseket, amelyekkel kiegészíthető a felhasználók önértékelése.

Az *edu.codespring.skillmaster.backend.model.Company* osztály reprezentálja a cégeket és szervezeteket a hozzájuk tartozó információkkal. A *Company* entitásokhoz tartozó

azonosítót alkalmazza a rendszer tenant azonosítóként az adatok csoportosításához és a hozzáférés korlátozásához.

A konkrét adatmodelleket leíró osztályok a *javax.persistence.Entity* annotációval vannak ellátva. A *BaseEntity* osztálytól örökölt azonosítót a *javax.persistence.Id* annotáció jelöli. Az ORM leképezés meghatározásához használva vannak továbbá a *javax.persistence.Table és a javax.persistence.Column* annotációk is, utóbbi egyben az adatbázis szintű megkötéseket is meghatározza.

Annak érdekében, hogy az adatok már a felsőbb szinteken belül is validálhatóak legyenek a felhasznált keretrendszerek által, az entitások bizonyos attribútumai a BeanValidation [15] specifikáció által meghatározott annotációkkal is el vannak látva.

3.2. Adathozzáférési réteg

A SkillMaster adathozzáférési rétegének alapját az Enterprise JavaBean (EJB) technológia [16][17], valamint a Java Persistence API (JPA) képezi.

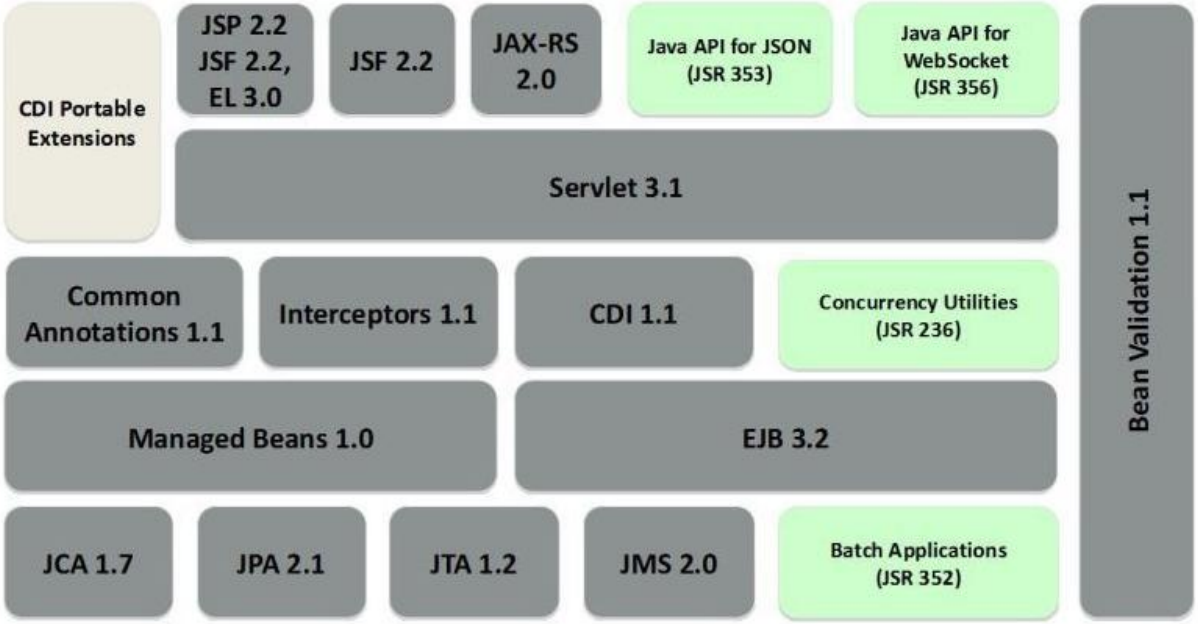
3.2.1. Enterprise JavaBean-ek

Az EJB egy szerver oldali komponens-modell, amely támogatja a moduláris felépítést, illetve az osztott architektúrát. Az EJB szabvány célja egy általános módszertan meghatározása az adatmentés, tranzakció-menedzsment, illetve adatbiztonság megvalósítására. Az EJB komponensek felelőssége az üzleti logikával kapcsolatos műveletek megvalósítása. A specifikáció két fő komponenstípust határoz meg, a Session bean-eket, amelyek működése közvetlen metódushívásokon alapszik, illetve az aszinkron üzeneteken alapuló kommunikációt támogató Message-driven bean-eket, amelyek működése tipikusan JMS (Java Message Service) [18] alapú üzeneteken alapszik. Ezt a két kategóriát egészítik ki a JPA specifikáció részét képező JPA entitások, amelyek az adatmodellt reprezentálják (a korábbi EJB specifikáció részét képező EntityBean-eket helyettesítik).

Az EJB technológia alkalmazásának előfeltétele egy EJB konténert biztosító Java EE alkalmazásszerver. A SkillMaster rendszer esetében ez a GlassFish [19], amely a Java Enterprise Edition szabványcsalád referencia implementációja.

A SkillMaster esetében az üzleti logikával kapcsolatos műveletek Session Bean-ek által vannak megvalósítva. Ezek két kategóriába sorolhatóak, lehetnek állapottal rendelkezőek

(stateful), vagy állapot nélküliek (stateless). A SkillMaster komponensei az utóbbi kategóriába tartoznak. A Session Bean-ek kommunikálhatnak lokális (az egy konténeren belül menedzselt bean-ek közötti kommunikáció esetében), remote (távoli metódushívások esetében), vagy endpoint (SOAP alapú klasszikus web szolgáltatások esetében) interfészeken keresztül. A SkillMaster komponensei lokális interfészeken keresztül szolgáltatják funkcionalitásaikat.



4. ábra GlassFish 4.0 komponensdiagram (forrás: http://www.h-online.com/)

3.2.2. Java Persistence API

A JPA szabvány célja standardizált eljárások meghatározása az adatok adatbázisba történő mentésére, valamint a tárolt adatok állapotának lekérdezésére, módosítására. Lehetőséget biztosít relációs adatbázisokkal történő kommunikációra egy sajátos, funkcionalitásaiban a standard SQL lekérdezőnyelvhez hasonló nyelv, a JPQL (Java Persistence Query Language) [20] biztosításával. A JPQL előnye, hogy a lekérdezések megfogalmazása során Java objektumok alkalmazhatóak, így nem tér el az objektumorientált szemléletmódtól, ebből adódóan nem szükséges különválasztani a Java kódtól.

A JPA szabványnak több ingyenesen használható, nyílt forráskódú implementációja létezik, például az OpenJPA, Hibernate és EclipseLink. A különböző implementációk a JPA specifikáció megvalósítása mellett egyéb, esetenként egyedi funkciókat is biztosítanak.

