



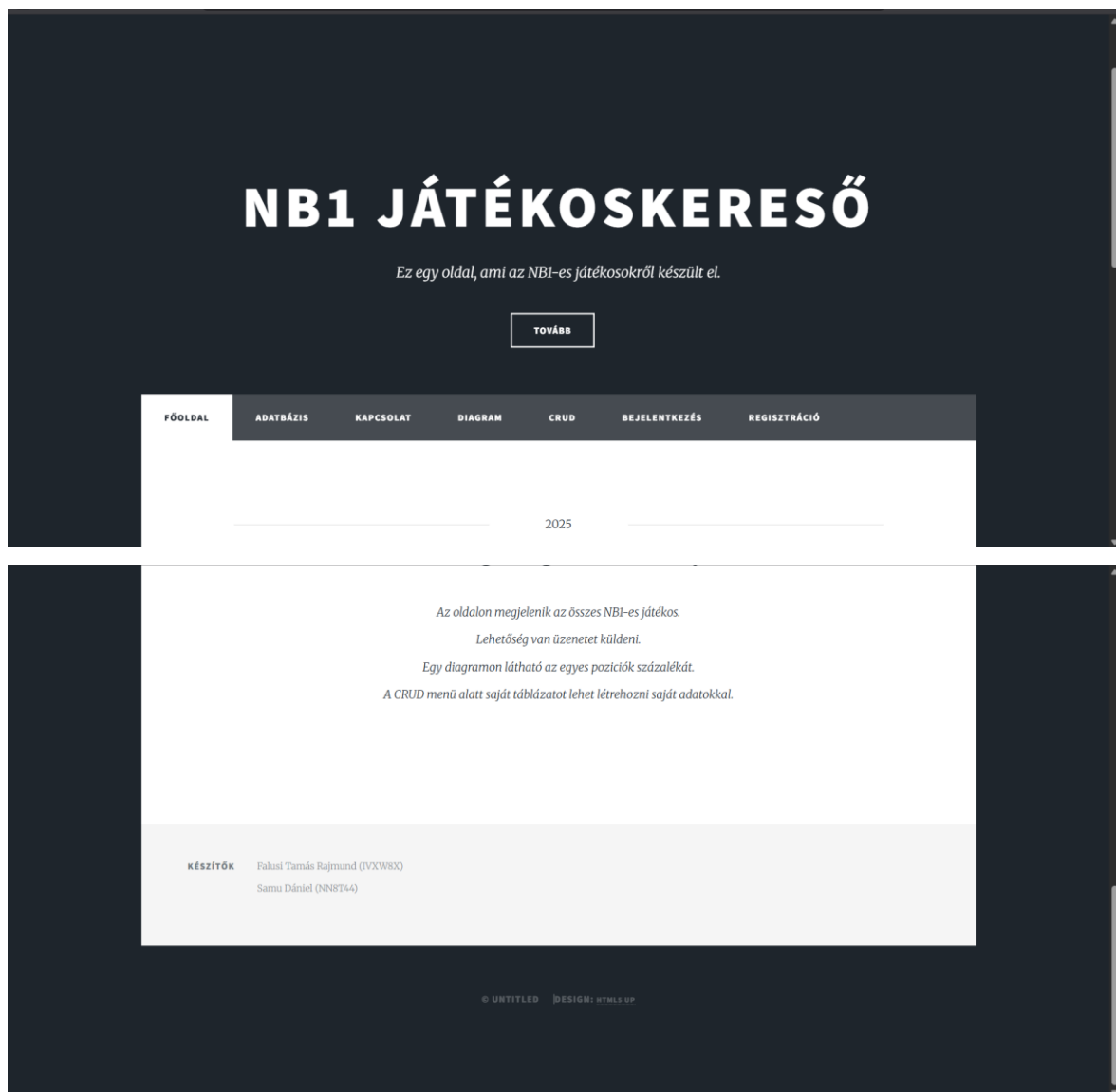
Neumann János Egyetem  
Műszaki és Informatikai  
Kar

# Java Spring Boot projektfeladat

Falusi Tamás Rajmund, Samu Dániel Botond  
IVXW8X, NN8T44

## 1. Reszponzív Téma

Az oldalunkon a <https://html5up.net/> egyik reszponzív sablonját használtuk fel. Az oldalon egy kevés átalakítással a saját formánkra alakítottuk át, így letisztult megjelenéshez és egyértelmű használathoz vezetett. Az oldal egyes részei mindegyik oldalon megegyeznek ezért ezek szerkeszthetősége érdekében külön html elemekbe szerveztük ki. Ezek a navbar illetve a footer. A navbar felelős az egyszerű navigálásért az oldalon. Az adatbázis menün egy adatbázisból jelenítünk meg rekordokat, a kapcsolat oldalon a felhasználó bejelentkezést követően írhat üzenetet, a diagram oldalon egy kördiagramban ábrázoljuk a játékosok posztjainak eloszlását, a CRUD menü egy táblázatba és adatbázisba írást olvasást szerkesztést és törlést valósít meg.



```

1 <nav id="nav" th:fragment="nav(active)" xmlns:th="http://www.thymeleaf.org
2   xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
3     <ul class="links">
4
5       <li th:classappend="${active} == 'fooldal' ? 'active': ''">
6         <a th:href="@{/}">Főoldal</a>
7       </li>
8
9       <li th:classappend="${active} == 'adatbazis' ? 'active': ''">
10        <a th:href="@{/adatbazis}">Adatbázis</a>
11      </li>
12
13      <li th:classappend="${active} == 'kapcsolat' ? 'active': ''">
14        <a th:href="@{/kapcsolat}">Kapcsolat</a>
15      </li>
16
17      <li sec:authorize="isAuthenticated()"
18        th:classappend="${active} == 'uzenet' ? 'active': ''">
19        <a th:href="@{/uzenet}">Üzenetek</a>
20      </li>
21
22      <li th:classappend="${active} == 'diagram' ? 'active': ''">
23        <a th:href="@{/diagram}">Diagram</a>
24      </li>
25
26      <li th:classappend="${active} == 'crud' ? 'active': ''">
27        <a th:href="@{/crud}">CRUD</a>
28      </li>
29

