XIX. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)
Kolozsvár, 2016. május 19-22.

A ProfiNet szolgáltatáskereső platform

Szerzők:
Vass Lilla
Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, Informatika szak, 3. évfolyam
Balog Csongor Loránd
Babeş-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika Kar, Informatika szak, 3. évfolyam

Témavezetők:
dr. Simon Károly
egyetemi adjunktus, BBTE, MIK, Magyar Matematika és Informatika Intézet
Kandó Norbert
projektmenedzser, Codespring
Kintzel Levente
projektmenedzser, Codespring

Kivonat

A ProfiNet szolgáltatások és szakemberek keresésére alkalmas webes platform. Felhasználói különböző szolgáltatásokat nyújtó szakemberek, illetve szolgáltatásokat kereső, szakmai segítséget igénylő személyek.

A projekt célja létrehozni egy olyan kommunikációs csatornát a felhasználók között, amely megkönnyíti a kapcsolatfelvételt és a megrendelt munkák állapotának nyomon követését mind a két fél számára. Ugyanakkor védi a szolgáltatásokat kínálók adatait és optimalizálja a keresések eredményeit.

A szakemberek ajánlatai előbbi munkáikra kapott visszajelzések alapján kerülnek rangsorolásra. A hirdetéseket a felhasználó szűrheti nyelv, kategória, illetve kulcsszavak alapján. Megnézheti az előbbi kliensek véleményét adott szakemberről, majd jóváhagyott kapcsolatfelvételi kérés után üzeneteket küldhet, időpontot kérhet. A szolgáltatást kínáló kitöltheti saját profilját, hirdethet, felülbírálhatja a kapcsolatfelvételi- és időpontkéréseket, jelezheti, hogy milyen periódusokban aktív.

A dolgozat a projekt implementációjához felhasznált módszereket, eszközöket és technológiákat mutatja be, kitérve az alkalmazás szerkezetére és a továbbfejlesztési lehetőségekre is.

Tartalomjegyzék

Bevezető

A dolgozat a PofiNet szolgáltatáskereső platform megvalósítását és működését mutatja be. A szoftver célja, hogy segítsen a felhasználóknak megbízható szakembereket találni, ezekkel a szakemberekkel kommunikálni, szolgáltatásokat keresni és igénybevenni. Ugyanakkor a szakemberek számára egy hirdetési felületet biztosít, láthatóvá téve szolgáltatásaikat a potenciális ügyfeleknek.

A működés szemléltetésére egy (sajnos) gyakran előforduló konkrét példa: csőtörés. A legelső gondolatunk ilyenkor: vízszerelő kell. De nem biztos, hogy könnyű lesz gyorsan megfelelő szakembert találni, főként, ha például újak is vagyunk a városban. Ilyenkor következik, hogy baráttól, ismerőstől próbálunk kérni telefonszámot, elérhetőséget. Az Interneten is próbálkozhatunk, de itt sem biztos, hogy könnyen találunk olyan szakembert, aki megbízható, kellőképpen szakképzett, elég közel van hozzánk és beszél velünk legalább egy közös nyelvet. Az ilyen helyzeteken próbálna segíteni a ProfiNet.

A rendszer szakemberként regisztrált felhasználói rangsorolva vannak a felhasználók visszajelzései alapján. Felhasználói profiljukból többek között kiderül órarendjük, földrajzi helyzetük, illetve az, hogy milyen nyelveket beszélnek. Ha nem elég megbízhatóak, a felhasználók közösségének visszajelzései alapján ez is hamar nyilvánvalóvá válik.

A ProfiNet ugyanakkor a szakemberek személyes adatait is védi. A szolgáltatást biztosító személy megválaszthatja, hogy kivel osztja meg a platformra felvitt adatait, üzenetet válthat a leendő klienssel, illetve felviheti a munkára szánt időintervallumot, ezzel is garantálva a félreértések elkerülését.

A dolgozatban bemutatott projekt két fő részből áll: egy központi szerverből és egy webes kliens alkalmazásból. A rendszer adminisztrátorai a webes felületen keresztül kezelhetik az adatokat, statisztikákat láthatnak a rendszer állapotáról. A felhasználók kéréseket küldhetnek a rendszerben regisztrált szakembereknek, üzeneteket válthatnak velük és magukat is szakembernek jelölhetik a megfelelő profiladatok megadásával. A szakemberek szintén a rendszer felhasználói, további funkcionalitás számukra, hogy a kliensek kéréseit kezelhetik.

A központi szerver Java alapú, Spring keretrendszerek használatával készült. A webes felhasználói felület fejlesztésére használt legfontosabb technológia az AngularJS.

A dolgozat első két fejezetében a projektben felhasznált eszközök és technológiák lesznek bemutatva. A következő részben a ProfiNet rendszer működése és megvalósítása lesz ismertetve. Az utolsó részben a továbbfejlesztési lehetőségek és a projekttel kapcsolatos következtetések lesznek felsorolva.

A ProfiNet projekt a Codespring Mentorprogram keretein belül, a szakmai gyakorlat időszaka alatt indult el 2015 augusztusában. A projekt szakmai koordinálására a Codespring három szakmai irányítót kért fel, Kandó Norbertet, Kintzel Leventét és dr. Simon Károlyt. Segítségükért hálás köszönet.

1. Felhasznált technológiák

A ProfiNet szerver oldala a Spring [8] keretrendszerre épül, amely minden funkcionalitást tartalmaz, amire egy összetettebb web alkalmazásnak szüksége lehet. A szerver MySQL adatbázist használ.

1.1 A Spring keretrendszer

A Spring egy platformfüggetlen, nyílt forráskódú, Java programozási nyelven alapuló keretrendszer, amely egy Inversion of Control (IoC) konténeren keresztül biztosítja a komponensek menedzsmentjét és a Dependency Injection (DI) minta alkalmazásának lehetőségét.

A Spring keretrendszer pehelysúlyú (lightweight), ami arra utal, hogy a különböző Spring modulokat könnyű beépíteni vagy lecserélni egy alkalmazáson belül. Mivel a Spring moduláris, csak azokat a részeit kell használnunk, amelyekre valóban szükségünk van az alkalmazásunk működéséhez.

A Spring projekt hozzávetőlegesen húsz keretrendszert tartalmaz, a továbbiakban a dolgozatban felhasznált fontosabb keretrendszerek lesznek röviden bemutatva.

1.1.1 Spring Boot

A Spring Boot gyors megoldást biztosít egy Spring alapú alkalmazás létrehozására, általában kisebb projektek esetében használják. A Spring Boot keretrendszer alapelve a konvenció a konfiguráció felett (convention-over-configuration), így próbálja csökkenteni a programozó által projekt-konfigurációra fordított időt, könnyebb és gyorsabb fejlesztést biztosítva.

1.1.2 Spring Data JPA

A Java Persistence API (JPA) meghatároz egy egységesített eljárást a rendszer központi entitásainak a kezelésére, illetve leképezésére egy relációs adatbázis sémára. A JPA meghatároz egy teljes objektum-relációs leképezést, amely kétféleképpen konfigurálható: annotációs mechanizmus vagy XML leíróállományok alkalmazásával.