A SkillMaster rendszer esetében alkalmazott JPA implementáció az EclipseLink, amely a JPA specifikáció referencia implementációja. A referencia implementáció

alkalmazásának előnye, hogy a standard bővítése esetén az új funkcionalitásokat elsőként biztosítja. Továbbá az EclipseLink biztosítja a SkillMaster esetében szükséges multi-tenancy támogatást, amely még nem része a JPA szabványnak és nem rendelkezik teljes támogatással egyéb JPA implementációk esetében.

3.2.3. Repository réteg

A SkillMaster rendszer adathozzáférési rétege két részből áll. Az *edu.codespring.skillmaster.backend.repository* csomag interfészeket tartalmaz, amelyek meghatározzák az egyes adatmodelleken elvégezhető műveleteket, az *edu.codespring.skillmaster.backend.repository.bean* csomag tartalmazza az interfészek implementációit, meghatározva az adatok manipulációjának konkrét módját.

Az adathozzáférés módját meghatározó interfészek esetében létezik egy közös ős, az *edu.codespring.skillmaster.backend.repository.BaseRepository* interfész, amely deklarálja az általános adathozzáférési műveleteket (create, read, update, delete; CRUD), valamint generikus típusparaméterek segítségével meghatározza az adatmodell, illetve annak egyedi azonosítójának típusát. A generikus típusparamétereknek köszönhetően a *BaseRepository* interfészt kiterjesztő konkrét repository interfészeknek elégséges az adott adatmodellhez tartozó sajátos adathozzáférési metódusokat deklarálni. Az interfészek esetében alkalmazott elnevezési konvenció az adatmodell neve kiegészítve a Repository utótaggal.

A *BaseRepository*-t kiterjesztő interfészek a *javax.ejb.Local* annotációval vannak ellátva, amely azt jelzi, hogy az interfész egy EJB lokálisan (JVM-en belül) elérhető üzleti logikáért felelős metódusait határozza meg.

Az adathozzáférést megvalósító EJB hierarchia csúcsán az *edu.codespring.skillmaster.backend.repository.bean.BaseRepositoryBean* absztrakt osztály áll. A *BaseRepositoryBean* megvalósítja a *BaseRepository* interfészt és JPA CriteriaQuery típusú műveletek alkalmazásával általános implementációkat ad a CRUD műveletekre, amelyek minden, a *BaseRepositoryBean*-t kiterjesztő EJB által öröklődnek.

A hierarchia soron következő, szintén absztrakt eleme az *edu.codespring.skillmaster.backend.repository.bean.BaseMultitenantRepositoryBean*, azok az EJB-k terjesztik ki, amelyek tenant-függőek. A *BaseMultitenantRepositoryBean* a repository csomag *BaseMultitenantRepository* interfészét valósítja meg lehetőséget biztosítva egy

repository kérés esetén a tenant azonosító beállítására. Azok az EJB-k, amelyek nem tenant-függőek a *BaseRepositoryBean*-t direkt módon terjesztik ki.

Az üzleti logikát megvalósító EJB-k a *javax.ejb.Stateless* annotációval vannak ellátva, így az EJB konténer két repository kérés között nem őriz meg állapotinformációkat. Az EJB-k esetében alkalmazott elnevezési konvenció az EJB-hez tartozó repository interfész nevéből áll kiegészítve a Bean utótaggal.

3.2.4. Multi-tenancy

A SkillMaster rendszer egyik alaptulajdonsága, hogy képes SaaS rendszerként működni. Egy központi szerveren keresztül biztosít hozzáférést a szolgáltatásokhoz több cég számára, és mindegyik cég csak a saját adatait láthatja és kezelheti. A közös adatbázis használata szükségessé teszi az adatok csoportosítását, valamint a hozzáférés korlátozását. Ennek érdekében a SkillMaster multi-tenancy mechanizmust használ.

A multi-tenancy alapgondolata egy központi szerveren futó alkalmazás, amely több felhasználócsoportnak (tenant) biztosít hozzáférést a szolgáltatásaihoz, adataikat elszigetelten kezelve. A SkillMaster rendszer esetében a különböző tenant-okat a *Company* entitások képviselik, melyek egyedi azonosítója szolgál az adatok csoportosításához szükséges kulcsként.

Az elszigetelést igénylő entitások közös őse a *BaseMultitenantEntity* absztrakt osztály, amely az *org.eclipse.persistence.annotations.Multitenant* annotációval van ellátva. Az adatbázis oldalon alkalmazott stratégiát az adatok elszigetelésére az *org.eclipse.persistence.annotations.MultitenantType* enumeráció elemei határozzák meg. Az alkalmazott stratégiák lehetnek: közös tábla, külön tenant-onkénti tábla, illetve virtuális privát adatbázis (Virtual Private Database, VPD) alkalmazása. A VPD stratégia alkalmazása feltételezi, hogy a háttérben használt adatbázis-kezelő rendszer támogatja a virtuális adatbázisok kezelését. A tárolandó adatmennyiség, figyelembe véve az adatbázis-kezelő rendszer képességeit nem indokolja a több tábla alkalmazását, ezért a SkillMaster a közös tábla stratégiát alkalmazza.

Az *org.eclipse.persistence.annotations.TenantDiscriminatorColumn* annotáció alkalmazásával a *BaseMultitenantEntity* meghatározza minden tenant-függő entitásnak megfelelő adatbázis táblában a tenant id-t tartalmazó oszlop nevét és típusát. A tenant id

adatbázis oldali típusa a *javax.persistence.DiscriminatorType* enumeráció valamelyik elemével állítható be.

A *BaseMultitenantEntity*-t kiterjesztő entitások számára nem szükséges a multi-tenancy-t érintő további konfiguráció, az EclipseLink biztosítja az objektumorientált hierarchiának megfelelő adatbázis-oldali reprezentációt. Ennek köszönhetően az entitások átláthatóbbak maradnak, karbantartásuk egyszerűsödik.

A tenant-függő entitások esetében minden adatbázis művelet megköveteli a tenant id jelenlétét. Ezt a SkillMaster rendszer az Interceptor [21] tervezési minta alkalmazásával valósítja meg, melynek előnye, hogy a rendszerben kommunikációt folytató komponensek számára teljesen transzparens marad.

Java EE környezetben az Interceptor tervezési minta megvalósítása a *javax.interceptor.Interceptors* annotáció segítségével történik. Az annotációt a megszakítani kívánt metódusra helyezve, a megszakítást kezelő egy vagy több interceptor típusát megadva célszerű megvalósítani. A megszakítást követően a kijelölt interceptor *javax.interceptor.AroundInvoke* annotációval jelzett metódusa kerül végrehajtásra.