```

```

30      <li sec:authorize="hasAuthority('ADMIN')"
31        th:classappend="${active} == 'admin' ? 'active': ''">
32        <a th:href="@{/admin}">Admin</a>
33      </li>
34
35      <li sec:authorize="isAuthenticated()">
36        <a sec:authentication="principal.nev"></a>
37      </li>
38
39      <li sec:authorize="isAuthenticated()">
40        <form method="POST" th:action="@{/logout}" style="display:inline;">
41          <button class="btn btn-danger">Kijelentkezés</button>
42        </form>
43      </li>
44
45      <li sec:authorize="isAnonymous()">
46        <a th:href="@{/bejelentkezés}">Bejelentkezés</a>
47      </li>
48
49      <li sec:authorize="isAnonymous()">
50        <a th:href="@{/regisztracio}">Regisztráció</a>
51      </li>
52    </ul>
53  </nav>

```

```

1 <footer id="footer">
2   <section class="split contact">
3     <section class="alt">
4       <h3>Készítők</h3>
5       <p>Falusi Tamás Rajmund (IVXW8X)<br/>Samu Dániel (NN8T44)</p>
6     </section>
7   </section>
8 </footer>
9
10 <div id="copyright">
11   <ul>
12     <li>© Untitled</li>
13     <li>Design: <a href="https://html5up.net">HTML5 UP</a></li>
14   </ul>
15 </div>
16

```

## 2. Authentikáció

### Cél

Az oldalra egyes funkciókhoz csak beregisztrálva vagy bejelentkezve férhetnek hozzá a felhasználók, ilyen az *üzenetek* menü vagy az üzent írása a *kapcsolat* oldalon, ami bár egy nem bejelentkezett látogató szintű felhasználó is megtekinthet, de írni nem tud. A *bejelentkezés* és *regisztráció* eltér az oldalunk sablonjától.

Az autentikációt megvalósító java elemek:

- **User (model)**
- **UserRepository (repository)**
- **AuthController**
- **CustomUserDetailsService**
- **SecurityConfig**
- **UserPrincipal**
- **UserService**
- **HomeController**

A SecurityConfig felel azért, hogy a felhasználó mit látogathat meg és milyen szinten, a bejelentkezés és kijelentkezésért.

A UserPrincipal felel azért, hogy a saját User objektumból átalakítja az adatokat a Spring által használható formára.

A CustomUserDetailsService felelős azért, hogy bejelentkezéskor az email alapján megkeresse a felhasználót az adatbázisban, és azt egy UserPrincipal objektummá alakítva visszaadja a Spring Security-nek.

A UserService felel a regisztrációért, itt adjuk meg a felhasználó szerepkörét, hash módszerrel titkosítjuk a jelszót majd elmentjük a felhasználót, illetve automatikus bejelentkezést is megvalósít.

Az AuthController-ben a regisztráció oldal post kérésére lekéri a jelszót, majd a UserService metódusait használva beregisztrálja majd bejelentkezteti a felhasználót.

A HomeController felel a bejelentkezés és regisztráció oldal megjelenítéséért.

## Bejelentkezés

Email cím

Jelszó

Bejelentkezés

Vissza

## Regisztráció

Név

Email cím

Jelszó

Jelszó megerősítése

regisztráció

Vissza

```
1 package com.example.javagyaknb1.model;
2 import jakarta.persistence.*;
3 import java.util.List;
4
5 @Entity
6 @Table(name = "users")
7 public class User {
8     @Id
9     @GeneratedValue(strategy = GenerationType.IDENTITY)
10    private int id;
11    @Column(name = "nev") 2 usages
12    private String nev;
13    @Column(name = "email") 2 usages
14    private String email;
15    @Column(name = "password") 2 usages
16    private String password;
17    @Column(name = "role") 2 usages
18    private String role;
19    @OneToMany(mappedBy = "user") 2 usages
20    private List<Uzenet> uzenetek;
21
22    public User() {} new *
23    public int getId() {return id;} new *
24    public void setId(int id) {this.id = id;} new *
25    public String getNev() {return nev;} new *
26    public void setNev(String nev) {this.nev = nev;} new *
27    public String getEmail() {return email;} 5 usages new *
28    public void setEmail(String email) {this.email = email;} no usages new
29    public String getPassword() {return password;} new *
30    public void setPassword(String password) {this.password = password;}
31    public String getRole() {return role;} new *
32    public void setRole(String role) {this.role = role;} new *
```

```

1 package com.example.javagyaknb1.config;
2 import java.util.Collection;
3 import com.example.javagyaknb1.model.User;
4 import com.example.javagyaknb1.repository.UserRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.security.core.GrantedAuthority;
7 import org.springframework.security.core.authority.AuthorityUtils;
8 import org.springframework.security.core.userdetails.UserDetails;
9 import org.springframework.security.core.userdetails.UserDetailsService;
10 import org.springframework.security.core.userdetails.UsernameNotFoundException;
11 import org.springframework.stereotype.Service;
12 import org.springframework.transaction.annotation.Transactional;
13
14 @Service & FanTomi2 *
15 @Transactional
16 public class CustomUserDetailsService implements UserDetailsService {
17     @Autowired
18     private UserRepository userRepo;
19
20     @Override no usages & FanTomi2
21     public UserDetails loadUserByUsername(String userName) {
22         User user = userRepo.findByEmail(userName)
23             .orElseThrow(() -> new UsernameNotFoundException("Email " + userName + " not found"));
24
25         return new UserPrincipal(user, getAuthorities(user));
26     }
27
28     @ private static Collection<? extends GrantedAuthority> getAuthorities(User user) { 1 usage & FanTomi2
29         Collection<GrantedAuthority> authorities = AuthorityUtils.createAuthorityList(user.getRole());
30         return authorities;
31     }
32 }

