

Java alkalmazások

Java Spring Boot beadandó – 2024. 12.01 – Hajdu Gergő, KMZ7M1

Rezümé

Az alkalmazás egy egyszerű, főként diákoknak szánt munkaközvetítő oldalt valósít meg. A designokhoz Bootstrap került felhasználásra. Az egész weboldal MVC tervezési struktúrát alkalmaz.

Tartalomjegyzék

Rezümé.....	1
Feladatok.....	2
Feladatok megvalósítása	3
Választott adatbázis	3
1. Feladat	4
2. Feladat	5
3. feladat.....	8
4. Feladat	9
5. feladat.....	11
6. feladat.....	13
7. feladat.....	14
GitHub repository	16
Futtatható JAR fájl.....	16

Feladatok

1. Az első oldalon mutassa be a céget egy látványos weboldalon **(2 pont)**
2. Legyen Regisztráció, Bejelentkezési lehetőség
 - A „Belépés” menüpont akkor látható, ha nincs bejelentkezve a felhasználó.
 - A „Kilépés” menüpont akkor látható, ha be van jelentkezve a felhasználó.
 - A rendszer fejlécen jelenítse meg a bejelentkezett felhasználót, ha be van lépve
3. Legalább 3 felhasználói szerepet különböztessen meg: **(3 pont)**
Admin, User, Látogató
A menüpontok megjelenése és az oldalak elérhetősége változik attól függően, hogy melyik felhasználó használja az oldalt. (pl. admin oldal)
4. Legyen egy oldal, ahol a választott adatbázisból jelenít meg adatokat **(2 pont)**
Ehhez 3 tábla adatait használja fel az adatbázisból.
5. Az egyik oldalon legyen egy kapcsolat űrlap, amelynek segítségével üzenetet **(2 pont)**
lehet küldeni az oldal tulajdonosa számára. Ellenőrizze megfelelően az űrlap helyes kitöltését szerver oldali validációval is. Az elküldött Űrlap adatokat mentse le az adatbázisba.
6. Tegye lehetővé megtekinteni egy hatodik oldalon táblázatban az előző pontban **(2 pont)**
elküldött üzeneteket az adatbázisból fordított időrend szerint (a legfrissebb legyen elől). Írja minden üzenethez a küldés idejét és az üzenetküldő nevét. Ha nem bejelentkezett felhasználó írta, akkor: "Vendég".
7. Valósítson meg az alkalmazásban egy RESTful API-t. **(3 pont)**
Tesztelje az API funkcióit cURL-el és Postman-al is. Mindkét tesztről tegyen képernyőképeket a dokumentációba
8. Használják a GitHub (github.com) verziókövető rendszert. **(2 pont)**
(Kötelező elem! A forrás ez alapján lesz javítva)
9. A GitHub-on a projektmunka módszert alkalmazzák: látszódjék, hogy a **(2 pont)**
csoport tagjai melyik részt készítették el és kb. fele-fele arányban járuljanak hozzá a projekthez. Ne csak a kész alkalmazást töltsék fel egy lépésben, hanem a részállapotokat is még legalább 5 lépésben személyenként.

Feladatok megvalósítása

A beadandó egyedül készült el ezért GitHubon 2 account felhasználásával készültek a commitok.

A könyvtárban található fájlok segítségével lehet tesztelni az applikációt.

Telepítéshez az „adatok.sql” fájlt importálni kell MySQL szerverre, valamint a „futtat.jar” indításával elindul a Tomcat szerver. Ezután a weboldal elérhető a localhost:8080 címen érhető el.

Teszteléshez szükséges felhasználói adatok:

- Admin felhasználó
 - email: admin@gmail.com
 - jelszó: jelszo1
- Normál felhasználó
 - email: user@gmail.com
 - jelszó: jelszo2



Választott adatbázis

A beadandóhoz választott adatbázis a „2-15-Diákmunka-3 táblás” elnevezésű mappában szereplő adatokból készült adatbázis. A megadott táblák adataikkal együtt bekerültek a „SprintUp” adatbázisba, amelybe ezen kívül a felhasználói adatokkal kapcsolatos, valamint a kapcsolatok panelen keresztül küldött üzenetek tartoznak még.

Táblák:

diak (*diakaz, nev, szulido*)

<i>diakaz</i>	A diák azonosítója (számláló), ez a kulcs
<i>nev</i>	A diák neve (szöveg)
<i>szulido</i>	A diák születési ideje (dátum)

munkaado (*mhelyid, nev, telepules*)

<i>mhelyid</i>	A munkahely azonosítója (számláló), ez a kulcs
<i>nev</i>	A munkahely neve (szöveg)
<i>telepules</i>	A munkahely települése (szöveg)