Az *edu.codespring.skillmaster.backend.interceptor.MultitenantInterceptor* interceptor osztály biztosítja a tenant azonosítók beállítását. A folyamat úgy működik, hogy a tenant-függő repository hívásokat megszakítja, az interceptált EJB-hez hozzárendelt SessionContext objektumból kinyeri a bejelentkezett felhasználó adatait, amelyek alapján meghatározza az adott felhasználóhoz tartozó tenant azonosítót. Az így kapott azonosítót konfigurációs paraméterként átadja a megszakított kérést végző EJB-hez hozzárendelt EntityManager szolgáltatásnak, majd tovább engedi a kérést. Az EclipseLink keretrendszer által biztosított EntityManager ezt követően gondoskodik a tenant azonosító adatbázis oldalra történő továbbításáról, a lekérdezések megfelelő kiegészítéséről.

3.2.5. Kivételkezelés és naplózás

A SkillMaster rendszer a repository réteg esetében definiál egy saját kivételtípust, az *edu.codespring.skillmaster.backend.repository.exception.RepositoryException* osztályt.

A *RepositoryException* a *java.lang.RuntimeException* osztályt terjeszti ki, ezáltal a nem ellenőrzött kivételek családjába tartozik. A repository réteg esetében fellépő kivételeket a rendszer az SLF4J [22] segítségével létrehozott absztrakciós szinten keresztül a LOG4J [23]

keretrendszer segítségével naplózza, majd a fellépő kivételt egy *RepositoryException*-hoz láncolva továbbítja a felsőbb rétegek fele.

Mivel a repository réteg esetén fellépő kivételek a konkrét típusuknak megfelelően kell legyenek továbbítva és ezen kívül lehetnek megszakadt vagy hibás tranzakciók eredményei, a *RepositoryException* el van látva a *javax.ejb.ApplicationException* annotációval. Az *ApplicationException* annotáció konfigurálható olyan módon, hogy az annotált kivétel fellépése esetén az EJB konténer visszaállítja a tranzakció megkezdése előtti állapotot, így garantálva az adatok integritását.

3.3. Szolgáltatás réteg

A SkillMaster szolgáltatási rétegének szerepe az adathozzáférési réteg szolgáltatásainak publikációja a kliensalkalmazások fele, illetve a más, összetettebb üzleti logikával kapcsolatos műveletek megvalósítása. A szolgáltatási réteg architektúrája nagyban hasonlít az adathozzáférési réteg architektúrájához.

Az *edu.codespring.skillmaster.backend.service* csomag tartalmazza a különböző entitásokhoz kapcsolódó szolgáltatásokat meghatározó interfészeket, amelyek egy közös őstől, az *edu.codespring.skillmaster.backend.service.BaseService* interfésztől öröklik az általános szolgáltatásokat deklaráló metódusokat. A *BaseService* interfész a *BaseRepository*-hoz hasonlóan generikus paraméterekkel biztosítja az általános műveleteket a konkrét szerviz interfészek számára. Továbbá minden szerviz interfész el van látva a *javax.ejb.Local* annotációval, így a keretrendszer biztosítja, hogy a szolgáltatások csak a szolgáltatás réteggel egy virtuális gépben futó kliensalkalmazások számára legyenek elérhetőek.

Az *edu.codespring.skillmaster.backend.service.bean* csomag tartalmazza a szerviz interfészek implementációját, azaz a SkillMaster alkalmazás szerver oldali üzleti logikájáért felelős komponenseket. A repository réteggel ellentétben a szolgáltatási réteg nem biztosít a *BaseService* interfészt megvalósító általános implementációt a közös műveletek megvalósítására, mivel az üzleti logika ezekben az esetekben eltérő lehet. A szerviz komponensek állapot nélküli működését a *javax.ejb.Stateless* annotáció megadásával teszi lehetővé az EJB konténer.

A repository réteghez hasonlóan a szolgáltatási réteg is saját kivételtípust határoz meg, amelyet az *edu.codespring.skillmaster.backend.service.exception.ServiceException* osztály implementál. A *ServiceException* tulajdonságait tekintve megegyezik a *RepositoryException*

kivételtípus tulajdonságaival. A kivételkezelés és naplózás a repository réteg esetén ismertetett stratégia alapján működik a szolgáltatási réteg esetében is. Az alkalmazott elnevezési konvenció is megegyezik a repository réteg esetében alkalmazott konvencióval, a Repository utótagot Service utótagra cserélve.

3.3.1. Inversion of Control

A SkillMaster rendszer az IoC (Inversion of Control) és a DI (Dependency Injection) [24] tervezési minták alkalmazásával teszi lehetővé a szerver oldali komponensek életciklusának menedzsmentjét.

A DI minta alkalmazásának célja a komponensek életciklus menedzsmentjének teljes átruházása az IoC konténerre. A DI jelentősen egyszerűsíti a komponensek közti kapcsolatok karbantartását, mivel a kapcsolathoz szükséges referencia a *javax.ejb.EJB* annotáció alkalmazásával egyszerűen beinjektálható bármelyik menedzselt állapotban lévő komponensbe.

A SkillMaster által alkalmazott IoC konténer másik előnye, hogy interfész típusú referencia injektálása esetén automatikusan beazonosítja az interfészhez tartozó implementációt, létrehozza az adott osztály példányát és az annak megfelelő konkrét típust injektálja be. Ennek köszönhetően a repository és szolgáltatás rétegek implementációi könnyen, más modulok számára transzparens módon lecserélhetőek.

3.3.2. Biztonság

A SkillMaster rendszer szerver oldali biztonságkritikus pontjai az adatok biztonságos tárolása, illetve az adathozzáférés korlátozása.

Az adatbázis kapcsolat létesítéséhez a SkillMaster rendszer a GlassFish alkalmazásszerver által biztosított DataSource objektumot alkalmaz. A DataSource objektum tartalmazza az adatbázis kapcsolat paramétereit, beleértve a felhasználó azonosításához szükséges adatokat.

Az adathozzáférés korlátozására alkalmazott stratégia szerepkörökön alapszik. Minden felhasználó egy vagy több szerepkörbe tartozik és kizárólag a szerepköreinek megfelelő adatokhoz van hozzáférése.

A SkillMaster az adathozzáférési réteg szolgáltatásait a service rétegen keresztül teszi elérhetővé. A biztonsági mechanizmus a JAAS (Java Authentication and Authorization Service) specifikáción alapszik. A szolgáltatási réteg műveleteit megvalósító összes metódus el van látva a *javax.annotation.security.RolesAllowed* annotációval, amelynek segítségével az adott funkció használata korlátozható az annotációban megadott szerepkörök segítségével.

A rendszerben tárolt felhasználók jelszavai titkosítva vannak tárolva, hashelt formában. A titkosítás szerver oldalon történik.