```

```

15 @Configuration & FanTomi2 +1 *
16 @EnableWebSecurity
17 public class SecurityConfig {
18     @Autowired
19     private UserDetailsService userDetailsService;
20     @Bean & FanTomi2
21     public static PasswordEncoder passwordEncoder(){
22         return new BCryptPasswordEncoder();
23     }
24     @Bean & FanTomi2 +1 *
25     public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
26         http
27             .csrf( <CsrfConfigurer<HttpSecurity> csrf -> csrf.disable())
28             .authorizeHttpRequests(
29                 <AuthorizationManagerRequestMatcher<?> auth -> auth
30                     .requestMatchers( <String>"/**", <String>"/login", <String>"/bejelentkezes", <String>"/regisztracio",
31                                     <String>"/resources/**", <String>"/stylesheets/**", <String>"/images/**", <String>"/javascripts/**").permitAll()
32                     .requestMatchers( <String>"/uzenetek").authenticated()
33                     .requestMatchers( <String>"/admin/**").hasAuthority("ADMIN")
34                 )
35             .formLogin(
36                 <FormLoginConfigurer<HttpSecurity> form -> form
37                     .loginPage( <String>"/bejelentkezes")
38                     .loginProcessingUrl( <String>"/login")
39                     .defaultSuccessUrl( <String>"/", alwaysUse: true)
40                     .failureUrl( <String>"/authenticationFailureUrl: "/bejelentkezes?error")
41                     .permitAll()
42                 )
43             .logout(
44                 <LogoutConfigurer<HttpSecurity> logout -> logout
45                     .logoutSuccessUrl( <String>"/")
46                     .permitAll()
47             );
48         return http.build();
49     }
50 }

```

```

7 public class UserPrincipal implements UserDetails { 1 usage 2 FanTomi2 *
8     private final User user; 6 usages
9     private final Collection<? extends GrantedAuthority> authorities; 2 usages
10
11     public UserPrincipal(User user, Collection<? extends GrantedAuthority> authorities) {
12         this.user = user;
13         this.authorities = authorities;
14     }
15
16     public String getUsername() { 2 FanTomi2
17         return user.getUsername();
18     }
19
20     public String getEmail() { no usages 2 FanTomi2
21         return user.getEmail();
22     }
23
24     public String getRole() { 2 FanTomi2
25         return user.getRole();
26     }
27
28     @Override no usages 2 FanTomi2
29     public Collection<? extends GrantedAuthority> getAuthorities() {
30         return authorities;
31     }
32
33     @Override 2 FanTomi2
34     public String getPassword() {
35         return user.getPassword();
36     }

```

```

7 public class UserPrincipal implements UserDetails { 1 usage
37
38     @Override 2 FanTomi2
39     public String getUsername() {
40         return user.getUsername();
41     }
42
43     @Override no usages 2 FanTomi2
44     public boolean isAccountNonExpired() {
45         return true;
46     }
47
48     @Override no usages 2 FanTomi2
49     public boolean isAccountNonLocked() {
50         return true;
51     }
52
53     @Override no usages 2 FanTomi2
54     public boolean isCredentialsNonExpired() {
55         return true;
56     }
57
58     @Override 2 FanTomi2
59     public boolean isEnabled() {
60         return true;
61     }
62 }

```

```

13  @Service 2 usages & FanTomi2 *
14  public class UserService {
15      @Autowired private UserRepository userRepo;
16      @Autowired private PasswordEncoder encoder;
17      @Autowired private AuthenticationManager authenticationManager;
18
19      @Transactional & FanTomi2 *
20  @ public void register(User user) {
21      // role
22      if (user.getEmail().equals("admin@admin")) {
23          user.setRole("ADMIN");
24      } else {
25          user.setRole("USER");
26      }
27      // hash
28      String rawPassword = user.getPassword();
29      user.setPassword(encoder.encode(rawPassword));
30
31      userRepo.save(user);
32  }
33
34  public void autoLogin(String email, String rawPassword) { 1 usage & FanTomi2
35      UsernamePasswordAuthenticationToken token =
36          new UsernamePasswordAuthenticationToken(email, rawPassword);
37      Authentication auth = authenticationManager.authenticate(token);
38      SecurityContextHolder.getContext().setAuthentication(auth);
39  }
40  }