A Spring Data JPA keretrendszer tulajdonképpen egy absztrakciós szint a JPA fölött. A keretrendszer segítségével a legtöbb esetben nem szükséges implementálni az adathozzáférési réteg komponenseit, elegendő a specifikus elnevezési konvenciók alapján létrehozott interfészeket létrehozni. Az összetetteb lekérdezések esetében az interfészek metódusain alkalmazott annotációk segítségével JPQL query-k is megadhatóak. Természetesen, speciális esetkben lehetőség van az interfészek megvalósítására is, így a JPA minden funkcionalitása elérhető.

A ProfiNet esetében a háttérben a Hibernate [11] keretrendszer szolgál JPA implementációként.

1.1.3 Spring Data ElasticSearch

Bizonyos esetekben, például ha teljes szöveg alapú keresést (full text search) szeretnénk megvalósítani, a relációs adatbázisok által biztosított indexelési és keresési lehetőségek már nem biztos, hogy megfelelő gyorsaságot és hatékonyságot nyújtanak. Az ilyen esetekben jó alternatívát jelenthetnek az olyan specializált technológiák és rendszerek, mint a Lucene, illetve az erre épülő Elasticsearch. A Spring egy absztrakciós szint bevezetésével is támogatja az Elasticsearch használatát, a Spring Data Elasticsearch keretrendszer által. Ez egy rugalmas és hatékony, nyílt forráskódú keretrendszer, amely az adatokat JSON formában tárolja, valós idejű kereséseket és elemzéseket téve lehetővé.

A ProfiNet esetében a felhasználóknak lehetőségük van teljes szöveg alapú keresésre is a szakértők profiljában megadott adatok alapján. Ennek a funkcionalitásnak a biztosításához használja a rendszer az előbbiekben említett technológiákat.

2.1.4 Spring Security

A Spring a szoftverrendszerek biztonsági mechanizmusával kapcsolatos megoldásokat is biztosít a Spring Security keretrendszer formájában, a ProfiNet is ezt a keretrendszert alkalmazza. A keretrendszer segítségével azonosítani lehet a felhasználókat, illetve ellenőrzni lehet hozzáférési jogaikat bizonyos szolgáltatásokhoz.

A Spring Security keretrendszer beépítése az alkalmazásokba viszonylag egyszerű, néhány konfigurációs adaton és egyszerű annotáción alapszik. A technológia többféle autentikációs és autorizációs mechanizmust biztosít, például egyszerű felhasználónév és jelszó

alapút, vagy token alapút, amelyre saját megközelítés, vagy az OAuth2 szabványnak megfelelő megoldás is használható. A ProfiNet token alapú azonosítást alkalmaz.

1.1.5 Spring MVC REST

A REST (Representational State Transfer) osztott rendszereken belüli kommunikációt meghatározó szoftver architektúra. A modellnek megfelelően az adatok erőforrásokként vannak kezelve, amelyekhez a következő típusú műveletekkel lehet hozzáférni:

- POST – új erőforrás létrehozása
- PUT – létező erőforrás módosítása
- GET – erőforrás lekérése
- DELETE – erőforrás törlése

A Spring Web MVC keretrendszer alapvetően MVC (model-view-controller) elveknek megfelelő web alkalmazások fejlesztését támogatja. Ugyanakkor REST architektúrával rendelkező alkalmazások fejlesztését is elősegíti. Ezt az annotációs mechanizmus alapján valósítja meg. A fontosabb annotációk:

- @RequestMapping: az erőforrás elérési útvonala, a kérés típusa, valamint a visszatérítendő adat formátuma állítható be,
- @RestController: erőforrások létrehozását teszi lehetővé,
- @RequestBody, @RequestParam, @PathVariable: paraméterek forrását specifikálják.

Adatátvitel szempontjából az adatok többféle formátumban továbbítódhatnak: egyszerű szöveg, JSON, HTML, XML stb.

A ProfiNet esetében az adatátvitel JSON formátumban történik. Az adatcsere a kliens és a szerver között DTO (Data Transfer Object) mintának megfelelően valósul meg.

1.2 Kliens oldali webes technológiák

Az AngularJS egy nyílt forráskódú, Google által kifejlesztett MVW (Model-View-Whatever) keretrendszer. JavaScript alapú, web alkalmazások kliens oldali részének fejlesztését támogatja.

A keretrendszer segítségével a kliens modul négy fő részre osztható: modell réteg, megjelenítés (view) réteg, vezérlő réteg és szolgáltatás réteg.

A modell réteg tartalmát a JavaScript változók szolgáltatják, a $scope objektum segítségével. Ha a modell változik, akkor a felületen megjelenített értékek is változnak.

A kontroller réteget JavaScript függvények alkotják, ez a megjelenítéshez szükséges logikai rész. Minden kontroller saját $scope-pal rendelkezik, de a $rootscope bárhol egységesen elérhető, globális.

A view réteget a HTML oldalon belüli kód alkotja, amit az AngularJS kiegészít saját attribútumokkal, amelyek segítségével megadható, hogy milyen adatok jelenjenek meg, valamint az, hogy a felület adott eseményekre hogyan reagáljon.

A szolgáltatás réteg a szerverrel való kommunikáció megvalósításáért felelős.

Az AngularJS lehetőséget ad különböző direktívák megírására is, amelyek a DOM elemek manipulációját teszik lehetővé, illetve adott viselkedési móddal ruházzák fel ezeket.

Az AngularJS támogatja a különböző HTML oldalaknak megfelelő URL-ek névhez való kötését is. A HTML oldalon belül alkalmazott direktíva által megadható, hogy az adott URL-hez kötött oldal mikor töltődjön be.

A ProfiNet webes felülete a Bootstrap keretrendszer segítségével van felépítve. A Bootstrap [10] egy előre megírt eszközkészlet, amely a web kliens stílusának és kinezetének megtervezésére ad lehetőséget. Az előre megírt CSS tulajdonságok mellett számos JavaScript bővítménnyel is rendelkezik. A Twitternél fejlesztették ki, és 2011 óta open-source licensszel rendelkezik.

2. A ProfiNet projekt

A következőkben a rendszer funkcionalitásai, modellje, illetve architektúrája lesz bemutatva. A megvalósítás fontosabb részleteinek leírása mellett, röviden az alkalmazás működést is tárgyalja a fejezet.

2.1 Fontosabb követelmények, funkcionalitások

A ProfiNet rendszert két fő komponens alkotja: a szerver, illetve a webes felhasználói felület. A szerver felel az adatok perzisztenciájáért, az üzleti logika megvalósításáért és a klienssel való kommunikációért. A webes felület biztosítja a felhasználó és a szerver közötti kommunikációt.