A felhasználók hitelesítését a GlassFish biztosítja egy JDBC Security Realm-en keresztül. A JDBC Security Realm a SkillMaster által használt adatbázisból nyeri ki a tárolt felhasználók adatait, beleértve az egyes felhasználók szerepköreit. A Security Realm alkalmazásának előnye, hogy könnyen és gyorsan konfigurálható, illetve az alkalmazásszerverre kitelepített alkalmazás esetében a hitelesítésre vonatkozó műveleteket automatikusan megvalósítja.

4. Webes technológiák

A Java webes keretrendszerek két csoportba sorolhatóak. Az első csoportot a kliens-központú keretrendszerek alkotják, mint például a GWT (Google Web Toolkit) [25]. Ezeknek a keretrendszereknek az esetében a web alkalmazás és a szerver közti kommunikáció minimális, a kliensalkalmazás a kliens eszközön fut.

A második csoportot a szerver-központú megoldások alkotják, mint a JSF (Java ServerFaces) [26], vagy a SkillMaster esetében alkalmazott Vaadin [27] [28] keretrendszer. A szerver-központú keretrendszerek esetében a grafikus komponensek kliens oldalon jönnek létre, de szerver oldalon vannak implementálva és a kliens és szerver oldali példányok állapota folyamatosan szinkronban van. A vezérlést megvalósító komponensek szintén szerver oldalon találhatóak.

A SkillMaster esetében használt Vaadin keretrendszer egy nyílt forráskódú, Java alapú webes keretrendszer, amely modern, összetett felhasználói felülettel rendelkező és sok funkcionalitást biztosító webes alkalmazások fejlesztését támogatja. A felhasználói felületek Java kód által építhetőek fel, a szerver oldali Java komponensek alapján a Vaadin GWT alkalmazásával automatikusan JavaScript kódot generál.

4.1. Architektúra

A SkillMaster projekthez tartozó web alkalmazás architektúrája az MVC (Model-View-Controller) elvet betartva, jól meghatározott szereppel rendelkező komponensekből épül fel.

A felhasználói felület egyetlen weboldalból áll, amelyet a *com.vaadin.ui.UI* absztrakt osztály kiterjesztésével az *edu.codespring.skillmaster.web.SkillMasterUI* osztály valósít meg. Ez az osztály képviseli a web alkalmazás belépési pontját.

Az oldalon megjelenő különböző tartalmakat meghatározó nézetek (view) az *edu.codespring.skillmaster.web.view* csomagban találhatóak. Minden egyedi tartalmat biztosító nézet megvalósítja a *com.vaadin.navigator.View* interfészt és rendelkezik egy névvel. Az elnevezésnél alkalmazott konvenció a komponens által biztosított tartalom megnevezése ellátva a View utótaggal.

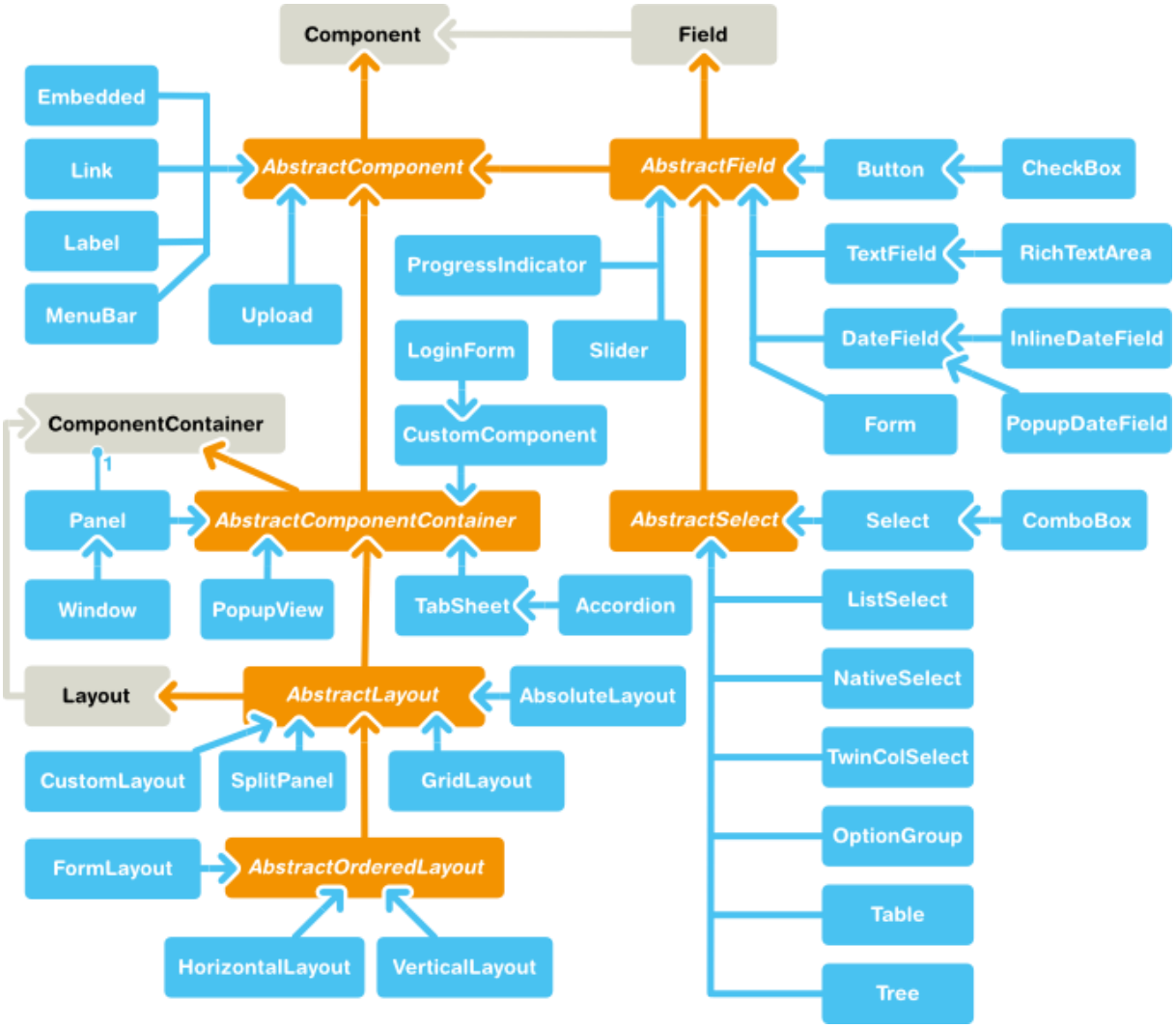
A különböző nézetek közti navigációt a *com.vaadin.navigator.Navigator* osztály valósítja meg. A navigátor alkalmazása előtt szükséges a különböző nézetek regisztrációja a navigátornál, ami a nézetek nevét és konkrét típusát megadva történik. Csak azok a nézetek kerülnek regisztrálásra, amelyek az aktuális bejelentkezett felhasználó számára megtekinthetőek.