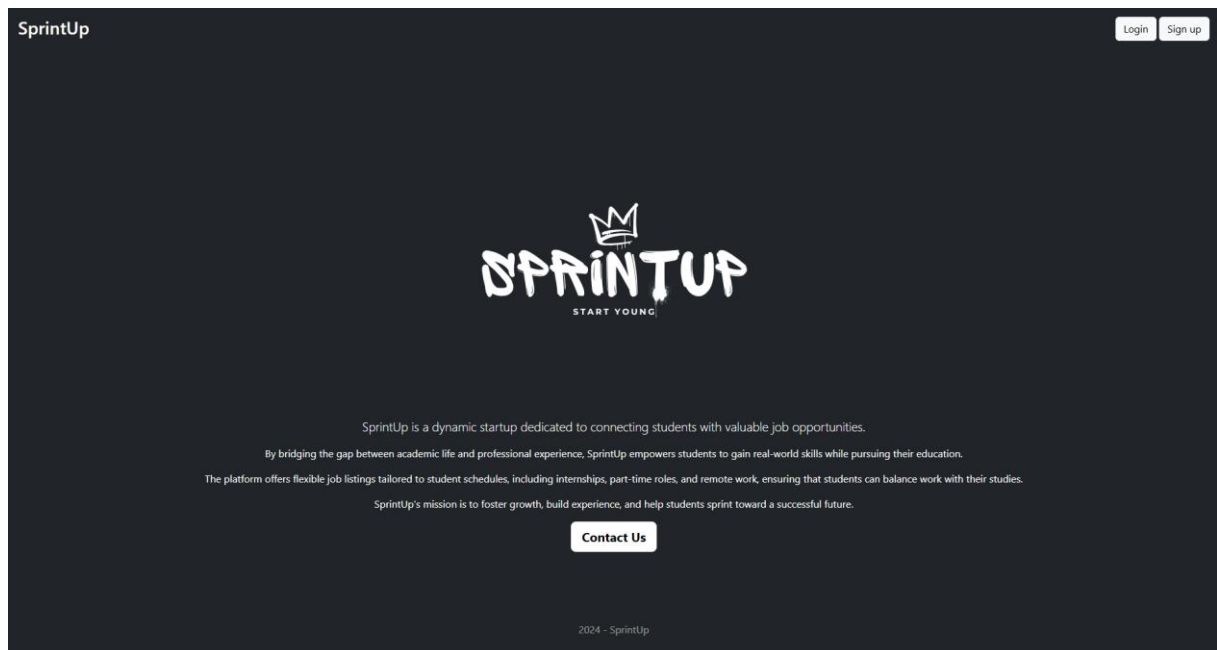
munka (*munkaid, mhelyid, diakaz, allas, datum, oradij, oraszam, kozepiskolas*)

<i>munkaid</i>	A munka azonosítója (számláló), ez a kulcs
<i>mhelyid</i>	A munkaadó azonosítója (szám)
<i>diakaz</i>	A munkát végző diák azonosítója; csak akkor kitöltött, ha a munkára jelentkezett valaki, és el is végezte (szám)
<i>allas</i>	Az állás megnevezése (szöveg)
<i>datum</i>	A munkavégzés dátuma (dátum)
<i>oradij</i>	A munka óradíja (szám)
<i>oraszam</i>	A munkavégzés időtartama órákban (szám)
<i>kozepiskolas</i>	A munkavégző lehet-e középiskolás (logikai)

1. Feladat

Feladat: Az első oldalon mutassa be a céget egy látványos weboldalon

Ennek megvalósításához egy fiktív cég, a „SprintUp” került megtervezésre. Ez egy munkaerőközvetítő cégnek felel meg, amely főként diákok szolgál segítségül. Ehhez egy modern, letisztult és minimalista weboldal készült el. Graffitire hajazó logó miatt kifejezetten népszerű lehetne fiatalabb munkakeresők körében. Tervezés során teljes mértékig kihasználja a bootstrap nyújtotta látványos lehetőségeket. Többi oldal esetén kifejezett figyelem volt fordítva a hasonló kinézetre és megfelelő színek használatára, hogy azok az adott oldal funkcióit reprezentálja.



Felső részen jobbra a vezérlésért felelős menügombok találhatóak míg balra az oldal neve. Oldal közepe tájékán az 5. feladat megvalósításáért felelős oldalra navigáló gomb található.

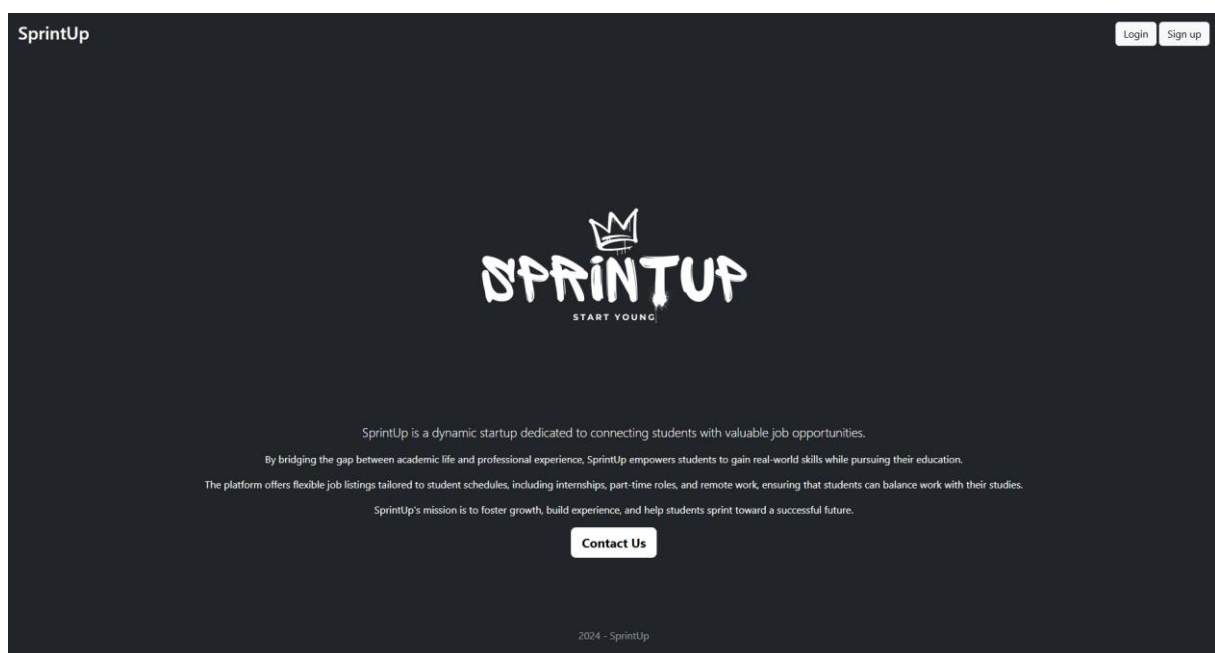
2. Feladat

Feladat:

1. Legyen Regisztráció, Bejelentkezési lehetőség

- A „Belépés” menüpont akkor látható, ha nincs bejelentkezve a felhasználó.
- A „Kilépés” menüpont akkor látható, ha be van jelentkezve a felhasználó.
- A rendszer fejlécen jelenítse meg a bejelentkezett felhasználót, ha be van lépve

A felső részen látható gombok valósítják ezt meg. Az alap esetben mikor a felhasználó bejelentkezés előtt megnyitja az alábbi verziót látja:

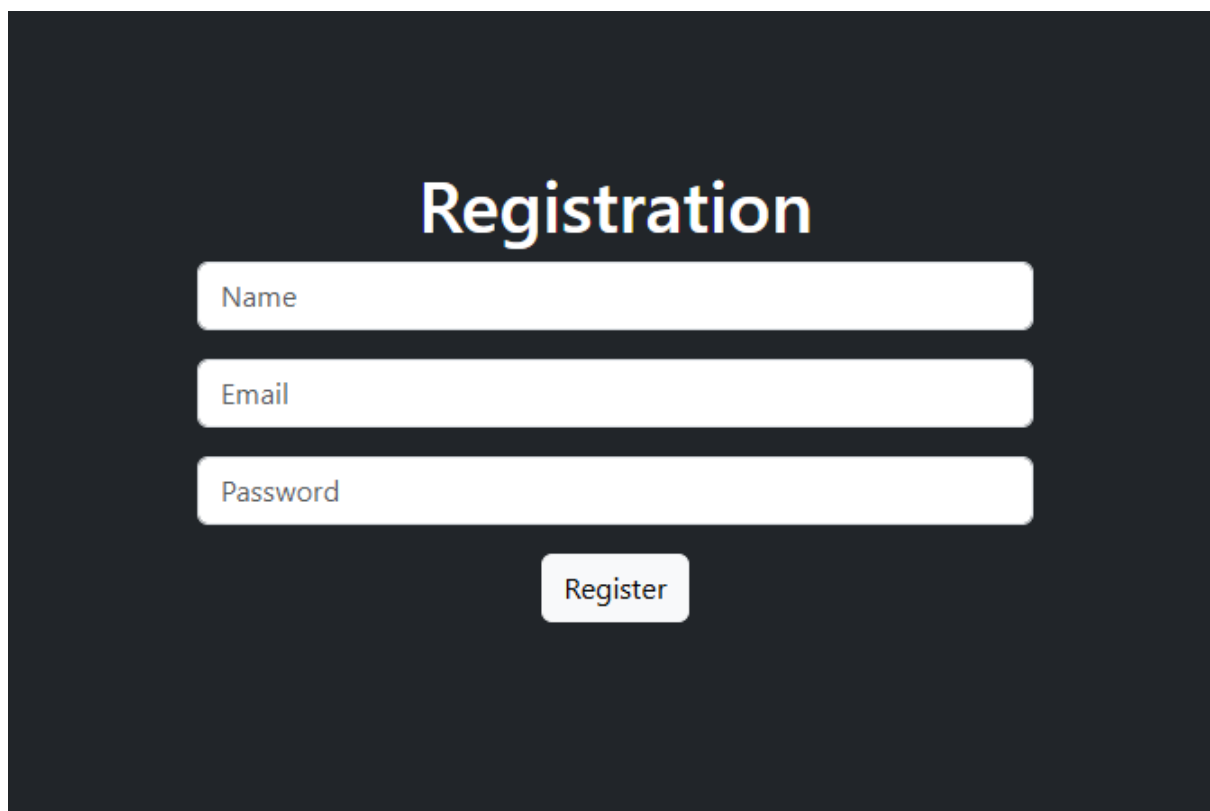


Belépés vagyis a „Login” menü megnyitása után az alábbi elemek jelennek meg a weboldalon:

The image shows a login form with a light gray background. At the top, it says 'Please sign in' in a large, bold, dark font. Below this are two input fields: the first is labeled 'Username' and the second is labeled 'Password'. Both fields have a light blue border. Below the password field is a large blue button with the text 'Sign in' in white.

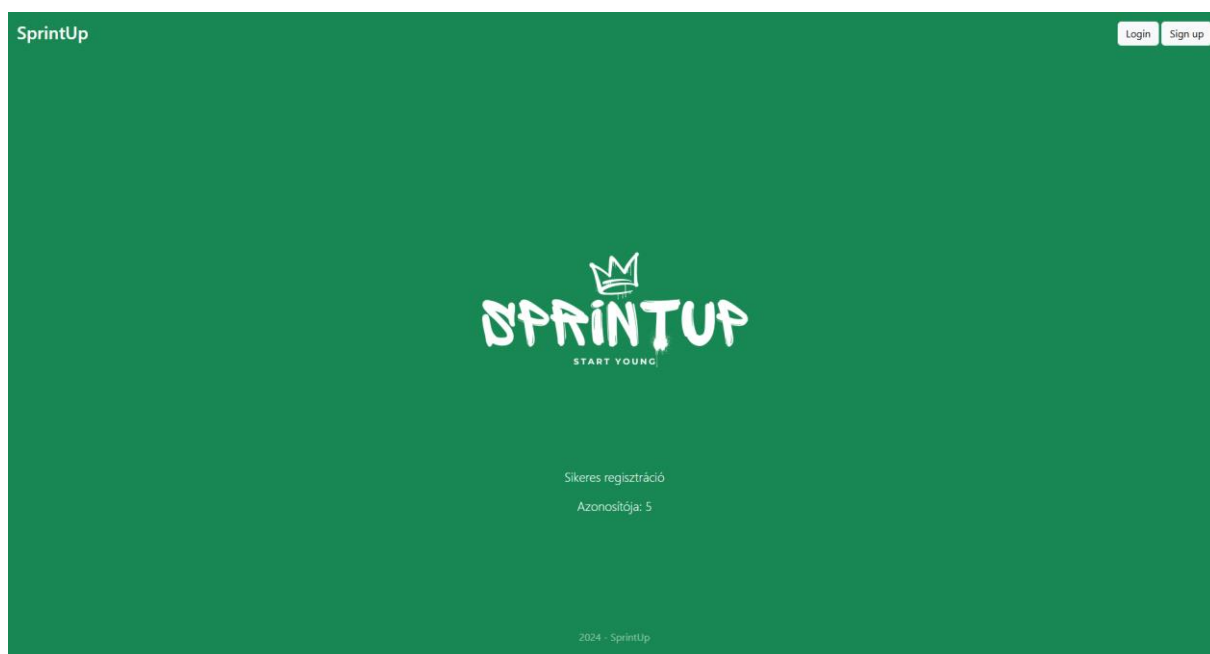
Itt figyelni az alkalmazás, hogy valóban email címet ad-e meg a felhasználó.