2.1.1 A ProfiNet szerver

A szerver oldal négy rétegre bontható: modell, amely a központi entitásokat tartalmazza, adathozzáférési réteg, alkalmazáslogikai réteg, illetve a kommunikációért felelős RESTful API. A szerver fontosabb functionalitásai:

- Biztosítja a modellobjektumok létrehozását, lekérését, módosítását és törlését, tehát az adathozzáférési réteget, illetve az indexelést.
- Az üzleti logika réteg végzi el az adatokkal kapcsolatos műveleteket, illetve statisztikákat készít.
- A szolgáltatási réteg RESTful webszolgáltatásokon keresztül biztosítja a kommunikációt a felhasználói felülettel.
- A szerver biztonsági megoldásokat biztosít: kezeli a felhasználókat és azok jogosultságait.

2.1.2 A ProfiNet Web UI

A Web UI feladata a webes felhasználói felületek biztosítása. Főbb funkcionalitásai:

- Lehetőséget biztosít regisztrálásra, bejelentkezésre, felhasználó jelszavának módosítására.

- Megjeleníti a szakemberek profiljait és ajánlatait, keresést és szűrést biztosít ezekkel kapcsolatban, továbbá térképen is bejelöli ezeket a Google Maps API segítségével.
- Biztosítja a kommunikáció lehetőségét a felhasználók és a szakemberek között.
- Lehetőséget ad a szakemberek által elvégzett munkák értékelésére.
- Biztosítja a felhasználók számára, hogy a megfelelő profiladatok megadása után szakemberként használják a rendszert, szolgáltatásokat ajánljanak a többi felhasználónak.
- Adminisztrátor jogosultsággal rendelkező felhasználóknak biztosítja az adatbázis elemek szerkesztését, rendszerstatisztikákat mutat és lehetőséget nyújt a felhasználók menedzselésére.

2.2 Környezeti elemzés

Az edu.codespring.profinet.domain csomag a ProfiNet szoftver központi entitásait reprezentáló JavaBean-eket tartalmaz, ezt szemlélteti az 1. ábra.

A domain osztályok kiterjesztik a BaseEntity osztályt, amely biztosítja az elsődleges kulcsnak megfelelő azonosítót, implementálja a java.io.Serializable interfészt.

A User osztály példányai a rendszer felhasználóinak adatait tárolják. Az osztály kiterjeszti az AbstractAuditingEntity osztályt, amelyen keresztül nyomon követhetőek a felhasználó entitás adatainak módosítására vonatkozó információk. Az Authority osztály a felhasználó rendszeren belüli szerepkörét reprezentálja (*ADMIN/USER/EXPERT*).

Az Expert osztály példányai a rendszer szakembereinek adatait tárolják. Szakemberként lehetőségünk van információt megadni a beszélt nyelvekről és a szakterületünkre vonatkozó kulcsszavakról, ezeket az információkat a Language és a Keyword osztályok példányai tárolják.

Az ExpertField osztály egy szakember szakterületeihez tartozó információkat reprezentál. A szakterületeket a Field osztály tárolja, ezek közül tud választani egy szakember saját szakterületeinek meghatározásánál.

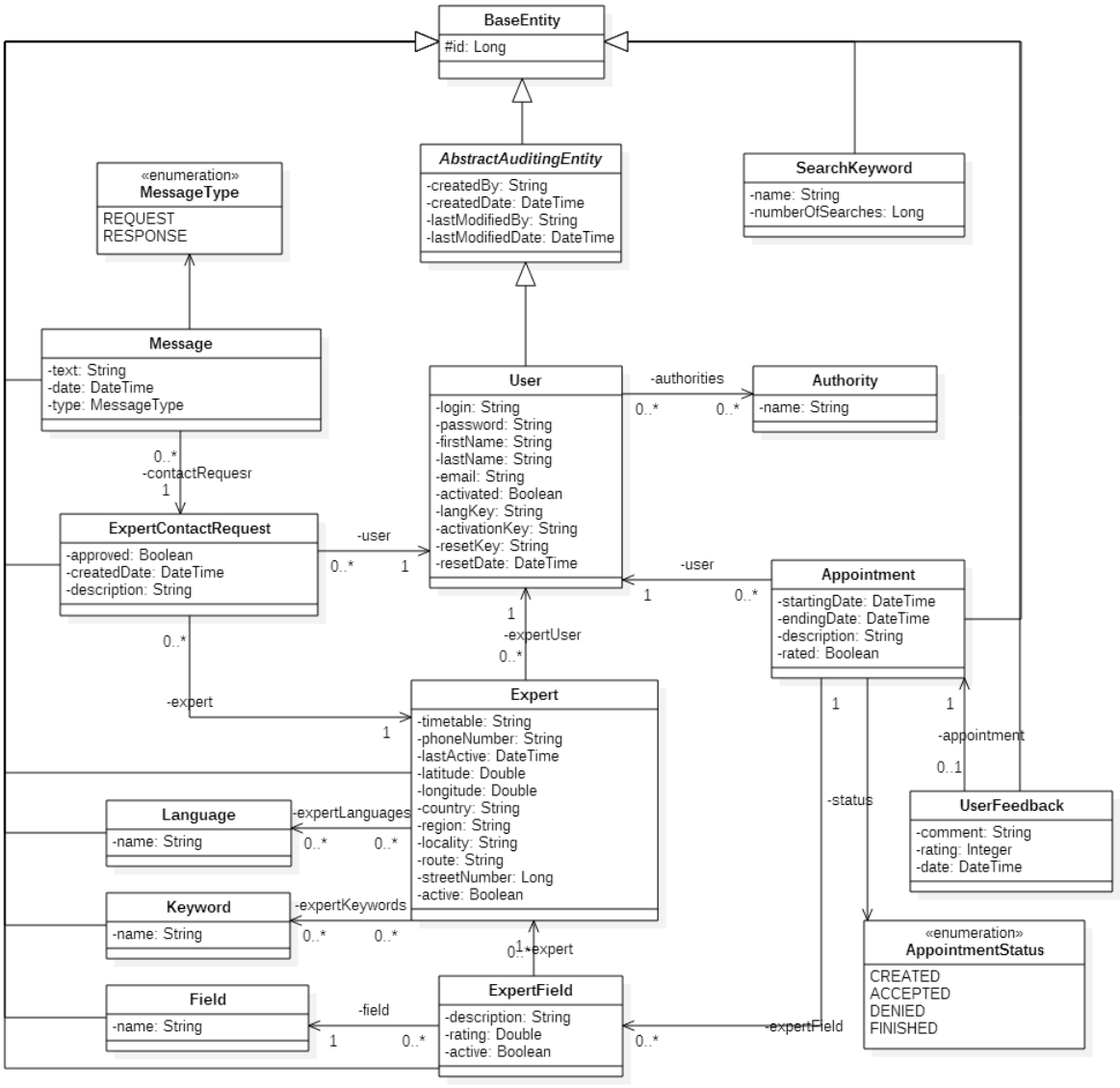
Az ExpertContactRequest osztály példányai a rendszer felhasználói közötti kapcsolatokat tárolják.

A Message osztály a kapcsolatban lévő felhasználók és szakemberek közti üzeneteket reprezentálja. Minden üzenethez tartozik egy típus amit a MessageType enum határozza meg attól függően, hogy felhasználó vagy szakember szerepkörből küldtük az üzenetet.