A navigátor a web alkalmazáson belülről a UI.getCurrent().getNavigator() metódushíváson keresztül bárhonnan elérhető. A nézetek közti váltás a navigátor példány által, a kívánt nézet nevét megadva történik.

A SkillMaster webes felületének megvalósítása túlnyomó részt a standard, Vaadin keretrendszer által biztosított komponensek felhasználásával történik. Az egyedi komponensek az *edu.codespring.skillmaster.web.view.component* csomagban találhatóak.

A vezérlést megvalósító komponensek (controller) az *edu.codespring.skillmaster.web.controller* csomagban találhatóak. Minden nézet rendelkezik egy hozzá rendelt kontrollerrel, amely a backend oldali szolgáltatási rétegen keresztül kommunikál az adathozzáférési réteggel. A kontrollerek esetében alkalmazott elnevezési konvenció a nézet neve, amelyben a View utótag helyet a Controller utótag szerepel.

A kontrollerek az *edu.codespring.skillmaster.web.controller.BaseController* absztrakt osztályt terjesztik ki, amely általános, minden kontroller esetében érvényes műveleteket határoz meg.



5. ábra Vaadin komponensmodell (forrás: https://vaadin.com/)

4.1.1. Context and Dependency Injection

A SkillMaster projekt web modulján belül a Vaadin keretrendszerrel együtt használt technológia a CDI (Context and Dependency Injection) [29]. A CDI működése kontextusokon alapszik, célja pedig az IoC (Inversion of Control) és a Dependency Injection (DI) minta megvalósítása. A DI alkalmazásának előnye, hogy az alkalmazott IoC konténer által megvalósítható a komponensek életciklusának menedzsmentje, beleértve a példányosítást is és a konténer automatikusan kezeli a komponensek közötti függőségeket is.

A CDI konténer által menedzselt komponensek különböző kontextusokba, hatókörökbe (scope) tartoznak. A Vaadin komponensek életciklusának menedzsmentje a *com.vaadin.cdi.UIScoped* annotáció segítségével ruházható át a Vaadin-CDI rendszerre. Az *UIScoped* annotáció mellet léteznek speciális, az általános UI hatókört kiterjesztő hatókörök.

A *com.vaadin.cdi.CDIUI* annotáció jelenléte kötelező minden *com.vaadin.ui.UI* absztrakt osztályt kiterjesztő osztály számára, így a *SkillMasterUI* is el van látva ezzel az annotációval. A *CDIUI* annotáció lehetőséget biztosít egy URI (Unified Resource Identifier) részlet adott UI-hoz rendelésére. Ennek hiányában, alapértelmezetten a felhasználói felület a UI osztály egyszerű nevének megadásával érhető el.

A SkillMaster web alkalmazás különböző tartalmait biztosító nézetek a *com.vaadin.cdi.CDIView* annotációval vannak ellátva, amely a navigátor hatóköréhez rendeli ezeket. A *CDIView* annotáción keresztül állítható be a nézetek azonosítását szolgáló név.

A web modul egyéb menedzselt komponensei, így a kontrollerek és a SkillMaster által alkalmazott egyedi grafikus komponensek az általános *UIScoped* annotációval vannak ellátva.

A komponensek közti kapcsolatok megvalósítása a szükséges referenciák injektálásával történik. A *javax.inject.Inject* annotációval ellátott mezőket a CDI rendszer példányosítja és állítja be. A CDI alkalmazása esetén a szükséges referenciák még nincsenek beállítva az objektumok létrehozásakor, így az objektumok inicializációjakor az ezeket is érintő műveletek a konstruktor helyett speciális, *javax.annotation.PostConstruct* annotációval ellátott metódusokon belül vannak implementálva. Ezt a metódust minden létrehozott példány esetében meghívja a keretrendszer, garantálva azt, hogy a végrehajtás pillanatában minden szükséges referencia be van injektálva.

4.2. Biztonság

A Vaadin keretrendszer architektúrájából adódóan egy biztonságos modellt biztosít web alkalmazások fejlesztéséhez. A web alkalmazásokat érintő legismertebb és leggyakoribb sebezhetőségek ellen védelmet nyújt, levéve a fejlesztők válláról ennek terhét.

A Vaadin keretrendszer a felhasználók hitelesítésére (authentication) nem biztosít beépített megoldást, de több Java platformon elérhető keretrendszerrel képes együttműködni, például az Apache Shiro, a Spring Security vagy a JAAS (Java Authentication and Authorization Service) [30] specifikációt implementáló keretrendszerekkel is.

A SkillMaster esetében alkalmazott hitelesítés JAAS specifikáción alapszik, megvalósítását a GlassFish alkalmazásszerver biztosítja. Az alkalmazásszerverre telepített SkillMaster fenntart egy különleges URL (Uniform Resource Locator) tartományt, amelyhez az alkalmazásszerver csak a hitelesítést követően enged hozzáférést. Minden védett erőforrás a */main/** URL tartományban érhető el.

A felhasználók hitelesítését egyedi felhasználónév és jelszó alkalmazásával, űrlap (form) alapú bejelentkezést követően a GlassFish alkalmazásszerver végzi el. A GlassFish csak sikeres bejelentkezést követően irányít tovább a védett tartományban lévő felhasználói felülethez.

A hozzáférés jogosultságának ellenőrzését (authorization) a bejelentkezett felhasználó szerepköreinek figyelembe vételével a *Navigator* végzi. Minden egyedi nézetet reprezentáló osztály el van látva a *javax.annotation.security.RolesAllowed* annotációval. Az annotáció segítségével minden nézet esetében korlátozhatóak azok a szerepkörök, amelyek jogosultak a nézet megtekintésére. A navigátor minden nézetváltás előtt ellenőrzi, hogy a nézetváltást kezdeményező felhasználó jogosult-e a kért nézet megtekintésére.