```

```

1  package com.example.javagyaknb1.repository;
2
3  import com.example.javagyaknb1.model.User;
4  import org.springframework.data.repository.CrudRepository;
5  import java.util.Optional;
6
7  public interface UserRepository extends CrudRepository<User,Integer> {
8      Optional<User> findByEmail(String email); 3 usages & FanTomi2
9  }
10

```

```

10  @Controller & FanTomi2 *
11  public class AuthController {
12      @Autowired private UserService userService;
13
14      @PostMapping("/regisztracio") & FanTomi2
15  @ public String register(@ModelAttribute User user) {
16
17      String rawPassword = user.getPassword();
18      userService.register(user);
19      userService.autoLogin(user.getEmail(), rawPassword);
20      return "redirect:/";
21  }
22  }

```



```

11 <body>
12 <div class="container">
13 <h1>Bejelentkezés</h1>
14 <form method="post" th:action="@{/login}">
15 <div class="mb-3">
16 <label for="email" class="form-label">Email cím</label>
17 <input type="email" class="form-control" id="email" name="username">
18 </div>
19 <div class="mb-3">
20 <label for="password" class="form-label">Jelszó</label>
21 <input type="password" class="form-control" id="password" name="password">
22 </div>
23 <button id="login" type="submit" class="btn btn-success">Bejelentkezés</button>
24 <button type="button" class="btn btn-info"><a th:href="@{/}"
25 class="link-offset-2 link-underline link-underline-opacity-0">Vissza</a></button>
26 <p th:if="${param.error}">Helytelen email vagy jelszó!</p>
27 </form>
28 </div>

```

```

11 <body>
12 <div class="container">
13 <h1>Regisztráció</h1>
14 <form method="post" th:action="@{/regisztracio}">
15 <div class="mb-3">
16 <label for="nev" class="form-label">Név</label>
17 <input type="text" class="form-control" id="nev" name="nev">
18 </div>
19 <div class="mb-3">
20 <label for="email" class="form-label">Email cím</label>
21 <input type="email" class="form-control" id="email" name="email">
22 </div>
23 <div class="mb-3">
24 <label for="password" class="form-label">Jelszó</label>
25 <input type="password" class="form-control" id="password" name="password">
26 </div>
27 <div class="mb-3">
28 <label for="jelszoEllen" class="form-label">Jelszó megerősítése</label>
29 <input type="password" class="form-control" id="jelszoEllen" name="jelszoEllen">
30 </div>
31 <button id="register" type="submit" class="btn disabled btn-danger">regisztráció</button>
32 <button type="button" class="btn btn-info"><a th:href="@{/}"
33 class="link-offset-2 link-underline link-underline-opacity-0">Vissza</a></button>
34 <p th:if="${param.error}">Hiba történt a regisztrációkor!</p>
35 </form>
36 </div>

```

```

8 @Controller
9 public class HomeController {
10
11     @GetMapping("/")
12     public String index() { return "fooldal"; }
13
14
15
16     @GetMapping("/bejelentkezés")
17     public String login() {
18         return "bejelentkezés";
19     }
20
21
22     @GetMapping("/regisztracio")
23     public String registerPage(Model model) {
24         model.addAttribute("user", new User());
25         return "regisztracio";
26     }
27 }

```

### 3. Főoldal

#### Cél

A weboldal egyszerű bemutatása pár mondatban.

A **HomeController** egyszerű get kérésre a fooldal nézetet adja vissza.



```
8 @Controller  SDaniel0210 +1 *
9 public class HomeController {
10
11     @GetMapping("/")  SDaniel0210
12     public String index() { return "fooldal"; }
13
14
15
16     @GetMapping("/bejelentkezes")  FanTomi2 +1
17     public String login() {
18         return "bejelentkezes";
19     }
20
21
22     @GetMapping("/regisztracio")  FanTomi2 +1
23     public String registerPage(Model model) {
24         model.addAttribute(attributeName: "user", new User());
25         return "regisztracio";
26     }
27 }
```

## 4. Adatbázis menü

### Cél

Az *Adatbázis* menüpont célja, hogy az NB1 játékosok adatbázisát egy strukturált, jól olvasható táblázatban jelenítse meg.

Az oldal a Massively (HTML5 UP) sablonra épül, ezért a megjelenés:

- világos háttérkártyán futó táblázat,
- minimalista, modern tipográfia,
- a sablonra jellemző széles margók,
- középre rendezett tartalom,
- egységes világos-sötét kontrasztok.

A rendszer Spring Boot + JPA + Thymeleaf segítségével tölti be a MySQL adatbázisban tárolt rekordokat.

Az adatok betöltéséért az **AdatokController** felel.

A vezérlő:

1. Lekéri az összes játékost a labdarugo táblából
2. Csatolja hozzá a kapcsolódó klubot és posztot
3. A listát átadja a Thymeleaf sablonnak
4. Megjelenítés: adatok.html

NB1

FŐOLDAL	ADATBÁZIS	KAPCSOLAT	DIAGRAM	CRUD	BEJELENTKEZÉS	REGISZTRÁCIÓ
---------	-----------	-----------	---------	------	---------------	--------------

