A dolgozat formai követelményei

Ajánlott oldalszám:

- BSc szakdolgozat: 20-25 oldal,
- MSc diplomamunka: 35-50 oldal
- Tanárszakos szakdolgozat: 50000-80000 karakter szóközökkel (lásd. TKK szabályzat)

Másfeles sorköz, sorkizárt, 12-es betű méret (általában Computer Modern), margók: jobb, bal, lent és fent 2,5 cm (ahogy ez a template van beállítva, az jó).

Kötelező elemei a dolgozatnak

- Címlap
- Tartalmi összefoglaló (kivonat)
- Tartalomjegyzék
- Érdemi rész (Szakterület specifikus fejezeteket tartalmaz, kérjük, konzultáljon a témavezetővel, hogy a példában közzétett kísérletes tudományterületeken használt fejezetekből az Ön dolgozatában melyikre van szükség!)
- Irodalomjegyzék
- Nyilatkozat

Szegedi Tudományegyetem
Természettudományi és Informatikai Kar
Informatikai Intézet, Szoftverfejlesztés Tanszék

SZAKDOLGOZAT

Eseményszervező webalkalmazás konfigurációs lehetőségekkel

Configurable event organizing web application

Készítette:
Koncz Máté
Programtervező Informatikus BSc hallgató

Témavezető:
Dr. Pengő Edit
egyetemi docens

Szeged
2025

Kivonat

A dolgozat tartalmának rövid (max. 1 oldal) összefoglalása. A következő részekből áll: rövid irodalmi összefoglaló, a dolgozat elkészítéséhez használt módszerek, eredmények, konklúzió

Kulcsszavak: a dolgozat tartalmára specifikusan jellemző 4-6 szó, egymástól vesszővel elválasztva

Tartalomjegyzék

1. fejezet

Bevezetés

1.1. Motiváció

Az alkalmazás ötlete még gimnazista koromban született meg. Az osztálytársaimmal gyakran jártunk össze focizni, ám ezeket az alkalmakat mindig komoly kihívás volt megszervezni. Az legtöbb alkalom azért maradt el, mert nem sikerült olyan időpontot találni, amelyre legalább 8 ember biztosan el tudott jönni. A szavazások nehezen folytak le, rendszerint a messenger-en küldött 'foci péntek délután?', 'foci szombat délelőtt?' stb. üzenetekre adott reakciók száma alapján dőlt el a focizás időpontja. Az alkalmazásom az ehhez hasonló események szervezését könnyíti meg, ahol a legnagyobb kihívás az ideális időpont megtalálása, hiszen a legfontosabb, hogy az minél több emnbernek feleljen meg.

2. fejezet

Fő funkciók bemutatása

2.1. A felhasználók kezelése

Az alkalmazásban a felhasználók kezelése nem a legfontosabb feladat, számos, hasonló alkalmazásban gyakori funkciókat, mint a felhasználói profil szerkesztése, vagy a profil törlése, nem valósítottam meg. A felhasználók autentikációja és autorizációja azonban fontos feladat.

2.1.1. regisztráció

A felhasználó regisztrálhat egy egyedi felhasználónévvel, egyedi email-címmel, és a legalább 8 karakter hosszú jelszavának kétszer történő megadásával. Amennyiben a megadott névvel vagy email-címmel már foglalt, a rendszer értesíti erről a felhasználót.

2.1.2. bejelentkezés, kijelentkezés

A helyes felhasználónév és jelszó megadásával a létrejön egy session, ami 4 órán keresztül biztosít hozzáférést az alkalmazáshoz a felhasználó számára. A session lejárta után az alkalmazás automatikusan kijelentkezteti a felhasználót.

2.1.3. autorizáció

Az események adataihoz csak a szervező és a meghívottak férhetnek hozzá. Az eseményeket és az ezekhez tartozó naptárakat csak a szervezők szerkeszthetik.

2.2. Események kezelése

2.2.1. létrehozás

Az eseményeket a nevük és egy rövid leírás megadásával lehet létrehozni. Naptár hozzáadása opcionális, ez később is megadható, a részleteiről részletesebben lejjeb írtam. Az esemény létrehozásakor a szervező automatikusan a meghívottak közé is bekerül.

2.2.2. csatlakozás

A szervező és a meghívottak számára látható az esemény meghívó-kódja. Ők ezt a kódot továbbíthatják másoknak, akik ezt a kódot használva a meghívottak közé kerülnek.