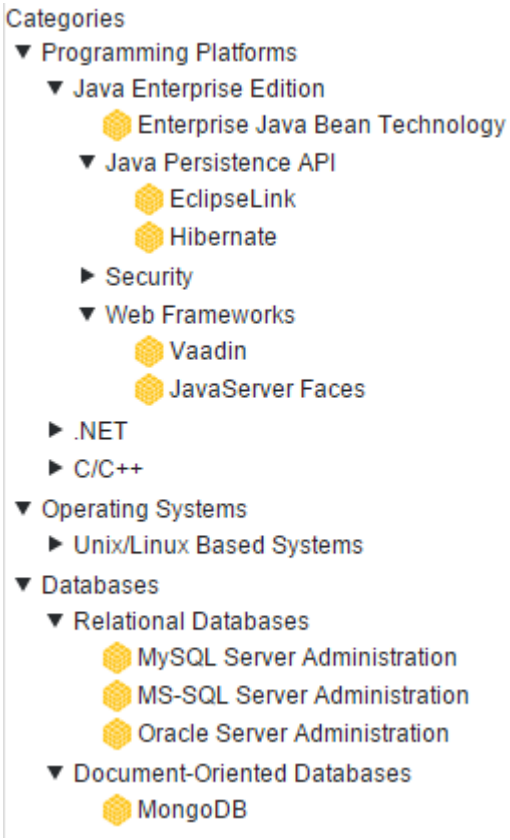
5. A grafikus felület felépítése és működése

A SkillMaster grafikus felületét (UI) egyetlen web alkalmazáson keresztül biztosítja. A különböző tartalmakat nézetek biztosítják, amelyek csak bizonyos szerepkörből érhetőek el.

A nézetek közti navigáció egy menüsáv segítségével történik, amely a UI részét képezi, ezért mindig elérhető. Az aktuális nézet a menüsáv alatt helyezkedik el, a UI egy speciális, nézetek számára fenntartott területén. Nézetváltás esetén csak ennek a területnek a tartalma változik, a teljes oldal nem töltődik újra.

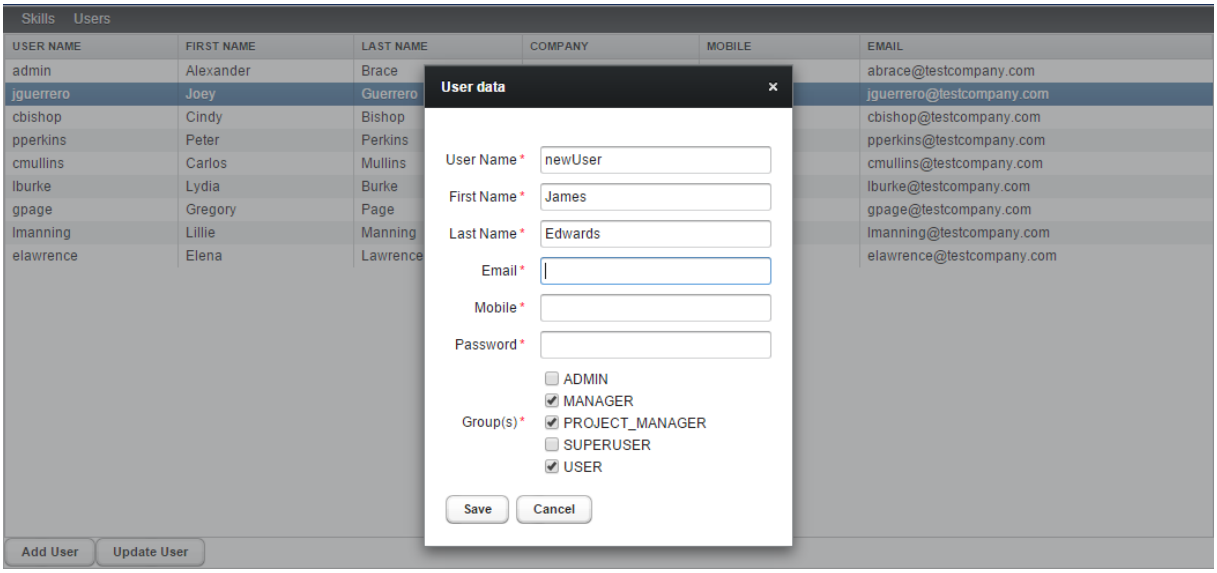
5.1. Adminisztráció

Az adminisztrációs felület két nézetből áll. Az első a kategóriák és szakismeretek megjelenítésére és kezelésére, a második nézet a felhasználók menedzsmentjére biztosít lehetőséget. A kategóriák és szakismeretek közti viszony hierarchikus grafikus ábrázolására több nézet esetében is szükség van, ezért a SkillMaster web alkalmazás saját Vaadin komponenst biztosít hozzá, amely a 6. ábra látható.



6. ábra Kategóriák és szakismeretek viszonyát megjelenítő grafikus komponens

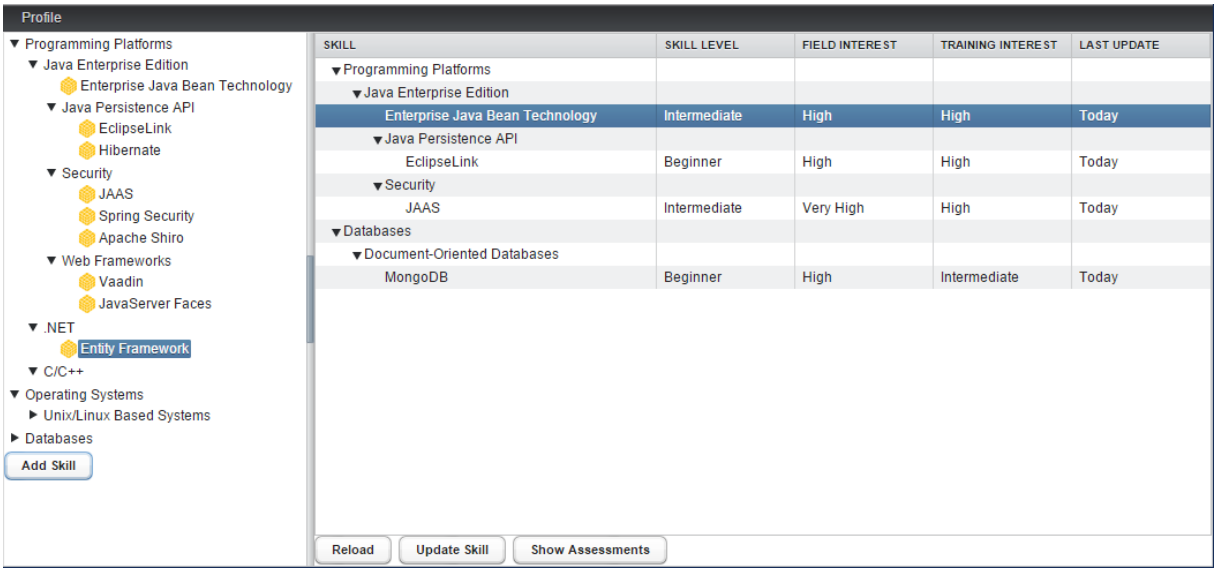
A második nézet lehetőséget biztosít az adminisztrátor számára a rendszerben lévő felhasználók megtekintésére, illetve új felhasználók létrehozására, valamint a meglévő felhasználók adatainak módosítására (7. ábra).