Az Appointment osztály példányai az igényelt munkára vonatkozó információkat tárolnak. Minden munkának van egy állapota, ezt az állapotot az AppointmentStatus enum határozza meg.

A UserFeedback osztály a felhasználók visszajelzéseit reprezentálja, ezeket a munka megkezdése után hozhatja létre a felhasználó, ezáltal értékelve a szakembert.

A SearchKeyword osztály példányai a felhasználó által keresett kulcsszavakat tárolják.



1. ábra:*A domain osztályok vázlatos diagramja (hiányoznak a konstruktorok, a getter, illetve setter metódusok, valamint az Object ősosztály újradefiniált metódusai)*

2.3 Architektúra

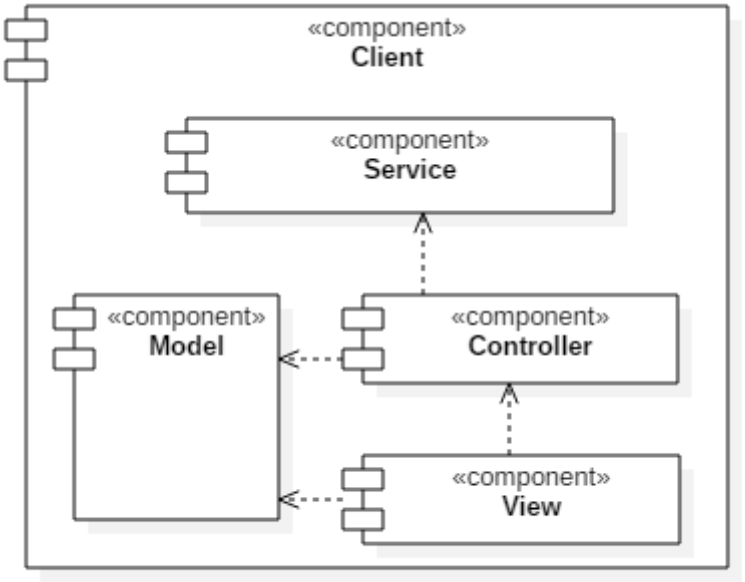
A 2. és 3. ábrákon a ProfiNet alkalmazás rétegei, valamint a komponensek közötti kapcsolatok láthatóak.

Az alkalmazás adatait a ProfiNet egy MySQL relációs adatbázisban tárolja. Azoknak az adatoknak az indexelését, amelyek esetében teljes szöveg alapú keresés szükséges, egy Lucene alapú rendszer biztosítja.

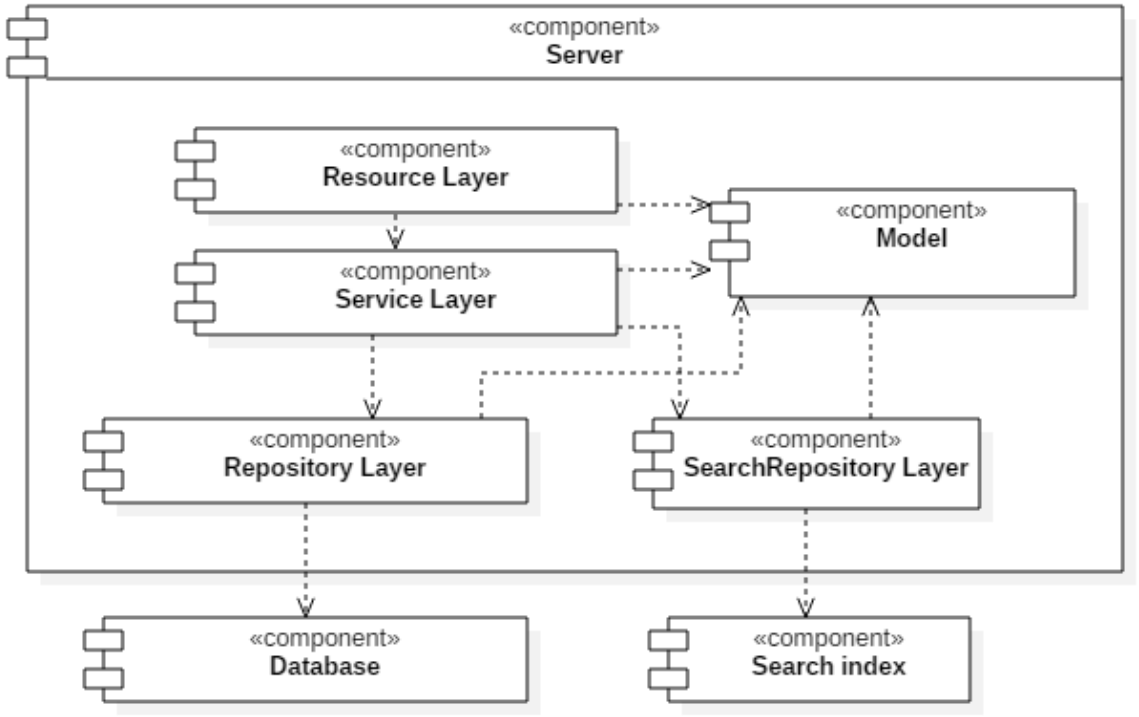
A Model komponens tartalmazza az alkalmazás entitásainak megfelelő osztályokat, a Repository réteg feladata az ezekkel végzett adatbázis műveletek megvalósítása. A Service réteg a Repository rétegtől kapott adatokkal kapcsolatos üzleti logikát implementálja.

A szerver a funkcionalitásait webszolgáltatások formájában teszi elérhetővé. A REST Resource réteg fogadja a kliensek kéréseit és a Service rétegtől kapott adatokat továbbítja a klienseknek.

A Client modul alkotja a webes felhasználói felületet, illetve az ezen keresztül beérkező kéréseket továbbítja a szervernek. A Client modulon belül az MVVM tervezési minta érvényesül. Az adatok szervertől való lekéréséért a Service modul felelős. Az adatokat a Controller modul dolgozza fel, majd módosítja a kliens oldali modelleket, így a View rétegen belül mindig a megfelelő adatok lehetnek megjelenítve.



2. ábra: *A ProfiNet rendszer architektúrája – Client modul*



3. ábra: *A ProfiNet rendszer architektúrája – Server modul*

2.4 Rövid összefoglaló a megvalósításról

A megvalósítás fontosabb részletei a két fő komponensre lesznek leosztva: a backend-re és a webes felhasználói felületre.

2.4.1 ProfiNet Backend

A ProfiNet esetében a backend Javaban van megírva a Spring keretrendszert felhasználva. Négy rétegre osztható: modell, repository, szolgáltatás és webszervíz rétegek.

