1. fejezet

Feladat specifikáció

A szakdolgozatban tárgyalt projekt célja egy olyan webalkalmazás, illetve egy ehhez tartozó Android alkalmazás elkészítése, ami az előző szakaszban felsorolt problémákra nyújt korszerű megoldást.

Az elkészült projekt négy részre tagolható: a kliens oldal kéréseit kiszolgáló szerver oldalra, az Android alkalmazásra, a kliens oldal adminisztrációs felületére, illetve a kliens oldal "kliens" felületére, ami a publikus funkciókat biztosítja.

A programok publikus megjelenítésére a webes kliens alkalmazás, illetve az Android alkalmazás ad lehetőséget:

1.1. Az Android alkalmazás

Az Android alkalmazás egy olyan androidos programkereső alkalmazás, amiben a felhasználó megadhatja, hogy milyen kategóriákban és milyen távolságban érdeklik programok, rendezvények. Ezek alapján lehet listázni a programokat, keresni köztük, megnézni egy részletesebb leírást, illetve megnyitni egy térképes és egy naptáras megjelenítést is. Az események elmenthetőek, ekkor Android értesítés érkezik róluk.

1.2. A kliens felület

A kliens felület az Android alkalmazás webes megfelelője. A felhasználó listázhatja a programokat, kereshet helyszín, időpont, programtípus vagy kulcsszó alapján. Adott eseményre kattintva megjelennek a részletes adatok, illetve térképen az esemény helyszíne.

Regisztráció után a felhasználó "megcsillagozhat", elmenthet eseményeket, amiket így később könnyen megtalálhat, illetve email értesítést kaphat róluk.

1.3. Az adminisztrációs felület

Az elkészített alkalmazáshoz tartozik egy adminisztratív felület, ahol az arra jogosult programszervezők létrehozhatnak saját eseményeket, majd szerkeszthetik, törölhetik azokat. A webalkalmazással felvett események bekerülnek az adatbázisba, amit a webes kliens felület és az Android alkalmazás használ. Az adminisztratív felületen két jogosultságot különböz-

tetünk meg, az admint és a szuperadmint.

Az admin bejelentkezés után megtekintheti a korábban létrehozott eseményeit listanézetben. Az egyes eseményeket törölheti vagy megnyithatja a szerkesztés módot, ahol egy űrlapon módosíthatja a program adatait, majd elmentheti a változásokat. Létrehozhat új eseményt is, amit a szerkesztéshez hasonlóan egy űrlap kitöltésével tehet meg. Itt a következő adatok beállítására van lehetőség:

- esemény neve
- helyszín
- rövid összefoglaló leírás
- részletes leírás
- kezdés időpontja
- befejezés időpontja
- belépő
- kategória
- honlap URL
- Facebook URL

A szuperadmin mindent megtehet, amit az admin, azzal a különbséggel, hogy ő az összes admin által létrehozott eseményhez hozzáfér, és módosíthatja, törölheti azokat.

1.4. Use case

A 1.1. ábra összegzi a különböző szerepkörök use case-eit. A be nem jelentkezett felhasználó az anonim felhasználónak felel meg, a regisztráció után belépett egyszerű felhasználó Regisztrált felhasználónak. A hirdetők adminisztrátor szerepkört kapnak, míg az üzemeltető a szuperadminnak felel meg. Megkülönböztetjük még az androidos felhasználót bejelentkezéstől függetlenül.

1.4.1. Az egyes use case-ek részletes leírásai

Use case: események listázása

Leírás: A felhasználó listázhatja az elérhető eseményeket, ekkor az események néhány fontosabb tulajdonsága jelenik meg.

Szereplők: Anonim, Regisztrált, Androidos, Hirdető, Üzemeltető

Megkötések: Csak jövőbeli események jelennek meg.

Use case: események keresése

Leírás: A felhasználó különböző keresési feltételek szerint kereshet az események között, úgy mint dátum intervallum, helyszín, névbeli egyezés, kategória.

Szereplők: Anonim, Regisztrált, Androidos, Hirdető, Üzemeltető

Megkötések: -

Use case: esemény részletes nézetének megnyitása

Leírás: A felhasználó megtekintheti egy adott esemény részletes nézetét, hogy bővebb információhoz jusson az esemény részleteit illetően.

Szereplők: Anonim, Regisztrált, Androidos

Megkötések: -

Use case: térképes nézet megnyitása

Leírás: A felhasználó megnyithatja a térképes nézetet, amin az egyes eseményekhez rendelt markerek jelennek meg, rájuk kattintva az eseményhez tartozó infobox jön elő.

Szereplők: Anonim, Regisztrált, Androidos

Megkötések: -

Use case: naptáras nézet megnyitása

Leírás: A felhasználó megnyithatja a naptáras nézetet, amin az egyes napokra kattintva az aznap történő eseményeket listázza a program.