### NB1 JÁTEKOSOK TÁBLÁZATA

MEZSZÁM	NÉV	KLUB	POSZT	LÉGIÓS	SZÜLETÉSI IDŐ	ÉRTÉK
19	Andric Nemanja	Balmazújváros FC	bal szélső	Nem	1987-06-13	250 ezer €
8	Arabuli Bachana	Balmazújváros FC	középcsatár	Igen	1994-01-05	300 ezer €
21	Batarelo Ante	Balmazújváros FC	védekező középpályás	Igen	1984-11-21	150 ezer €

```

1 package com.example.javagyaknb1.controller;
2
3 import com.example.javagyaknb1.repository.LabdarugoRepository;
4 import org.springframework.stereotype.Controller;
5 import org.springframework.ui.Model;
6 import org.springframework.web.bind.annotation.GetMapping;
7
8 @Controller @S Daniel0210
9 public class AdatokController {
10
11     private final LabdarugoRepository labdarugoRepository; 2 usages
12
13     public AdatokController(LabdarugoRepository labdarugoRepository) { @S Daniel0210
14         this.labdarugoRepository = labdarugoRepository;
15     }
16
17     @GetMapping("/adatbazis") @S Daniel0210
18     public String adatbazis(Model model) {
19         var labdarugok = labdarugoRepository.findAllWithKlubAndPosztSorted();
20         model.addAttribute("labdarugok", labdarugok);
21         return "adatok";
22     }
23 }
24

```

## 5. Kapcsolat menü

### Cél

A felhasználó üzenetet tud írni a weboldal működtetőjének. Ezeket az üzeneteket adatbázisba mentjük el és eltároljuk minden üzenethez a felhasználó id-ját az oldal további funkciói megfelelően működéséhez. A látogató a kapcsolat oldalon nem tud beírni bejelentkezés nélkül.

Az üzenetírás menete:

Bejelentkezés

Cím és üzenet

Elküldés gombra kattintva elmenti az adatbázisban

Ezt az **Uzenet** modell, az **UzenetRepository**, és az **UzenetController** végzi el.

Az **Uzenet** modell az adatbázisunk üzenetek táblának megfelelő adatok tárolásáért felelős, itt mentjük el az elküldés idejét az onCreate metódussal.

Az **UzenetRepository** felelős az adatok elmentéséért és azok lekérdezéséért egy adott felhasználó számára vagy az összes üzenetet, ami majd az admin oldalon lesz jelentősége.

Az **UzenetController** felelős azért, hogy megjelenítse és elmentse az elküldött üzenetet.

The image displays two screenshots of a web application's contact page, labeled 'KAPCSOLAT' in the navigation bar. The page title is 'IRJON NEKÜNK ÜZENETET!' (Write us a message!).

The top screenshot shows the form with a red error message 'BEJELENTKEZÉS SZÜKSÉGES!' (Login required) displayed above the input fields. The 'CÍM' (Name) field is empty, and the 'ÜZENET' (Message) field is also empty. The 'ELKÜLDÉS' (Send) button is visible at the bottom.

The bottom screenshot shows the form with the 'CÍM' field containing the text 'Teszt1' and the 'ÜZENET' field containing the text 'Ez egy teszt üzenet' (This is a test message). The 'ELKÜLDÉS' button is also visible.

```

36 <div id="main">
37 <h2>Írjon nekünk Üzenetet!</h2>
38 <!-- Posts -->
39 <section class="post featured">
40 <p th:if="${param.sikeres}" style="background-color: #d4edda;">Üzenet sikeresen elküldve!</p>
41 </p>
42 <h4 style="background-color: #d4edda; border: 1px solid #c3e6cb; padding: 5px;">Bejelentkezés szükséges!</h4>
43 <section>
44 <form method="post" th:action="@{/kapcsolat}" th:object="${formUzenet}">
45 <div class="fields">
46 <div class="field">
47 <label>Cím</label>
48 <input type="text" th:field="*{cim}"
49 th:disabled="${#authorization.expr('isAnonymous()')}">
50 </div>
51 <div class="field">
52 <label>Üzenet</label>
53 <textarea th:field="*{uzenet}"
54 th:disabled="${#authorization.expr('isAnonymous()')}"></textarea>
55 </div>
56 </div>
57 <ul class="actions" id="middle">
58 <li><input type="submit" value="Elküldés" th:disabled="${#authorization.expr('isAnonymous()')}" /></li>
59 </ul>
60 </form>
61 </section>
62 </div>

```

```

4 import java.time.LocalDateTime;
5
6 @Entity & FanTomi2 *
7 @Table(name="uzenetek")
8 public class Uzenet {
9     @Id
10     @GeneratedValue(strategy = GenerationType.IDENTITY)
11     private int id;
12     @ManyToOne(2 usages)
13     @JoinColumn(name = "user_id", nullable = false)
14     private User user;
15     @Column(name = "cim") 2 usages
16     private String cim;
17     @Column(name = "uzenet") 2 usages
18     private String uzenet;
19     @Column(name = "letrehozás") 3 usages
20     private LocalDateTime letrehozás;
21
22     @PrePersist & FanTomi2
23     public void onCreate() {
24         this.letrehozás = LocalDateTime.now();
25     }
26
27     public int getId() {return id;} new *
28     public void setId(int id) {this.id = id;} new *
29     public User getUser() {return user;} new *
30     public void setUser(User user) {this.user = user;} new *
31     public String getCim() {return cim;} no usages new *
32     public void setCim(String cim) {this.cim = cim;} no usages new *
33     public String getUzenet() {return uzenet;} no usages new *
34     public void setUzenet(String uzenet) {this.uzenet = uzenet;} no usages new *
35     public LocalDateTime getLetrehozás() {return letrehozás;} no usages new *

```

```

8 public interface UzenetRepository extends CrudRepository<Uzenet, Integer> { 2 us
9     List<Uzenet> findByUserOrderByLetrehozásDesc(User user); // saját üzenetek
10     List<Uzenet> findAllByOrderByLetrehozásDesc(); 1 usage & FanTomi2
11 }