A modell elemei egy osztélyhierarchia részei, a közös tulajdonságok az ősosztályok szintjén vannak kiemelve. A perzisztenciáért a Java Persistence API (JPA) Hibernate implementációja felelős, a modelleket reprezentáló JPA entitások ORM leképezéséhez szükséges meta-adatok annotációk segítségével vannak megadva. A fejlesztés során történt modellbeli változásokat a Liquibase menedzseli, ezek alapján frissítve a relációs sémát. A relációs adatbázis mellett fontos szerepe van a teljes szöveg alapú keresés támogató indexelő

rendszernek is. Az Elasticsearch egy Lucene-alapú keresőszerver, amely a kapott adatokat JSON-ban tárolja, biztosítva az adatok közötti hatékony keresést és szűrést.

Az alkalmazás logikája a szolgáltatás rétegen belül van megvalósítva. Ugyanitt történik az e-mailes notifikációk kiküldése is. Erre a konfigurációs fájlban (application.yml) megadott adatok alapján beállított e-mail fiókot és szervert használ a rendszer.

A biztonsági mechanizmus és a felhasználók jogosultságai a Spring Security-re vannak bízva. Az autentikáció állapot nélküli („stateless”), a kliens-oldal minden bejelentkezéskor kap egy tokent, ami alapján a további interakciók során azonosítva lesz a felhasználó. Adott idő eltelte után a token lejár, és amennyiben a felhasználó már nem aktív az oldalon, a böngésző automatikusan kijelentkezett állapotba helyezi.

A szerver a web klienssel RESTful webszolgáltatásokon keresztül kommunikál. Az adatokat Mapper objektumok (Assembler-ek) által DTO-vá (Data Transfer Object) alakítva küldi át, JSON formátumban.

A futási időben történő logolás aspektus-orientáltan történik, ezt biztosítja a Spring Boot konfigurációs állományán belül megadott adatok alapján a Log4j és az SLF4J.

2.4.2 ProfiNet Web User Interface

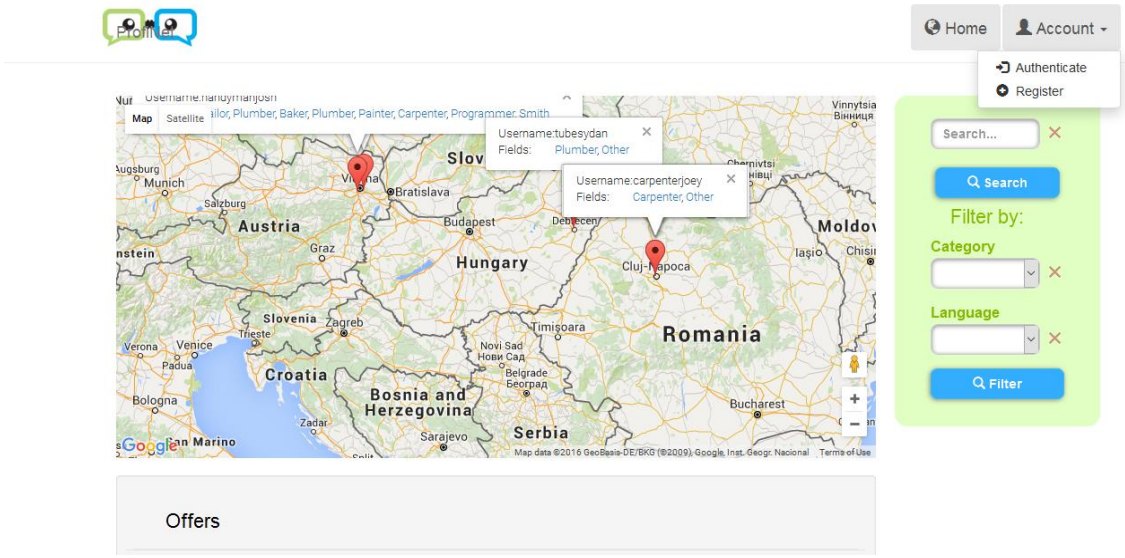
A webes felület megvalósítására HTML, CSS és JavaScript technológiák voltak használva.

A webes felület tervezéséhez a Bootstrap előre megírt eszközkészlet nyújtott segítséget. A felület MVVM tervezési minta szerinti megvalósítását az AngularJS JavaScript alapú keretrendszer biztosította.

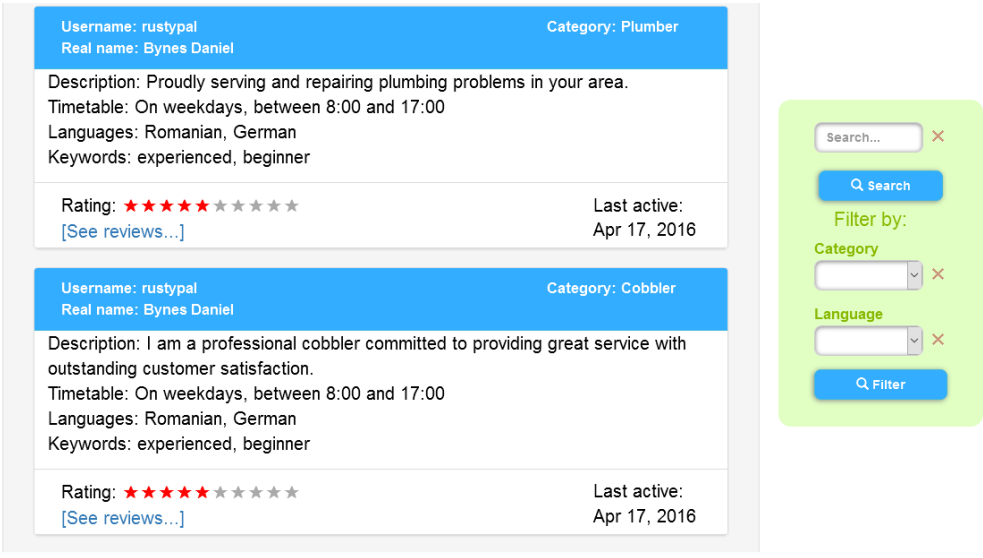
A kommunikáció REST alapú, a megjelenítéshez szükséges adatok JSON formátumban továbbítódnak.

2.5 A ProfiNet működése

A ProfiNet alkalmazás webböngészőből érhető el. Bejelentkezés előtt a felhasználó lehetőségei limitáltak: regisztrálhat, beléphet, láthatja az ajánlatokat a térképen megjelenítve, illetve listában felsorolva, korábbi visszajelzések alapján létrehozott sorrendben. A lista szűrhető nyelv, kategória és kulcsszó alapján. Nem biztosított viszont kapcsolatfelvételi lehetőség (4-5. ábrák).