Szereplők: Androidos

Megkötések: Csak jövőbeli dátumokat lehet kiválasztani.

Use case: események mentése

Leírás: A felhasználó menthet eseményeket, amikről a webes felületen email értesítés, az Android alkalmazásban Android értesítés érkezik az esemény kezdete előtt 5 órával.

Szereplők: Regisztrált, Androidos

Megkötések: -

Use case: feliratkozás hirdetőre

Leírás: A felhasználó feliratkozhat hirdetőkre, ekkor a hirdető által közzétett új eseményekről a webes felületen email értesítés, az Android alkalmazásban Android értesítés érkezik.

Szereplők: Regisztrált, Androidos

Megkötések: -

Use case: feliratkozás helyszínre

Leírás: A felhasználó feliratkozhat helyszínre, ekkor a helyszínen közzétett új eseményekről a webes felületen email értesítés, az Android alkalmazásban Android értesítés érkezik.

Szereplők: Regisztrált, Androidos

Megkötések: -

Use case: adatok módosítása

Leírás: A felhasználó módosíthatja az email címét, jelszavát és az értesítési beállításokat.

Szereplők: Regisztrált, Androidos, Hirdető, Üzemeltető

Megkötések: -

Use case: témaválasztás

Leírás: A felhasználó választhat világos és sötét téma közül, ekkor az alkalmazás színei, az ikonok és stílusok ennek megfelelően módosulnak.

Szereplők: Androidos

Megkötések: -

Use case: regisztráció

Leírás: A felhasználó létrehozhat egy új felhasználói fiókot megfelelő fióknév, email cím és jelszó megadásával.