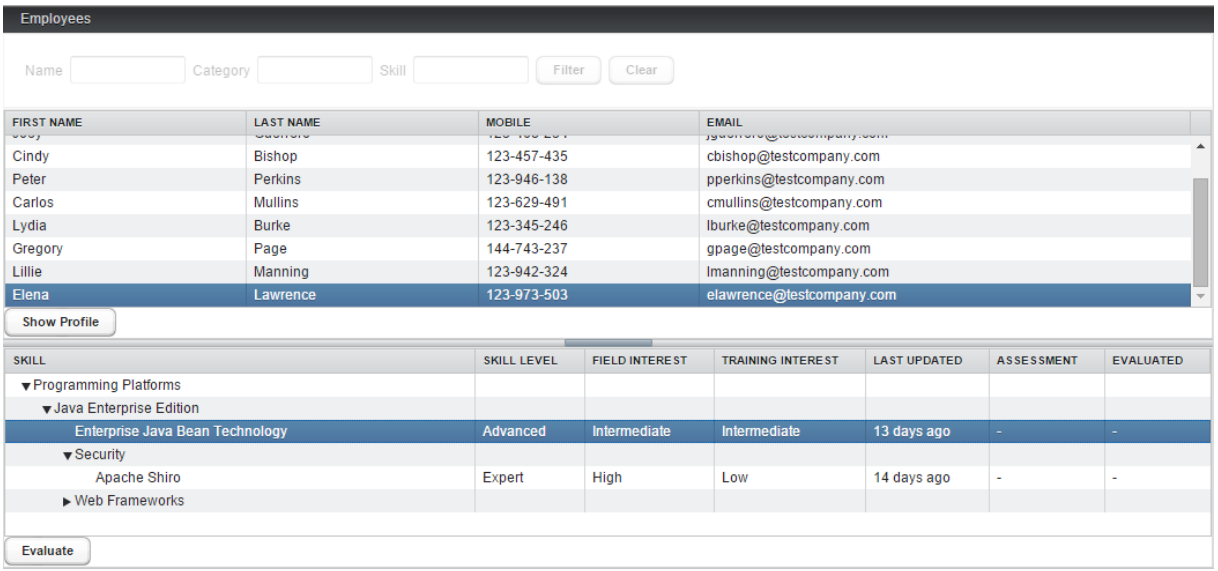
7. ábra Új felhasználó létrehozása az adminisztrációs felületen

5.2. Felhasználók

A felhasználó felület két nézetből áll. Az első nézet a személyes adatok megtekintését és módosítását szolgálja. A második lehetőséget biztosít szakismeretek profilhoz rendelésére, valamint a szaktudás, érdeklődési és továbbtanulási igény értékelésére, illetve a projektmenedzserek értékeléseinek megtekintésére (8. ábra).



8. ábra Saját szakismeretek megtekintését, értékelését és új szakismeretek felvételét biztosító nézet



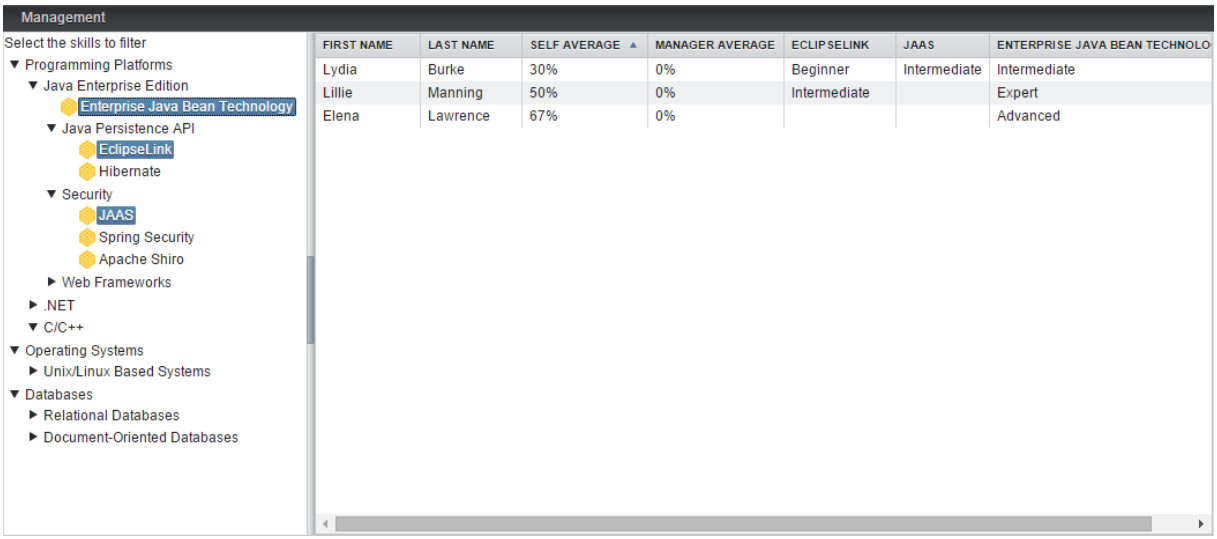
9. ábra Projektmenedzsment nézet, amely mutatja a felhasználók listáját (fent) és a kiválasztott felhasználó szakismereteiről szóló adatokat (lent)

5.3. Projektmenedzsment

A projektmenedzsment felület egyetlen nézetből áll, amely megjeleníti a felhasználók listáját, lehetőséget nyújt az egyes felhasználókhoz tartozó profil és önértékelés megtekintésére, továbbá a projektmenedzser értékelésének rögzítésére és frissítésére (9. ábra).

5.4. Menedzsment

A menedzsment felület lehetőséget biztosít felhasználók szakismeretek szerinti szűrésére. A rendszer a kiválasztott szakismeretek (vagy kategóriák) alapján megjeleníti azokat a felhasználókat, amelyek a megadott szakismeretek valamelyikével rendelkeznek.



10. ábra Felhasználók szakismeret szerinti szűrését megvalósító nézet

Következtetések és továbbfejlesztési lehetőségek

A projekt keretein belül sikerült létrehozni egy web alkalmazást, amely alkalmazottak szakismereteinek menedzselését teszi lehetővé vállalatok számára. A fejlesztés során sikerült feltérképezni az összetett vállalati rendszerek fejlesztésénél használható módszereket, eszközöket és Java technológiákat. Ezek közül megpróbálva a legalkalmasabbakat kiválasztani, sikerült egy skálázhatóság, megbízhatóság, karbantarthatóság és biztonság szempontjából megfelelő szoftverrendszert összeállítani. A szoftver SaaS rendszerként képes működni, de saját infrastruktúrára is telepíthető és megpróbálja kiküszöbölni a hasonló rendszerek bizonyos hiányosságait és korlátait.