```

```

11  @Service 2 usages 2 FanTomi2 *
12  public class UzenetService {
13      @Autowired private UzenetRepository uzenetRepo;
14
15      public void save(Uzenet uzenet) { 2 FanTomi2
16          uzenetRepo.save(uzenet);
17      }
18
19      public List<Uzenet> getUserUzenetek(User user) { 1 usage 2 FanTomi2
20          return uzenetRepo.findByUserOrderByLetrehozasc(user);
21      }
22
23      public List<Uzenet> getAllUzenetek() { 1 usage 2 FanTomi2
24          return uzenetRepo.findAllByOrderByLetrehozasc();
25      }
26
27      public Optional<Uzenet> findById(int id) { 1 usage 2 FanTomi2
28          return uzenetRepo.findById(id);
29      }
30
31
32      public void delete(int id) { 2 usages 2 FanTomi2
33          uzenetRepo.deleteById(id);
34      }
35  }
36
37  @Controller 2 FanTomi2 *
38  public class uzenetController {
39      @Autowired private UserRepository userRepo;
40      @Autowired private UzenetService uzenetService;
41
42      @GetMapping("/kapsolat") 2 FanTomi2
43      public String kapsolat(Model model) {
44          model.addAttribute("formUzenet", new Uzenet());
45          return "kapsolat";
46      }
47
48      @PostMapping("/kapsolat") 2 FanTomi2 *
49      public String kuldes(@ModelAttribute("formUzenet") Uzenet beUzenet,
50                          @AuthenticationPrincipal UserDetails currentUser) {
51          User user = userRepo.findByEmail(currentUser.getUsername()).get();
52          beUzenet.setUser(user);
53
54          uzenetService.save(beUzenet);
55
56          return "redirect:/kapsolat?sikeres";
57      }
58  }

```

## 6. Üzenetek menü

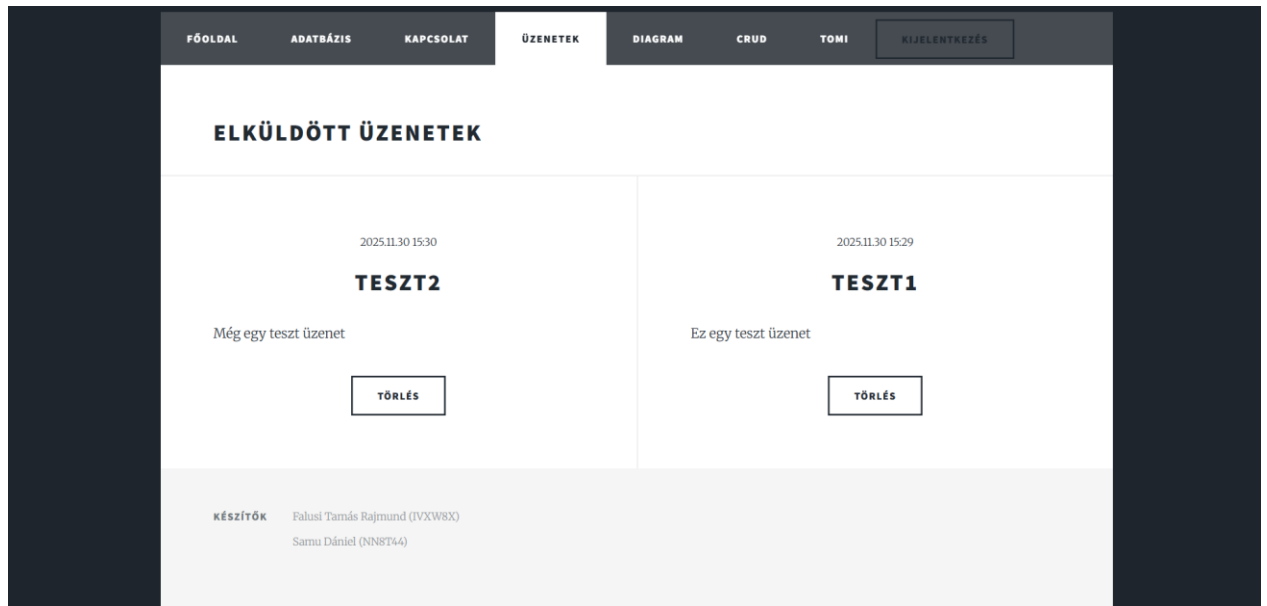
### Cél

Kapcsolat oldalon elküldött üzenetek megjelenítése és egyes üzenetek törlése.

Ezt az **UzenetService** és az **UzenetController** és az *uzenet* html valósít meg.

Az **UzenetService** felel az üzenetek lekérdezéséért az összesért és az egyes felhasználók üzenetéért.