2.2.3. extra mezők

A szervező extra mezőket adhat az eseményhez. Ezekhez egy cím és a mező lehetséges értékei tartoznak. A mező aktuális értékét beállíthatja a szervező, vagy szavazásra is bocsájthatja azt. Emellett a szervező azt is beállíthatja, hogy a meghívottak vehetnek-e fel új lehetséges értékeket az adott mezőhöz.

2.2.4. eseménysémák használata

Egy esemény résztvevője jogosult arra, hogy az eseményből eseménysémát készítsen. A séma tartalmazza az eseményhez rendelt extra mezőket azok tulajdonságaival és lehetséges értékeivel együtt, és alkalmas arra, hogy a használatával új eseményt lehessen létrehozni, amely extra mezői ugyanezekkel a tulajdonságokkal rendelkeznek.

2.2.5. esemény adatainak véglegesítése

Egy esemény szervezője véglegesítheti egy esemény adatait (ebbe az esemény időpontja és az extra mezők értékei tartoznak bele). Ezután az esemény adatai nem módosíthatóak, a meghívottak pedig értesítést kapnak az esemény véglegesítéséről. A művelet visszavonható, ekkor újra módosíthatóvá válnak az adatok, a meghívottak pedig erről is értesítést kapnak.

2.3. Naptárak kezelése

A naptárfelület és a hozzá tartozó funkciók az alkalmazás legfontosabb részét alkotják, hiszen ennek a feladata, hogy segíse az esemény szervezőjét az ideális időpont kiválasztásában. (((ábra a naptárfelületről)))

2.3.1. létrehozás

Naptár hozzáadása az időintervallum (legfeljebb 60 nap) és az időzóna megadásával történik. Az intervallum kezdő időpontja nem lehet a múltban, és az intervallum kezdete és vége nem lehet ugyanaz az időpont.

2.3.2. szerkesztés

Az esemény szervezője szűkítheti a választható időpontok körét. Az órákat és napokat letilthatja egyenként és naponta, illetve hetente ismétlődő jelleggel is. A szervező ezeket a műveleteket bármikor visszavonhatja.

2.3.3. vélemény megadása

Az események résztvevői (ebbe az esemény szervezője is beletartozik) minden elérhető időpontról véleményt formálhatnak. Egy időpontról megadható, hogy megfelelő, elfogadható, vagy nem megfelelő. Vélemény megadásához órákat vagy egész napokat lehet kijelölni. A vélemény rögzítése visszavonható művelet.

2.3.4. statisztikák megtekintése

Az események résztvevői minden időpont esetében megtekinthetik, hogy azokhoz mely felhasználók milyen véleményt rögzítettek. Az esemény szervezője átfogóbb statisztikákat is lekérhet. (((ide ábrák kellenek)))

2.3.5. időzónák kezelése

Az időzóna megadása azért fontos, mert a kiválasztható időpontok egész órák, amikből az óraátállítások esetén nem 24 van. A óraátállítások pedig különbözhetnek időzónánként. Fontos, hogy a felhasználók ne tudjanak olyan órákat kijelölni, amik nem léteznek, hiszen ez megtévesztő, és az esetleges plusz órát is jelölni kell.

3. fejezet

Felhasznált technológiák

3.1. Spring Boot

A backend alkalmazásomat a Spring Boot használatával készítettem el. Ennek segítségével gyorsabban és egyszerűbben tudom elkezdeni a Spring Frameworkre épülő alkalmazásom fejlesztését, mivel a Spring Boot projektek rendelkeznek alapértelmezett konfigurációs beállításokkal, nekem ezeket csak akkor kell átírnom, ha valahol az alapértelmezettől eltérő viselkedést akarok elérni. A fejlesztést InteliJ IDEA (Community Edition) környezetben végeztem.

https://spring.io/projects/spring-boot

3.1.1. Jakarta Persistence API Hibernate ORM-el

A Hibernate segítségével végeztem objektum-relációs leképezéseket a backend alkalmazásomban, ezt azonban a Jakarta Persistence API-n keresztül használtam. A JPA-val emellett saját nativ query-ket is írtam, ahol az adatokat egyéni módon akartam kezelni.