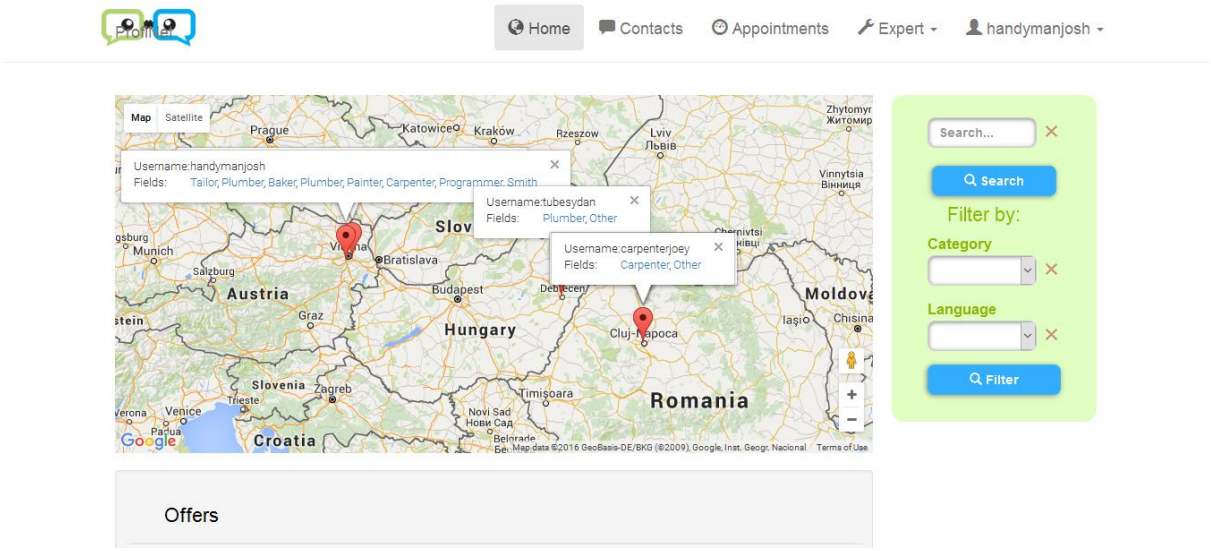


4. ábra: *A felhasználó által bejelentkezés előtt elérhető felület*

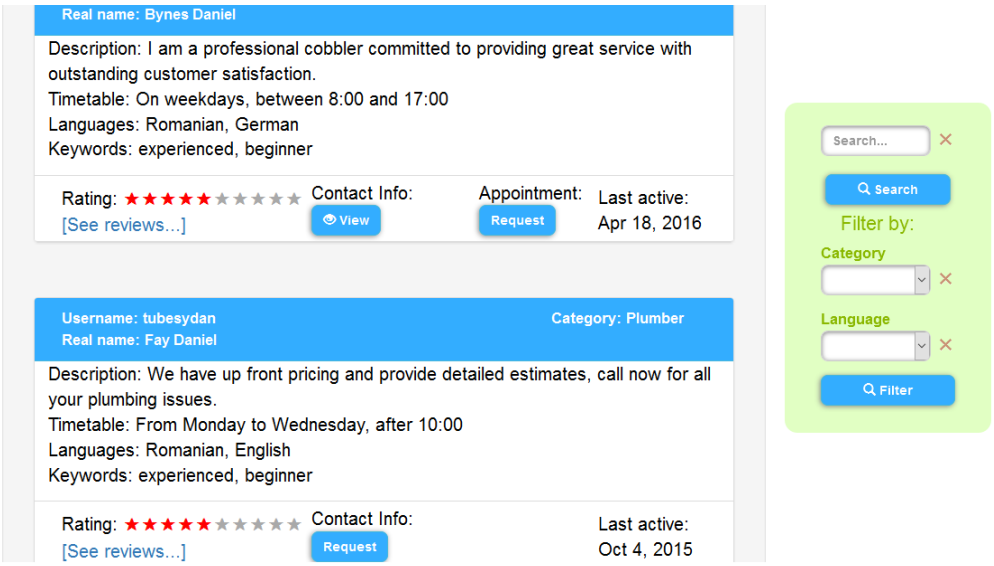


5. ábra: *A főoldalon az ajánlatok lista formában is megjelennek*

A felhasználó az Account menü alatti Registration, illetve Authenticate opciókat használhatja regisztrációra, illetve bejelentkezésre (4.ábra).

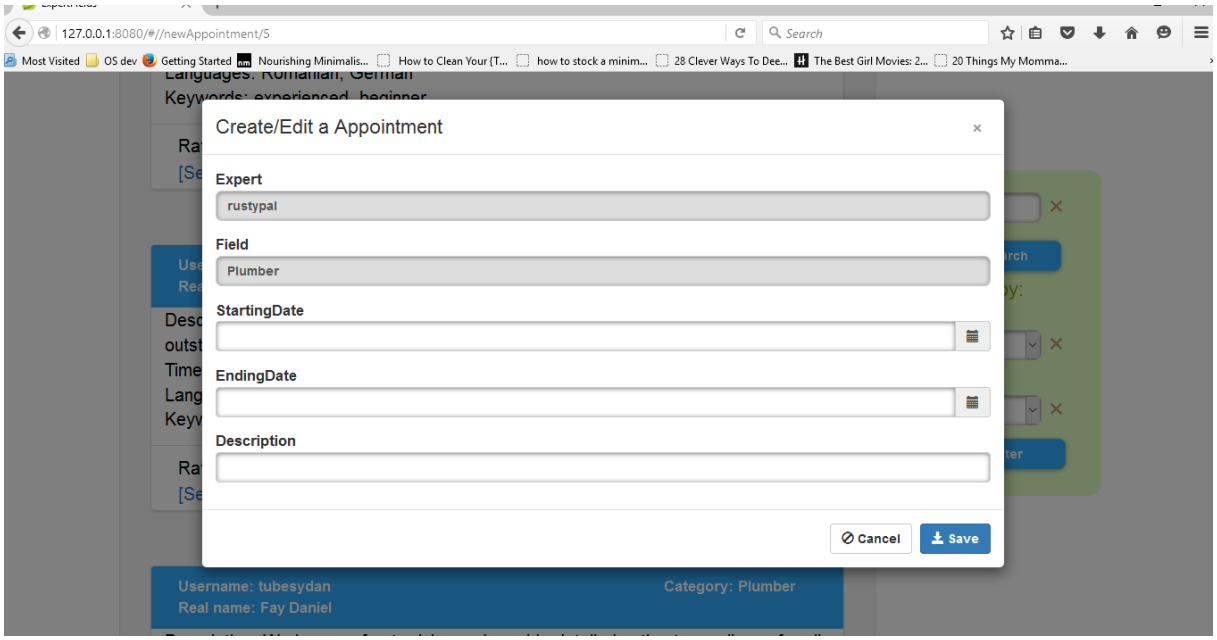


6. ábra: *Bejelentkezés után a menüsor kibővül.*



7. ábra: *A bejelentkezett felhasználók számára a listanézetben új opciók jelennek meg.*

Bejelentkezés után a menüsor kibővül, a felhasználó megtekintheti a főoldalon megjelenő ajánlatok hirdetőinek profilját, elérhetőségi információkat igényelhet (6-7. ábrák). Amennyiben a szakember úgy dönt, hogy megosztja az adatait az illető felhasználóval, egy külön nézetben jóváhagyhatja a kérést, vagy elutasíthatja az, illetve később is visszavonhatja jóváhagyását.