Regisztráció vagyis „Sign up” esetén az alábbi jelenik meg:

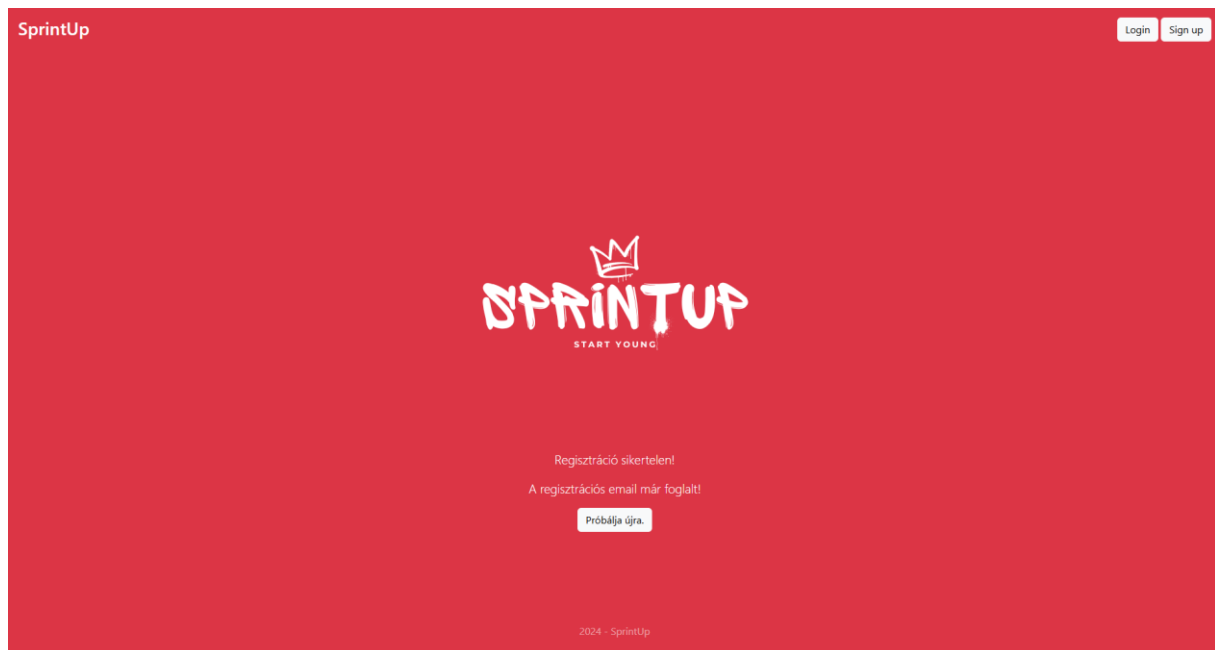


The image shows a registration form on a dark background. At the top, the word "Registration" is written in a large, white, sans-serif font. Below it, there are three white input fields stacked vertically. The first field is labeled "Name", the second "Email", and the third "Password". Below the input fields, there is a white button with the text "Register" in a dark font.

Az adatok után a felhasználó tájékoztatást kap az esetleges hibás regisztrációról is a következő 2 módon:

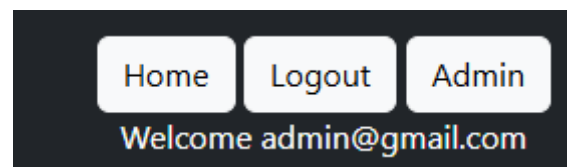


Sikeres regisztráció.



Sikertelen regisztráció.

Ha van bejelentkezett felhasználó az oldal a következő módon jeleníti meg a felhasználó adatát (ebben az esetben egy admin felhasználó miatt látható az admin gomb is):



Ezt a menürendszert az alábbi kód teszi lehetővé:

Látható, hogy ez a rész megkap egy autorizációs adatot, amely alapján eldönti, hogy mit jelenítsen meg a weboldalon. Ez a kód implementálásra kerül mindegyik oldal fejlécében.

```
<div xmlns:th="http://www.thymeleaf.org"
      xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
  <div>
    <span sec:authorize="isAnonymous()">
      <a class="btn btn-light" th:href="@{/login}">Login</a>
      <a class="btn btn-light" th:href="@{/regisztral}">Sign up</a>
    </span>
    <span sec:authorize="isAuthenticated()">
      <a class="btn btn-light" th:href="@{/home}">Home</a>
      <a class="btn btn-light" th:href="@{/logout}">Logout</a>
    </span>
    <span sec:authorize="hasRole('ROLE_ADMIN')">
      <a class="btn btn-light" th:href="@{/admin/home}">Admin</a>
    </span>
  </div>
  <div sec:authorize="isAuthenticated()">
    <a>Welcome <span sec:authentication="principal.username">User</span></a>
  </div>
</div>
```