A SkillMaster projekt továbbfejlesztésére több lehetőség is létezik. Ezek a lehetőségek főleg a SkillMaster célcsoportjának bővítésében rejlenek. Jelenleg a rendszer által nyújtott szolgáltatások kizárólag egy adott intézmény menedzsmentjét célozzák. A SkillMaster alapjául szolgáló rendszer könnyen kiegészíthető új, az adott intézményen belüli célcsoportok igényeinek kiszolgálásra. Néhány ilyen példa:

- Karrier-menedzsmenttel kapcsolatos funkcionalitások beépítése: ki milyen poszt(ok)ra/státusz(ok)ra pályázik, és mi az ezzel kapcsolatos tervezett „ütemterve”.
- Mentorálást és csapatok közötti szakmai tanácsadást támogató megoldások: pl. kik azok az alkalmazottak, akik ilyenre nyitottak, mikor és mennyi időt tudnak foglalkozni ilyesmivel.
- További HR megoldásokat támogató funkcionalitások beépítése (pl. általános elégedettségi/motivációs szinttel kapcsolatos kimutatások, problémák/igények és megoldások menedzsmentje).
- Az alkalmazás kiegészíthető egy további felülettel, amely globális keresésekre ad lehetőséget: pl. amennyiben több cég vállalja, hogy adatai anonim módon felhasználhatóak ilyen célra, akkor globális kimutatások készíthetőek és ezek alapján tanácsadás biztosítható az illető cégeknek.

Hivatkozások

1. *Scrum Hivatalos Weboldal.* [Online] https://www.scrum.org/ [Utolsó megtekintés dátuma: 2015. április 26.].

2. *Trello Hivatalos Weboldal.* [Online] Trello, Inc. https://trello.com/ [Utolsó megtekintés dátuma: 2015. április 26.].

3. *XWiki Hivatalos Weboldal.* [Online] http://www.xwiki.org/ [Utolsó megtekintés dátuma: 2015. április 26.].

4. *Mercurial Hivatalos Weboldal.* [Online] http://mercurial.selenic.com/ [Utolsó megtekintés dátuma: 2015. április 26.].

5. *RhodeCode Hivatalos Weboldal.* [Online] RhodeCode, Inc. https://rhodecode.com/ [Utolsó megtekintés dátuma: 2015. április 26.].

6. *TortoiseHG Hivatalos Weboldal.* [Online] http://tortoisehg.bitbucket.org/ [Utolsó megtekintés dátuma: 2015. április 26.].

7. *Maven Hivatalos Weboldal.* [Online] https://maven.apache.org/ [Utolsó megtekintés dátuma: 2015. április 26.].

8. *Artifactory Hivatalos Weboldal.* [Online] JFrog, Ltd. http://www.jfrog.com/artifactory/ [Utolsó megtekintés dátuma: 2015. április 26.].

9. Paul M. Duvall, Steve Matyas, Andrew Glover, *Continuous Integration: Improving Software Quality and Reducing Risk*, Addison-Wesley, 2007.

10. *Jenkins Hivatalos Weboldal.* [Online] https://jenkins-ci.org/ [Utolsó megtekintés dátuma: 2015. április 26.].

11. *SonarQube Hivatalos Weboldal.* [Online] SonarSource S.A, Switzerland. http://www.sonarqube.org/ [Utolsó megtekintés dátuma: 2015. április 26.].

12. *Eclipse Hivatalos Weboldal.* [Online] Eclipse Foundation. http://www.eclipse.org/ [Utolsó megtekintés dátuma: 2015. április 26.].

13. *A JPA Specifikációval Kapcsolatos Erőforrások Hivatalos Oldala.* [Online] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html [Utolsó megtekintés dátuma: 2015. április 26.].

14. *EclipseLink Hivatalos Weboldal.* [Online] http://eclipse.org/eclipselink/ [Utolsó megtekintés dátuma: 2015. április 26.].

15. *BeanValidation Specifikáció Hivatalos Weboldala.* [Online] http://beanvalidation.org/ [Utolsó megtekintés dátuma: 2015. április 26.].

16. *Az EJB Specifikációval Kapcsolatos Erőforrások Hivatalos Oldala.* [Online] http://www.oracle.com/technetwork/java/javaee/ejb/index.html [Utolsó megtekintés dátuma: 2015. április 26.].

17. Andrew Lee Rubinger, Bill Burke, *Enterprise JavaBeans 3.1, 6th Edition*, O'Reilly Media, 2010.

18. *JMS Hivatalos Weboldala.* [Online] http://docs.oracle.com/javaee/6/tutorial/doc/bncdq.html [Utolsó megtekintés dátuma: 2015. április 26.].

19. *GlassFish Hivatalos Weboldal.* [Online] https://glassfish.java.net/ [Utolsó megtekintés dátuma: 2015. április 26.].

20. *JPQL Specifikáció Hivatalos Dokumentációja.* [Online] http://docs.oracle.com/cd/E15523_01/apirefs.1111/e13946/ejb3_langref.html [Utolsó megtekintés dátuma: 2015. április 26.].

21. Az Interceptor Tervezési Minta Dokumentációja. [Online] http://www.servicedesignpatterns.com/WebServiceInfrastructures/ServiceInterceptor [Utolsó megtekintés dátuma: 2015. április 26.].

22. *SLF4J Hivatalos Weboldal.* [Online] http://www.slf4j.org/ [Utolsó megtekintés dátuma: 2015. április 26.].

23. *LOG4J Hivatalos Weboldal.* [Online] http://logging.apache.org/log4j/2.x/ [Utolsó megtekintés dátuma: 2015. április 26.].

24. Martin Fowler, *Inversion of Control Containers and the Dependency Injection pattern* [Online] http://martinfowler.com/articles/injection.html, 2004, [Utolsó megtekintés dátuma: 2015. április 26.].

25. *GWT Hivatalos Weboldal.* [Online] http://www.gwtproject.org/ [Utolsó megtekintés dátuma: 2015. április 26.].

26. JSF Hivatalos Dokumentáció. [Online] http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html [Utolsó megtekintés dátuma: 2015. április 26.].

27. *Vaadin Hivatalos Weboldal.* [Online] https://vaadin.com/home [Utolsó megtekintés dátuma: 2015. április 26.].

28. Marco Grönroos, *The Book of Vaadin: Vaadin 7 Edition - 4th Revision,* Vaadin, Ltd., 2014.

29. CDI Hivatalos Dokumentáció. [Online] http://docs.oracle.com/javaee/6/tutorial/doc/giwhl.html [Utolsó megtekintés dátuma: 2015. április 26.].

30. JAAS Hivatalos Dokumentáció. [Online] http://docs.oracle.com/javase/7/docs/technotes/guides/security/jaas/JAASRefGuide.html [Utolsó megtekintés dátuma: 2015. április 26.].