https://jakarta.ee/specifications/persistence/

https://hibernate.org/orm/

3.1.2. H2 In-memory DB

A H2 memórián belüli adatbázis megvalósítását az integrációs tesztjeimhez használtam. Mivel ez az adatbázis minden futtatás esetében újra inicializálódik, fejlesztés közben egyszerűbb a tesztjeimet futtatnom, hiszen nem kell minden, a leképezendő objektumok kapcsolatait érintő változás esetén a megfelelő strukturális változtatásokat véghezvinnem az adatbázisban. Emellett általánosságban egyszerűbben kezelhető, hiszen nincs szükségem adatbázisszerverre.

https://www.h2database.com/html/main.html

3.2. MariaDB

A backend alkalmazás éles futtatása során egy MariaDB adatbázisszerverre kapcsolódik. A MariaDB egy nyílt forráskódú, SQL alapú relációs adatbázis-kezelő rendszer (RDBMS). Más hasonló adatbázisszerverek közül azért erre esett a választásom, mert a tanulmányaim során ezt használtam a legtöbbet.

https://mariadb.org/

3.3. Angular

Az Angular egy Google által fejlesztett, nyílt forráskódú, TypeScript alapú keretrendszer webalkalmazások fejlesztésére. Az applikációm frontend részét ezzel valósítottam meg. A választásom azért erre esett, mert a hozzá hasonló keretrenszerek közül ezt ismerem a legjobban, és úgy ítéltem meg, hogy alkalmas modern, reszponzív felhasználói felületek létrehozására.

https://angular.dev/overview

3.3.1. Angular Material

A frontenden nagy számban használtam az Angular Material könyvtár komponenseit. Az Angular Material a Google hivatalos UI komponenskönyvtára, számos termékükben használják.

https://material.angular.io/

3.3.2. RxJs

Az RxJs könyvtár aszinkron műveletek kezelését segítő függvényeket és típusokat tartalmaz. A frontendemen könyvtárban található Observable típust használtam az aszinkron API lekérdezések megvalósításához, valamint a Subject típust használtam azokban az esetekben, amikor gyerek komponensben kellett szülő komponensben bekövetkező eseményre reagálni.

https://rxjs.dev/

3.4. git

A git verziókövető rendszert használtam a teljes fejlesztési folyamat során. Hogy a konzulensem is követhesse a projekt alakulását, létrehoztam egy nyilvános Github repository-t, és oda sűrűn pusholtam a local repository commit-jait.

https://git-scm.com/
https://github.com/about

4. fejezet

Az API fejlesztése

4.1. A model réteg

4.2. A service réteg

4.3. Controller réteg

4.4. Kivételkezelés

4.5. Tesztelés

@Test public void shouldSelectOptionByMostVotes() throws Exception{ User owner = new User("felh","asznalo1","email@gmail.com"); User savedOwner = userService.saveUser(owner); User user = new User("felh2","asznalo1","email2@gmail.com"); User savedUser = userService.saveUser(user);

Event event = new Event(savedOwner,"kertiparti", new HashSet<>(),"");

SelectionField selectionField = new SelectionField("mezo",false,true, Set.of(new Option("ertek"),new Option("ertek2"))); event.setSelectionFields(Set.of(selectionField));

Event savedEvent = eventService.saveEvent(event);

long fieldid = eventService.getEventById(savedEvent.getId()).getSelectionFields().stream().findFirst
long firstOptionId = eventService.getSelectionFieldById(fieldid).getOptions().stream().filter((o)->o.getValue().equals("ertek")).findFirst().get().getId(); long secondOptionId = eventService.getSelectio
>o.getValue().equals("ertek2")).findFirst().get().getId();

eventService.addVote(firstOptionId,fieldid,user);

Assertions.assertTrue(eventService.getOptionById(firstOptionId).isSelected()); Assertions.assertFalse(eventService.getOptionById(secondOptionId).isSelected());

eventService.addVote(secondOptionId,fieldid,user); eventService.addVote(secondOptionId,fieldid,ow

Assertions.assertFalse(eventService.getOptionById(firstOptionId).isSelected()); Assertions.assertTrue(eventService.getOptionById(secondOptionId).isSelected()); }

4.6. Védelem

4.6.1. CORS

4.6.2. Json Web Token

5. fejezet

A frontend fejlesztése

5.1. Service réteg

5.2. Fő komponensek

5.2.1. Dialógusablak

5.2.2. Naptárfelület

5.3. Routing

5.3.1. Auth Guard-ok

5.4. UX

5.4.1. Canvas Angularban

5.5. Aszinkron programozási megoldások

6. fejezet

Továbbfejlesztési lehetőségek

7. fejezet

Összefoglalás az eredményekről

7.1. Táblázatok, képletek, egyéb értéssegítők

Tartalmazhat táblázatokat, melyekre hivatkozni kell a szövegben, például 7.1. táblázat. A táblázatot középre kell rendezni és a szövegben lévő hivatkozás közelében kell elhelyezni.