3. feladat

Feladat:

Legalább 3 felhasználói szerepet különböztessen meg:

- Admin, User, Látogató
- A menüpontok megjelenése és az oldalak elérhetősége változik attól függően,
- hogy melyik felhasználó használja az oldalt. (pl. admin oldal)

A korábban említett részen túl ezt a feladatot testesíti meg az, hogy az adott felhasználói szintek csak a nekik megfelelő oldalt láthatják:

- Látogató csak az alap oldalt (index) és a kapcsolatot láthatja.
- Bejelentkezett felhasználó a User nevű oldalt is láthatja
- Admin felhasználó láthatja az Admin oldalt

Ezeknek elnevezése változó és a HomeController felel érte, de a jogosultságkezelést a WebSecurityConfig végzi:

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(securedEnabled = true, proxyTargetClass = true)
public class WebSecurityConfig {

    @Autowired
    private UserDetailsServiceImpl userDetailsService;

    @Bean
    public static PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder();
    }

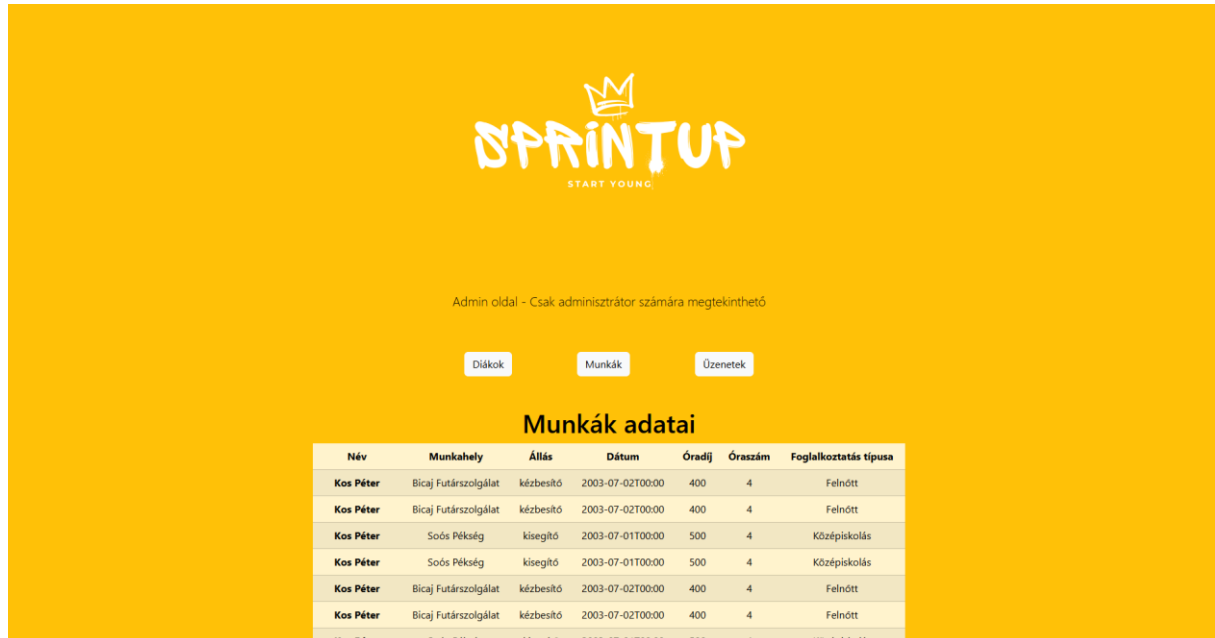
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http.csrf().csrfConfigurer<HttpSecurity> csrf -> csrf.disable()
            .authorizeHttpRequests(
                AuthorizationManagerRequestMatcher::auth -> auth
                    .requestMatchers(
                        @"/resources/**", @"/", @"/regisztral", @"/regisztral_feldolgoz", @"/munkahelyek", @"/kapcsolat**", @"/api**").anonymous()
                    .requestMatchers(
                        @"/resources/**", @"/", @"/home", @"/munkahelyek", @"/diakmunka", @"/kapcsolat**").authenticated()
                    .requestMatchers(
                        @"/admin/**", @"/diak", @"/uzenetek", @"/kapcsolat**", @"/munkak**").hasRole("ADMIN")
                )
            .formLogin(
                FormLoginConfigurer<HttpSecurity> form -> form
                    .defaultSuccessUrl("/home").permitAll()
            )
            .logout(
                LogoutConfigurer<HttpSecurity> logout -> logout
                    .logoutRequestMatcher(new AntPathRequestMatcher("/Logout"))
                    .logoutSuccessUrl("/")
                    .permitAll()
            );
        return http.build();
    }

    @Bean
    public AuthenticationManager authenticationManager(AuthenticationConfiguration configuration) throws Exception {
        return configuration.getAuthenticationManager();
    }
}
```


4. Feladat

feladat:

Legyen egy oldal, ahol a választott adatbázisból jelenít meg adatokat. Ehhez 3 tábla adatait használja fel az adatbázisból.



Az admin láthatja itt egybevonva mindhárom tábla adatait.

Ez az alábbi módon történik:

```
// Munkák adatai
@GetMapping("/{munkak}") 1 Hajdu-Gergo
public String munkakAdattai(Model model) {
    String str = munkakAdatok();
    model.addAttribute("str", str);
    return "admin";
}

private String munkakAdatok() { 1 usage 1 Hajdu-Gergo
    StringBuilder str = new StringBuilder();
    str.append("<h1>Munkák adatai</h1>");
    str.append("<table class='table table-warning table-striped w-50 mx-auto'>");
    str.append("<thead><tr><th scope='col'>Név</th><th scope='col'>Munkahely</th><th scope='col'>Állás</th><th scope='col'>Dátum</th><th scope='col'>Óradíj</th><th scope='col'>Óraszám</th><th scope='col'>Foglalkoztatás típusa</th></tr></thead><tbody>");
    for (Munka munka : munkaRepo.findAll()) {
        str.append("<tr>");
        str.append("<th scope='row'>");
        str.append(munka.getId().toString().append("</th><td>");
        str.append(munka.getMunkahely().toString().append("</td><td>");
        str.append(munka.getAllas().toString().append("</td><td>");
        str.append(munka.getDatum().toString().append("</td><td>");
        str.append(munka.getOradij().toString().append("</td><td>");
        str.append(munka.getOraszam().toString().append("</td><td>");
        str.append(munka.getFoglalkoztatasiTpus().toString().append("</td><td>");
        str.append("</tr>");
    }
    str.append("</tbody></table>");
    return str.toString();
}
```

Valamint az összekapcsolás a munka osztályban a következő módon valósul meg:

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Integer mhelyid;

@ManyToOne 3 usages
@JoinColumn(name = "diakaz")
private Diak diak;

@OneToOne 1 usage
@JoinColumn(name = "mhelyid")
private Munkahely mhely;
```

Kiegészítésként van egy táblás megjelenítés is:

Admin oldal - Csak adminisztrátor számára megtekinthető

Diákok

Munkák

Üzenetek

Diákok adatai

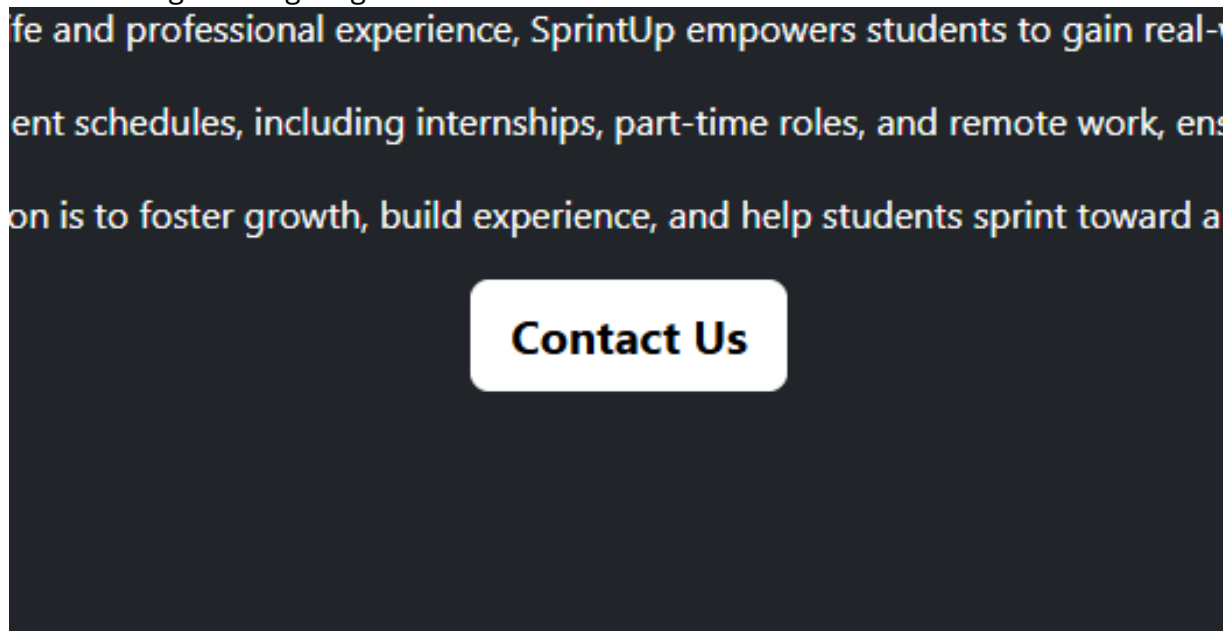
Azonosító	Név	Születési idő
1	Kos Péter	1987-11-05
2	Port Imre	1990-05-05
3	Kovács Imre	1984-03-02
4	Horváth Emil	1988-05-02
5	Kapos Petra	1985-12-23
6	Csóka Anna	1981-11-30
7	Nyúl Tamás	1988-02-16
8	Ordasi Emma	1989-01-03
9	Koppány Olga	1984-02-28
10	Kozma Patrícia	1983-06-01

2024 - SprintUp

5. feladat

Feladat:

Az egyik oldalon legyen egy kapcsolat űrlap, amelynek segítségével üzenetet lehet küldeni az oldal tulajdonosa számára. Ellenőrizze megfelelően az űrlap helyes kitöltését szerver oldali validációval is. Az elküldött Űrlap adatokat mentse le az adatbázisba. Ezt a funkciót a kapcsolat oldal valósítja meg amelyet bárki elérhet a főoldalról a Contact us gomb segítségével.



A gombot lenyomva az alábbi oldalra jutunk:

A screenshot of the SprintUp contact form page. The page has a dark blue background. At the top left is the "SprintUp" logo, and at the top right are "Login" and "Sign up" buttons. In the center, the "SPRINTUP" logo is displayed with a crown icon above it and the tagline "START YOUNG." below it. Below the logo, the heading "Kapcsolatfelvétel" is shown. Underneath the heading are two input fields: "Név:" and "Üzenet:". The "Üzenet:" field is a larger text area. Below these fields is a "Küldés" button. At the bottom center, the footer text "2024 - SprintUp" is visible.