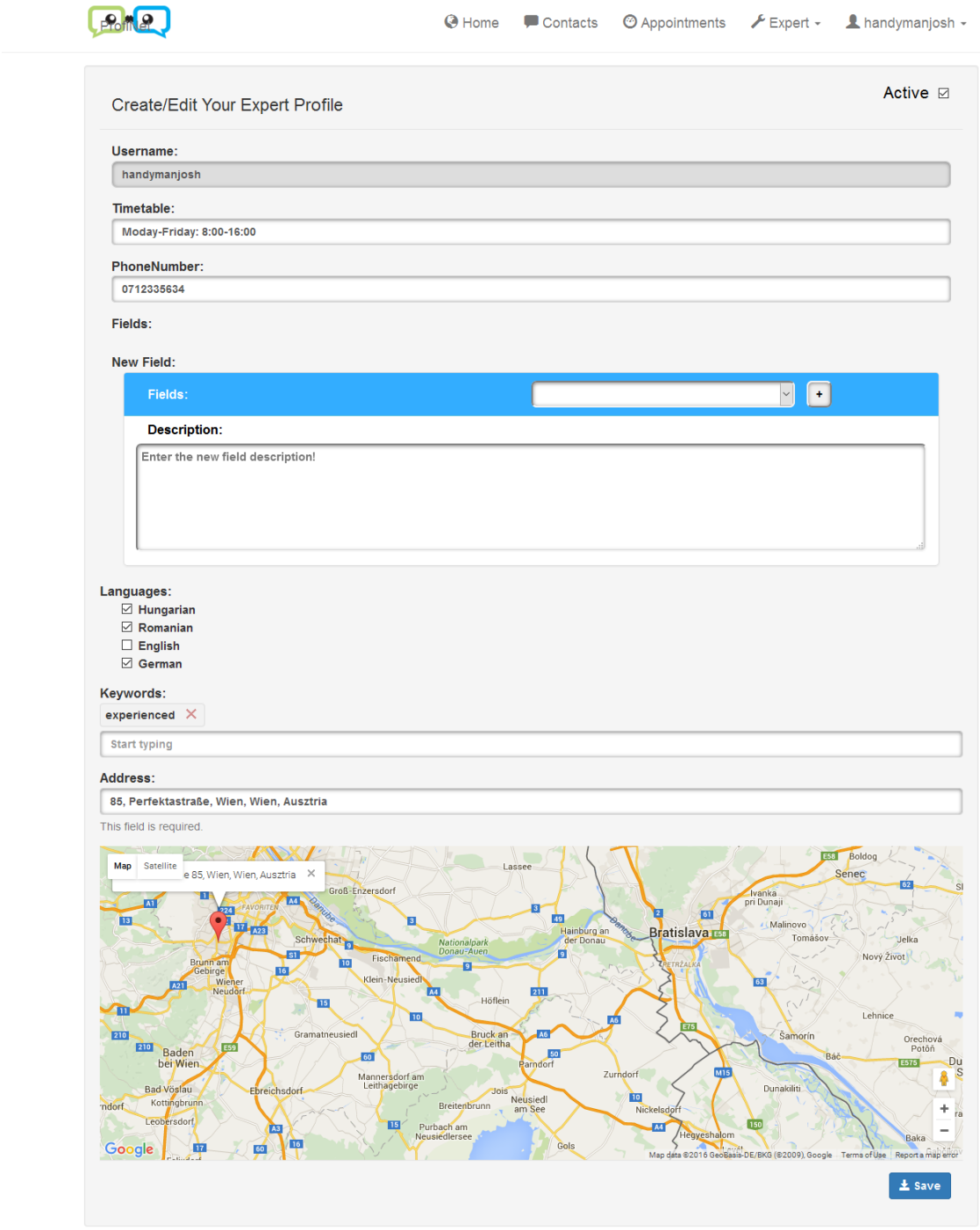


8. ábra: *A felhasználó időpontot kérhet*

Abban az esetben, ha a szolgáltatást nyújtó elfogadta a felkérést, a felhasználónak lehetősége lesz rendszeren belüli üzenetek formájában kommunikálni a szakemberrel, árajánlatot kérni, egyeztetni, hogy mikor és hol lesz szükség a munkálatokra. Ha minden rendben, és mindkét fél úgy ítéli meg, hogy semmi akadálya a munka elvégzésének, a kezdeményező felhasználó időpontot kérhet (8. ábra), és felviheti a rendszerbe a találkozó leírását, kezdeti időpontját és becsült végső időpontját. A szakember ezt elfogadhatja, de akár el is utasíthatja, ha nem talál mindent rendben.

A találkozókat utólag értékelheti a felhasználó, ezzel is pontosítva a főoldalon megjelenő lista elemeinek sorrendjét.

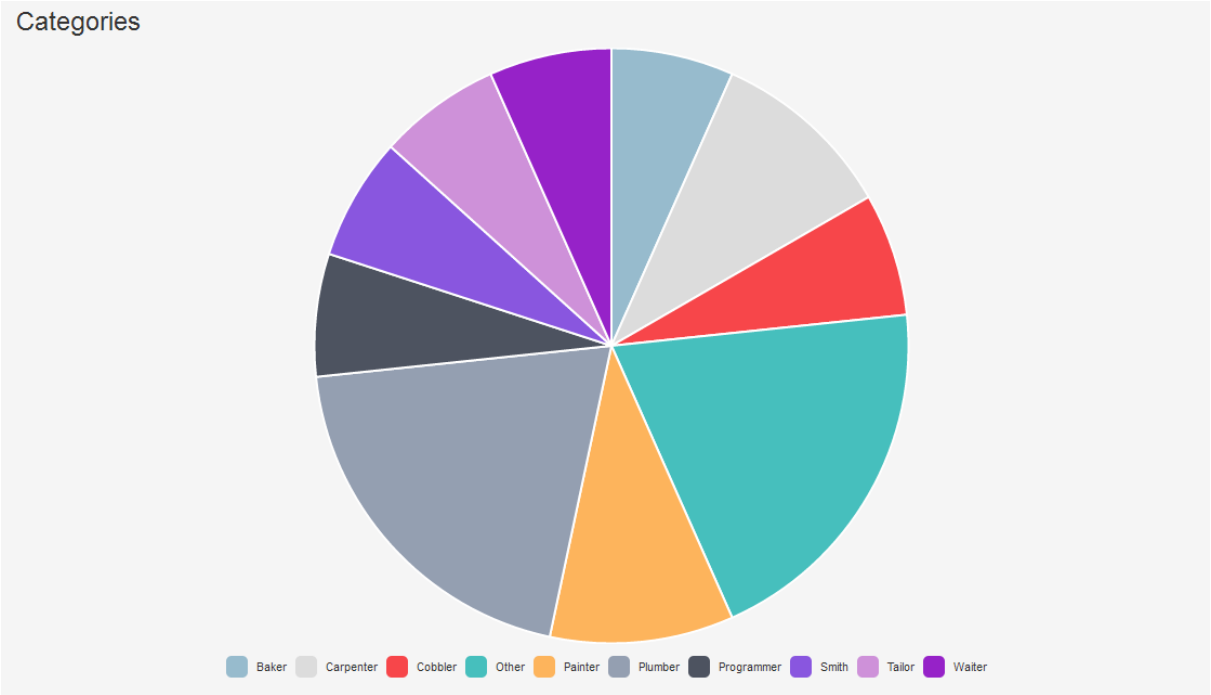
Ha szakemberként szeretnék megjelölni magukat az oldalon, a regisztráció és bejelentkezés után a felhasználók kitölthetnek egy „szakember adatlapot” (9. ábra), ahol felvihetik órarendjüket, telefonszámukat, az általuk beszélt nyelveket, kulcsszavakat, saját címüket és hirdetéseiket. Ugyanezen a profilon belül módosíthatják egy-egy ajánlat láthatóságát, illetve az összesét is, ha aktivitásukat nem kívánják folytatni az oldalon, vagy meghatározatlan ideig fel szeretnék függeszteni azt.