7.1. táblázat. A táblázat címe a táblázat fölé kerül.

Col1	Col2	Col2	Col3
4	545	18744	7560
5	88	788	6344

Tartalmazhat ábrákat. Ezeket középre kell rendezni és hivatkozni kell rájuk a szövegben (lásd **??**). Ha az Irodalmi áttekintés fejezet tartalmazott ábrá(ka)t, akkor a számozás ebben a fejezetben nem újra kezdődik, hanem folytatódik.

Egész oldalas ábrák mellékletben legyenek elhelyezve!

Használjuk bátran a LaTeXbrilliáns egyenletszerkesztő rendszerét!

A szövegközi x^2 képletetket nem, de a kiemelt formulákat általában számozzuk:

$$\mathrm{Var}(A(K_n^r))) \ll n^{-2}, \tag{7.1}$$

kivéve a hosszabb számolásokat

$$E(F_n(x_{n+1})^2) = \frac{1}{A(K)^{n+1}} \int_K \int_{K^n} \left(\sum_I \mathbf{I}(F_I \in \mathcal{F}_n(x_{n+1})) \right)^2 dX_n dx_{n+1}$$

$$= \frac{1}{A(K)^{n+1}} \int_K \int_{K^n} \left(\sum_I \mathbb{I}(F_I \in \mathcal{F}_n(x_{n+1})) \right)$$

$$\times \left(\sum_J \mathbf{I}(F_J \in \mathcal{F}_n(x_{n+1})) \right) dX_n dx_{n+1}$$

$$\leq \frac{1}{A(K)^{n+1}} \sum_I \sum_J \int_K \int_{K^n} \mathbf{I}(F_I \in \mathcal{F}_n(x_{n+1})) \mathbf{I}(F_J \in \mathcal{F}_n(x_{n+1}))$$

$$\times \mathbf{I}(d_H(K, K_n) \leq \varepsilon_K) dX_n dx_{n+1} + O((1 - c_K)^n) \tag{7.2}$$

A képletben használt mennyiségek legyenek megadva az első használatuknál.

8. fejezet

Összefoglalás

(új oldalon kezdve)

A dolgozat eredményeinek összefoglalása, következtetések levonása.

Az összefoglalásban egyértelműen jelezve legyen a hallgató saját szerepe/eredményei.

Irodalomjegyzék

[1] A. L. Edmonds, *The geometry of an equifacetal simplex*, Indiana University Reprint (2003).

[2] N. Altshiller-Court, *Modern Pure Solid Geometry*, Chelsea Publishing Co., N. Y., 1964.

[3] J. Gipsz, M. Kalányos, and Z. Kovács, *The Chasm*, Doom II. **76** (1963), 661–669.

[4] J. Monroe and M. O'Donnel, *The kiss*, Amer. Math. Monthly **76** (1969), 661–663.

Köszönetnyilvánítás

(nem kötelező elem), (új oldalon kezdve)

Ebben a fejezetben lehet köszönetet mondani mindazoknak, akik segítették a dolgozat elkészülését. Itt lehet megemlíteni továbbá a munkát támogató pályázatokat, ösztöndíjakat, stb.

Nyilatkozat

A szöveg kötött, kérjük ezt használni!

Alulírott, Végzős Edömér, xxxx szakos hallgató, kijelentem, hogy a szakdolgozatban ismertetettek saját munkám eredményei, és minden felhasznált, nem saját munkából származó eredmény esetén hivatkozással jelöltem annak forrását.

Szeged, 2025. március 5.

_____

Végzős Edömér

Mellékletek

(nem kötelező elem, a dolgozat oldalszámaiba nem tartozik bele.), (új oldalon kezdve) Ebben a fejezetben lehet elhelyezni a nagyobb táblázatokat, ábrákat, adathalmazokat.