Helyesen megadva az adatokat az URL-ben kapunk visszajelzést a sikeres elküldésről az adatbázisba.

Ezt a funkciót biztosító kód a következő:

```

@Autowired
private KapcsolatRepository kapcsolatRepository;

// Az űrlap megjelenítése
@GetMapping(Ⓜ"/kapcsolat") ⚡ Hajdu-Gergo
public String kapcsolatForm() {
    return "kapcsolat"; // Ez a kapcsolat.html fájl a resources/templates mappában van
}

// Az adatok fogadása és mentése
@PostMapping(Ⓜ"/kapcsolat") ⚡ Hajdu-Gergo
public String submitKapcsolat(
    @RequestParam("nev") String nev,
    @RequestParam("uzenet") String uzenet) {

    Kapcsolat kapcsolat = new Kapcsolat();
    kapcsolat.setNev(nev);
    kapcsolat.setUzenet(uzenet);
    kapcsolat.setDatum(LocalDate.now());

    if(kapcsolat.getNev().isEmpty() || kapcsolat.getUzenet().isEmpty())
        return "redirect:/kapcsolat?error";
    // Mentés az adatbázisba
    kapcsolatRepository.save(kapcsolat);

    return "redirect:/kapcsolat?success";
}

```

Láthatóan meghívásra kerül a modell, amely biztosítja a megfelelő adatokat a kapcsolat adatbázishoz.

Részlet a modelltől:

```

@Entity ⚡ Hajdu-Gergo
@Table(name = "kapcsolat")
public class Kapcsolat {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false) 2 usages
    private String nev;

    @Column(nullable = false, columnDefinition = "TEXT") 2 usages
    private String uzenet;

    @Column(nullable = false, updatable = false, columnDefinition = "TIMESTAMP DEFAULT CURRENT_TIMESTAMP") 2 usages
    private LocalDateTime datum;

    // Getters and setters
    public Long getId() { return id; }

    public void setId(Long id) { this.id = id; }

    public String getNev() { 2 usages ⚡ Hajdu-Gergo
        return nev;
    }

    public void setNev(String nev) { 1 usage ⚡ Hajdu-Gergo
        this.nev = nev;
    }

    public String getUzenet() { 2 usages ⚡ Hajdu-Gergo
        return uzenet;
    }

    public void setUzenet(String uzenet) { 1 usage ⚡ Hajdu-Gergo
        this.uzenet = uzenet;
    }
}

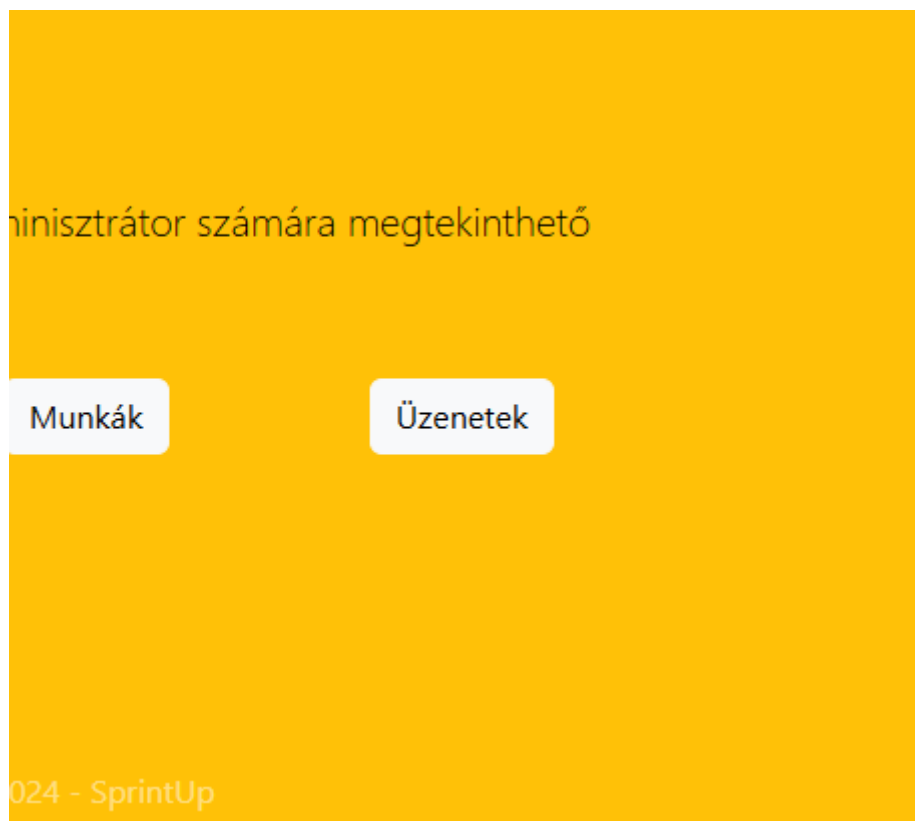
```