Szereplők: Anonim

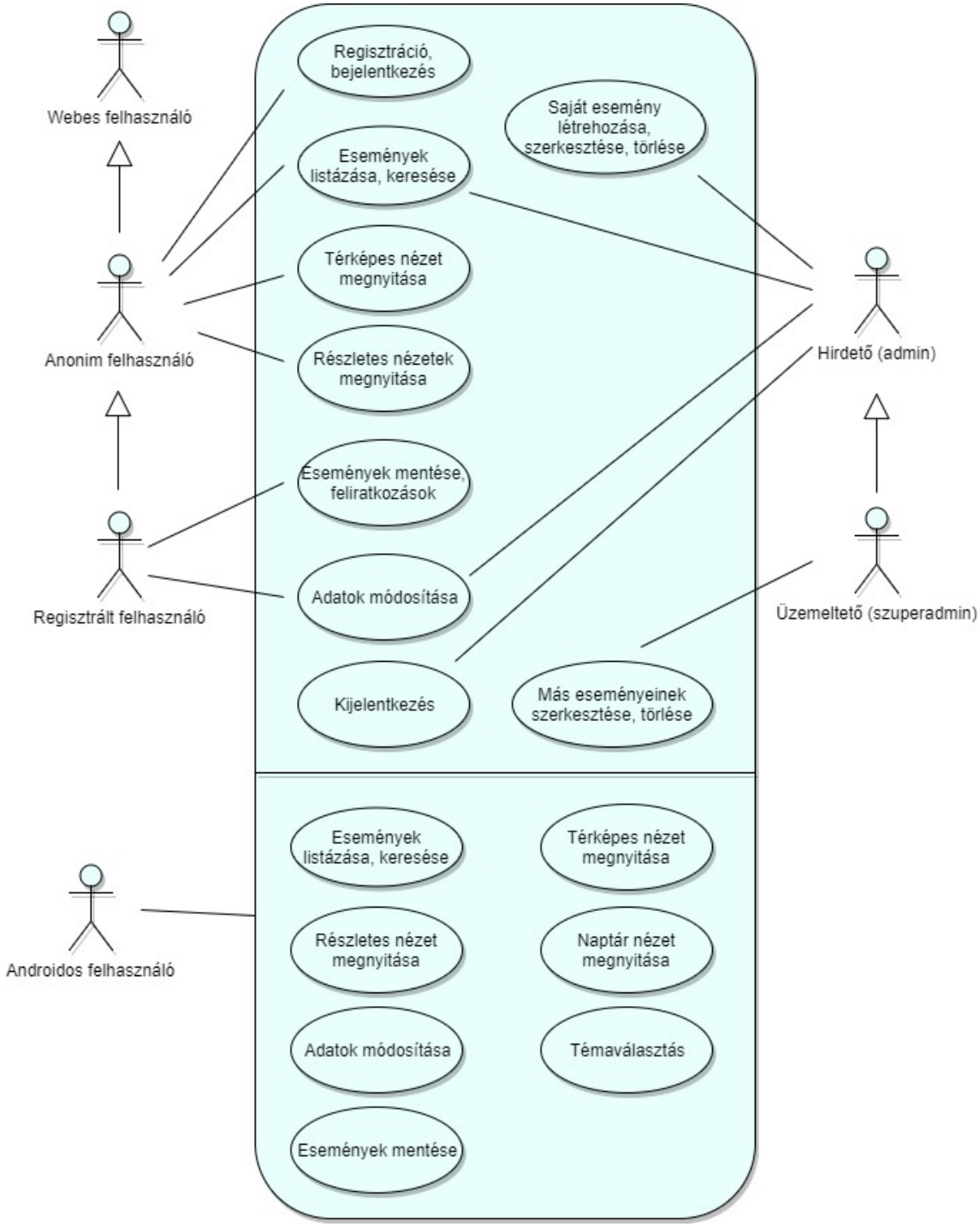
Megkötések: -

Use case: bejelentkezés, kijelentkezés

Leírás: Az Anonim felhasználó bejelentkezhet, a bejelentkezett felhasználó kijelentkezhet az alkalmazásból.

Szereplők: Anonim, Regisztrált, Hirdető, Üzemeltető

Megkötések: -



1.1. ábra. *Az alkalmazás szerepköreinek összevont use case diagramja*

2. fejezet

Választott technológiák

A következőkben felsorolom a projekt elkészítése során használt főbb technológiákat. A kisebb, kevésbé fontos technikákat, könyvtárakat itt nem részletezem, ezek használatára csak a következő fejezetekben térek ki (Tervezés és Megvalósítás fejezetek).

2.1. Spring Boot

Az alkalmazás megvalósításához a Spring Boot keretrendszert választottam. A Spring keretrendszer olyan működést támogat, amelyhez nem szükséges külön konfigurációt vagy programkódot írnunk, ha az elterjedt konvenciókat követjük. A szoftverünk csak azon részeihez szükséges konfigurációt vagy extra kódsorokat létrehoznunk, amelyek eltérnek a praktikusan megválasztott alapértelmezett működéstől.

A Spring Boot ezeket az alapelveket emeli még magasabb szintre. Tovább egyszerűsíti az alkalmazás konfigurálását és a komponensek közti integrációt. A megírt kód alapján feltérképezi a programozó szándékait, és automatikusan konfigurálja a kapcsolódó háttérszolgáltatásokat. Az így hozzáadott funkcionalitás programozói erőfeszítés nélkül jön létre, így jelentősen leegyszerűsödik a fejlesztés folyamata.

2.2. Maven

Egy parancssori build automatizáló eszköz, amely képes a függőségek (.jar-ok) letöltésére tranzitív módon. Előtérbe helyezi a teszteket (Unit és integrációs tesztek is támogatottak). Erősen testre szabható, de csak akkor van rá szükség, ha eltérünk a default-októl (pl. könyvtárstruktúra).

2.3. JPA

Egy szabványos ORM API, aminek az alkalmazás relációs adatainak kezelése a fő feladata. Csak interfészeket specifikál, ezzel téve lehetővé a több lehetséges implementációt.

Egy JPA entitás egy olyan osztály, melynek példányait a JPA relációs adatbázisban perzisztensen tárolja. Az O?R leképezés annotációkkal történik.

Az entitásokat az EntityManager interfészen keresztül tudjuk kezelni, ezen keresztül

érhetjük el a perzisztenciakontextust, ami a memóriabeli entitások és az adatbázis közti kapcsolatot jelenti.

2.4. Spring Data

Külön modul az adatelérés támogatására, amivel az adatelérési kód nagy része megspórolható, hiszen egyszerűen tudunk saját entitásra specifikus repository-t írni, ami a JpaRepository<T, ID extends Serializable> interfész leszármazottja.

Az interfészbe felvett findBy... metódusok nevei alapján egyéb lekérdezések generálhatók.

2.5. Spring MVC

//TODO

2.6. REST

A szabványos HTTP kérés?válaszokra építve, az elérendő erőforrásokat külön-külön egyedi URI-hoz rendelve építjük fel szolgáltatásunkat. Az URI-k egységes interfészt biztosítanak a kliens számára. Minden kérésre azonos formátumban reagál a szerver, ez általában JSON, HTML vagy XML.

2.7. Thymeleaf

A Thymeleaf egy olyan Java alapú template engine, ami XML/HTML/HTML5 template fájlok kezelését biztosítja rugalmasan bővíthető dialektusokkal. A th: névtérbe tett attribútumokkal bővítve egyszerűen designolható, a template fájl szerkesztéséhez nincs szükség szerver futtatására.

2.8. Angular 4

//TODO