9. ábra: *A szakember adatlapja*

Az oldal felhasználói közé tartozik az adminisztrátor is. Az ő feladatai közé tartozik a különböző entitások (kategórianevek, kulcsszavak, nyelvek, szakemberek, találkozók stb.) felülvizsgálata és kezelése. Emellett követheti a háttérben futó szerveralkalmazás állapotát, a HTTP és REST kérésekről készült statisztikákat, a felhasználók bejelentkezési kísérleteit, a rendszernaplókat. Az adminisztrátor letilthatja a felhasználókat, és megnézheti a rendszer felhasználóinak viselkedéséről készített statisztikákat (példa erre a 10. ábra):

- elérhetőségek igénylése az utóbbi hétben
- időpontkérések az utóbbi hétben
- hirdetések ország szerint
- hirdetések kategória szerint
- találkozók kategória szerint
- új felhasználók dátum szerint
- legkeresettebb kulcsszavak



10. ábra: *Hirdetések szakterületre lebontva*

3. Felhasznált eszközök

A ProfiNet fejlesztése során verziókövető rendszerként a Mercurial [1] volt használva, ami egy nyílt forráskódú, osztott verziókövető szoftver. A build és függőségmenedzsment támogatására Apache Maven-t [2] használ a projekt. A Maven XML-alapú, POM konfigurációs állományokat használ. A POM fájl tartalmazza a szerverre való kitelepítés konfigurációját is. A Maven nagy előnye, hogy a külső függőségeket, ha szükséges, le tudja tölteni hálózaton keresztül.

A folytonos integráció (Continuous Integration - CI) egy fejlesztési módszer, amely hatékonyabbá teszi a csapatban történő fejlesztést. A módszer alkalmazásának támogatására Jenkins-t [3] használt a fejlesztőcsapat. A Jenkins egy nyílt forráskódú, Java nyelvben megírt CI eszköz. A szoftver nagy előnye, hogy kommunikál verziókövető rendszerekkel, a beállításoktól függően automatikusan buildeli és futtatja a programot, és folyamatosan információt szolgáltat a projekt aktuális állapotáról.

A SonarQube [6] egy nyílt forráskódú szoftver a forráskód minőségének ellenőrzésére és követésére. Különböző minőségi kritériumok alapján elemzi a kódot és grafikonokat készít az aktuális állapot szemléltetésére. Nagy előnye, hogy együtt tud működni a CI rendszerekkel.

A Profinet projekt MySQL [7] relációs adatbázis-menedzsment rendszert használ. Ennek konfigurálását a MySQL Workbench tette lehetővé. A projekt fejlesztése Eclipse [4] és Netbeans [5] integrált fejlesztői környezetek segítségével történt.

4. Következtetések és továbbfejlesztési lehetőségek

A ProfiNet projekt keretein belül sikerült egy olyan szoftvert létrehozni, amely szolgáltatáskeresésben nyújt segítséget a kliensek számára, illetve segíti népszerűsíteni a szakemberek ajánlatait. Sikerült egy olyan rendszert kifejleszteni, ami támogatja a többnyelvűséget, és külön hangsúlyt fektet a felhasználók visszajelzéseinek feldolgozására.

A rendszer támogat különböző felhasználói szerepköröket, így az adminisztrátoroknak lehetőségük van a rendszer és a felhasználók menedzselésére. A szakemberek számára is külön felületet biztosít, amely segítségével tudják szerkeszteni a saját profiljukat és az ajánlataikat.

A ProfiNet jelenlegi állapotában egy demó verzió, amely még számos bővítési lehetőséget rejt magában. A fejlesztése során felmerült fontosabb ötletek:

- távolság szerinti szűrés a szakemberek ajánlatai között;
- útvonalterv a szakemberekhez;
- a szoftver elkészítése más platformokra (mobil: Android, iPhone, Windows Phone);
- közösségi média intergrációja (Facebook, Google+, Twitter, Linkedin stb.);
- notifikációk hatékonyabbá tétele;
- autentifikációs mechanizmus átváltása OAuth alapúra;
- Google Analytics integrálása statisztikák készítéséhez;
- kedvenc hirdetések elmentése.

Irodalomjegyzék

[1] Mercurial Hivatalos Weboldal [Online] http://mercurial.selenic.com/ [Utolsó megtekintés dátuma: 2016. április 28.].

[2] Maven Hivatalos Weboldal [Online] http://maven.apache.org/ [Utolsó megtekintés dátuma: 2016. április 28.].

[3] Jenkins Hivatalos Weboldal [Online] http://jenkins-ci.org/ [Utolsó megtekintés dátuma: 2016. április 28.].

[4] Eclipse Hivatalos Weboldal. [Online] https://www.eclipse.org/ [Utolsó megtekintés dátuma: 2016. április 28.].

[5] NetBeans Hivatalos Weboldal. [Online] https://netbeans.org/ [Utolsó megtekintés dátuma: 2016. április 28.].

[6] SonarQube Hivatalos Weboldal. [Online] http://www.sonarqube.org/ [Utolsó megtekintés dátuma: 2016. április 28.].

[7] MySQL Hivatalos Weboldal. [Online] https://www.mysql.com/ [Utolsó megtekintés dátuma: 2016. április 28.].

[8] Spring Hivatalos Dokumentáció. [Online] https://spring.io/docs [Utolsó megtekintés dátuma: 2016. április 28.].

[9] AngularJS Hivatalos Weboldal. [Online] https://www.angularjs.org/ [Utolsó megtekintés dátuma: 2016. április 28.].

[10] Bootstrap Hivatalos Weboldal. [Online] http://getbootstrap.com/ [Utolsó megtekintés dátuma: 2016. április 28.].

[11] Hibernate Hivatalos Weboldal. [Online] http://hibernate.org/ [Utolsó megtekintés dátuma: 2016. április 28.].

[12] Clarence Ho, Rob Harrop, Chris Schaefer, *Pro Spring*, Apress, 2014.

[13] Martin Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, R. Stafford, *Patterns of Enterprise Application Architecture*, Addison-Wesley Professional, 2002.