6. feladat

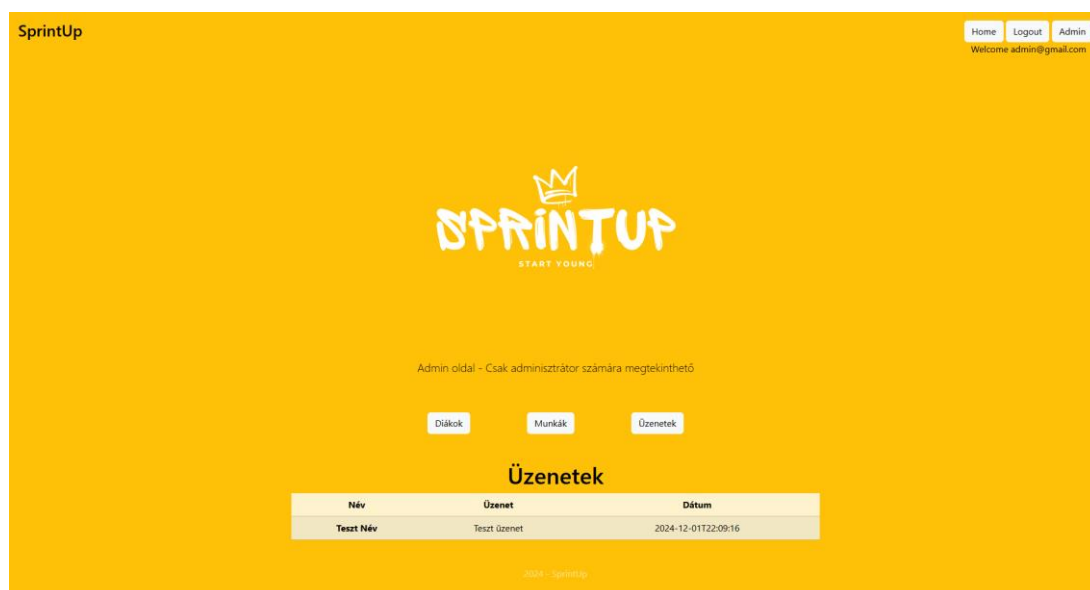
Feladat:

Tegye lehetővé megtekinteni egy hatodik oldalon táblázatban az előző pontban elküldött üzeneteket az adatbázisból fordított időrend szerint (a legfrissebb legyen elől)




Ezt az admin oldalon lévő üzenetek rész valósítja meg



A gombra kattintva az alábbi módon jeleníti meg az applikáció az adatokat:



Adatbázisban is megjelennek az adatok:

	id	nev	uzenet	datum
<input type="checkbox"/>  Módosítás  Másolás  Törölés	1	Teszt Név	Teszt üzenet	2024-12-01 22:09:16

7. feladat

Feladat: Valósítson meg az alkalmazásban egy RESTful API-t.

Ezt a funkcionalitást az alábbi kód valósítja meg:

```
@RestController  Hajdu-Gergo
@RequestMapping("/api/diakok")
public class DiakController {

    @Autowired
    private DiakRepository diakRepository;

    // Végpont a diákok számának lekérdezésére
    @GetMapping("/szam")  Hajdu-Gergo
    public Long getDiakokSzama() { return diakRepository.count(); }

    @GetMapping("/osszes")  Hajdu-Gergo
    public Iterable<Diak> getDiakok() {
        return diakRepository.findAll();
    }
}
```

CURL:

```
PS C:\Users\gergo> curl http://localhost:8080/api/diakok/osszes
StatusCode      : 200
StatusDescription : 
Content         : [{"diakaz":1,"nev":"Kos P  ter","szulido":"1987-11-05"},{"diakaz":2,"nev":"Port Imre","szulido":"1990-05-05"},{"diakaz":3,"nev":"Kov  cs Imre","szulido":"1984-03-02"},{"diakaz":4,"nev":"Horv  th Emil"...
RawContent      : HTTP/1.1 200
                  Vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers
                  X-Content-Type-Options: nosniff
                  X-XSS-Protection: 0
                  Pragma: no-cache
                  X-Frame-Options: DENY
                  Transfer-Encoding: ...
Forms           : {}
Headers         : {[Vary, Origin,Access-Control-Request-Method,Access-Control-Request-Headers], [X-Content-Type-Options, nosniff], [X-XSS-Protection, 0], [Pragma, no-ca
                  che]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 569
```

Postman:

GET http://localhost:8080/api/ + ...

HTTP http://localhost:8080/api/diakok/osszes

GET http://localhost:8080/api/diakok/osszes

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON

```
1  {
2    {
3      "diakaz": 1,
4      "nev": "Kos Péter",
5      "szulido": "1987-11-05"
6    },
7    {
8      "diakaz": 2,
9      "nev": "Port Imre",
10     "szulido": "1990-05-05"
11   },
12   {
13     "diakaz": 3,
14     "nev": "Kovács Imre",
15     "szulido": "1984-03-02"
16   },
17   {
18     "diakaz": 4,
19     "nev": "Horváth Emil",
20     "szulido": "1988-05-02"
21   },
22   {
23     "diakaz": 5,
24     "nev": "Kapos Petra",
25     "szulido": "1985-12-23"
26   },
27   {
28     "diakaz": 6,
29     "nev": "Csóka Anna",
30     "szulido": "1981-11-30"
31   },
```

```
32   {
33     "diakaz": 7,
34     "nev": "Nyúl Tamás",
35     "szulido": "1988-02-16"
36   },
37   {
38     "diakaz": 8,
39     "nev": "Ordasi Emma",
40     "szulido": "1989-01-03"
41   },
42   {
43     "diakaz": 9,
44     "nev": "Koppány Olga",
45     "szulido": "1984-02-28"
46   },
47   {
48     "diakaz": 10,
49     "nev": "Kozma Patricia",
50     "szulido": "1983-06-01"
51   }
52 }
```

GitHub repository

Az 8. és 9. feladatot testesíti meg.

A beadandó egyedül készült el ezért GitHubon 2 account felhasználásával készültek a commitok.

Ennek elérése az alábbi linken lehetséges:

<https://github.com/Hajdu-Gergo/JavaGyakorlatBeadando.git>

Futtatható JAR fájl.

Ezen eredeti elérése a /out/artifacts/beadando_jar mappában van, de a főkönyvtárban is megtalálható „futtat.jar” -ként

Adatbázisimportáláshoz szükséges SQL fájl a főkönyvtárban található „adatok.sql” néven.

A feladat során JDK 17 került alkalmazásra.