Az **UzenetController** felel az üzenetek megjelenítéséért és a törlésért.



```
36 @GetMapping("/{uzenet}") @FanTomi2
37 public String sajátUzenetek(Model model, @AuthenticationPrincipal UserDetails currentUser) {
38     User user = userRepo.findByEmail(currentUser.getUsername()).orElseThrow();
39     model.addAttribute(attributeName: "uzenetek", uzenetService.getUserUzenetek(user));
40     return "uzenet";
41 }
42
43 @PostMapping("/{uzenet}/{id}/delete") @FanTomi2
44 public String delete(@PathVariable int id, @AuthenticationPrincipal UserDetails currentUser) {
45
46     Uzenet u = uzenetService.findById(id).orElseThrow();
47
48     // csak a saját üzenetét törölhesse
49     if (!u.getUser().getEmail().equals(currentUser.getUsername())) {
50         return "redirect:/uzenet?hiba";
51     }
52
53     uzenetService.delete(id);
54     return "redirect:/uzenet?torolve";
55 }
```



```

11 @Service 2 usages 2 FanTomi2 *
12 public class UzenetService {
13     @Autowired private UzenetRepository UzenetRepo;
14
15     public void save(Uzenet uzenet) { 2 FanTomi2
16         UzenetRepo.save(uzenet);
17     }
18
19     public List<Uzenet> getUserUzenetek(User user) { 1 usage 2 FanT
20         return UzenetRepo.findByUserOrderByLetrehozasc(user);
21     }
22
23     public List<Uzenet> getAllUzenetek() { 1 usage 2 FanTomi2
24         return UzenetRepo.findAllByOrderByLetrehozasc();
25     }
26
27     public Optional<Uzenet> findById(int id) { 1 usage 2 FanTomi2
28         return UzenetRepo.findById(id);
29     }
30
31
32     public void delete(int id) { 2 usages 2 FanTomi2
33         UzenetRepo.deleteById(id);
34     }
35 }

```

```

40 <div id="main">
41     <h2>Elküldött üzenetek</h2>
42
43     <section class="posts">
44         <article th:each="uzenet : ${uzenetek}">
45             <header>
46                 <span class="date"
47                     th:text="${#temporals.format(uzenet.letrehozasc, 'yyyy.MM.dd HH:mm')}">
48                 </span>
49
50                 <h2 th:text="${uzenet.cim}"></h2>
51             </header>
52
53             <p th:text="${uzenet.uzenet}"></p>
54
55             <form th:action="@{/uzenet/{id}/delete(id=${uzenet.id})}"
56                 method="post"
57                 onsubmit="return confirm('Biztosan törölni szeretnéd?')">
58                 <button type="submit" class="btn">Törlés</button>
59             </form>
60         </article>
61     </section>
62 </div>

```

## 7. Diagram menü

### Cél

A Diagram menüpont célja, hogy vizuálisan szemléltessen egy statisztikát az NB1 játékosokról.

Az alkalmazásban ez a fejezet a játékosok pozíció szerinti megoszlását mutatja be egy kördiagramon (pie chart).

A megoldás modern, reszponzív grafikonkönyvtárat használ: Chart.js.

A diagram adatait a backend dinamikusan szolgáltatja egy REST végponton keresztül.

### Architektúra áttekintés

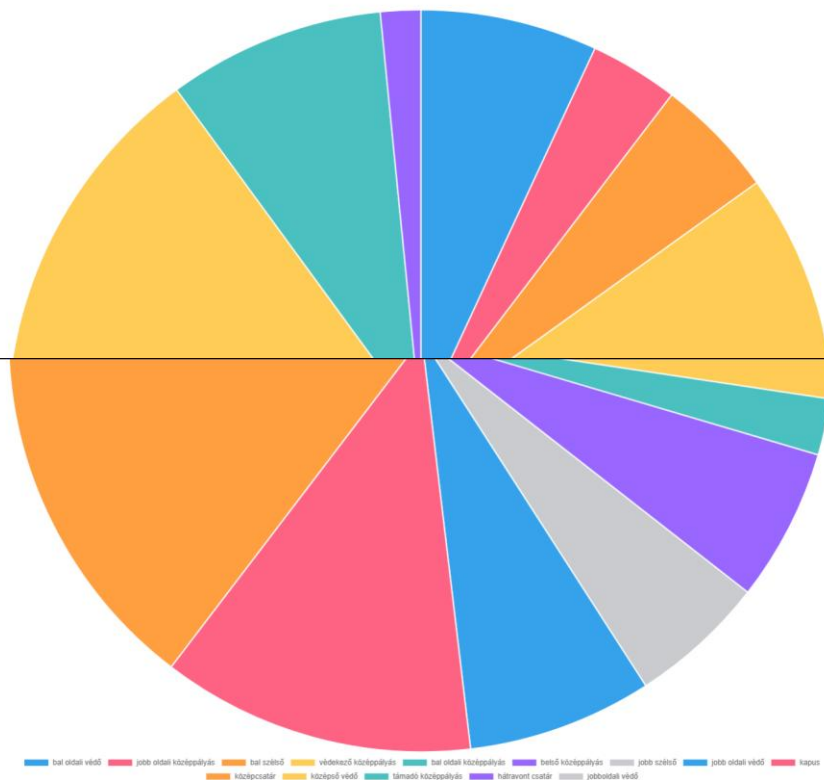
A Diagram menüpont három komponens együttműködésén alapul:

1. **Controller** – a diagram oldal megjelenítése (/diagram)
2. **REST API** – a diagram adatok előállítása JSON formátumban (/api/diagramdata)
3. **Frontend JavaScript** – Chart.js meghívása és adatbetöltés AJAX segítségével (piechart.js)

```
10
11 @Controller @S Daniel0210
12 public class DiagramController {
13
14     private final PosztRepository posztRepository; // 2 usages
15
16     public DiagramController(PosztRepository posztRepository) { @S Daniel0210
17         this.posztRepository = posztRepository;
18     }
19
20     // DIAGRAM OLDAL
21     @GetMapping("/{diagram}") @S Daniel0210
22     public String diagram() {
23         return "diagram";
24     }
25
26     // JSON ADATOK
27     @GetMapping("/api/diagramdata") @S Daniel0210
28     @ResponseBody
29     public ResponseEntity<Map<String, Double>> getDiagramData() {
30
31         var rows = posztRepository.countPlayersByPoszt(); // saját JPQL
32
33         double total = rows.stream().mapToDouble((Object[] r) -> (Long) r[1]).sum();
34
35         Map<String, Double> result = new LinkedHashMap<>();
36
37         for (Object[] r : rows) {
38             String nev = (String) r[0];
39             Long db = (Long) r[1];
40
41             double szazalek = (total == 0)
42                 ? 0
43                 : Math.round((db / total) * 10000.0) / 100.0;
44
45             result.put(nev, szazalek);
46         }
47
48         return ResponseEntity.ok(result);
49     }
50 }
51
52
53
```

CHARTJS

### JÁTEKOSOK POZÍCIÓ SZERINTI MEGOSZTLÁSA (%)



KÉSZÍTŐK   Falusi Tamás Rajmund (IVXW8X)  
Samu Dániel (NN8T44)

#### A végeredmény vizuálisan:

- modern, letisztult pie chart
- animált betöltés
- hover tooltip százalékkal
- automatikusan generált színek Chart.js által

## 8. CRUD menü

### Cél

A **CRUD** menü feladata, hogy egy külön, egyszerűbb táblában ( crud\_labdarugok ) lehessen saját játékosokat rögzíteni, majd ezeket **listázni, módosítani és törölni**. Ez a rész teljesíti a beadandó „CRUD alkalmazás” követelményét:

- **Create** – új játékos felvitele
- **Read** – meglévő rekordok listázása
- **Update** – rekord módosítása
- **Delete** – rekord törlése

A „Új játékos hozzáadása” link egy felugró ablakot (overlay) nyit meg. Ebben szerepel az összes szerkeszthető mező (név, mezszám, poszt, csapat, kor, érték).

A felhasználó kitölti a mezőket, majd a „Mentés” gombra kattintva az alkalmazás egy új rekordot hoz létre a tábla számára.

A mentés után automatikusan visszatér a CRUD oldalra, ahol a frissített lista már az új játékost is tartalmazza.

A táblázat minden sorában található egy **„Szerkesztés” gomb**. Erre kattintva ugyanaz az overlay jelenik meg, mint új felvitelkor, de már előre kitöltve az adott rekord értékeivel.

A felhasználó itt módosíthatja:

- a nevet,
- mezszámot,
- posztot,
- csapatot,
- életkort,
- értéket.

Mentés után az alkalmazás frissíti az adott rekordot az adatbázisban, és visszairányít a CRUD listára.

Az overlay így egyszerre szolgál **Create** és **Update** célokra.

## Rekord törlése

A táblázat utolsó oszlopában egy „Törlés” gomb található.

A működése:

1. A felhasználó rákattint.
2. Az alkalmazás törli az adott id-jú rekordot az adatbázisból.
3. A felhasználót visszairányítja a CRUD oldalra.

A törlés véglegesen eltávolítja a rekordot.

FŐOLDAL

ADATBÁZIS

KAPCSOLAT

DIAGRAM

CRUD

BEJELENTKEZÉS

REGISZTRÁCIÓ

ÚJ JÁTEKOS HOZZÁADÁSA

HOZZÁADOTT JÁTEKOSOK TÁBLÁZATA

MEZSZÁM	NÉV	KLUB	POSZT	KOR	ÉRTÉK	SZERKESZTÉS	TÖRLÉS
14	Nagy Péter	Fradí	Középpályás	27	1500 ezer €	SZERKESZTÉS	TÖRLÉS
10	Teszt Játékos	Teszt FC	Csatár	24	1000 ezer €	SZERKESZTÉS	TÖRLÉS
31	Kovács Ákos	Ák	Kapus	30	2000 ezer €	SZERKESZTÉS	TÖRLÉS

## 9. RESTful menü

A RESTful menü célja, hogy az alkalmazás külön API végpontokon keresztül is elérhető legyen.

Ez azt jelenti, hogy a CRUD műveletek (lekérés, hozzáadás, módosítás, törlés) nemcsak a webes felületen keresztül működnek, hanem **külső programokból, curl-lel vagy Postmanből is tesztelhetők.**

A REST API a crud\_labdarugok táblához készült, és az alábbi URL alatt érhető el:  
/api/crud

### 1) Összes rekord lekérése – GET

curl http://localhost:8080/api/crud

### 2) Egy rekord lekérése ID alapján – GET /{id}

curl http://localhost:8080/api/crud/7

### 3) Új rekord felvétele – POST

```
curl -X POST -H "Content-Type: application/json" ^  
-d '{"nev":"Teszt Játékos","mezzam":10,"poszt":"Csatár","csapat":"Teszt  
FC","ertek":1000,"kor":24}' ^  
http://localhost:8080/api/crud
```

### 4) Rekord módosítása – PUT /{id}

```
curl -X PUT -H "Content-Type: application/json" ^  
-d '{"nev":"Módosított  
Név","mezzam":99,"poszt":"Középpályás","csapat":"Új  
Csapat","ertek":3000,"kor":25}' ^  
http://localhost:8080/api/crud/7
```

### 5) Rekord törlése – DELETE /{id}

curl -X DELETE http://localhost:8080/api/crud/8

Az ID értékek csak példa jellegűek, ha nem létező ID-t használunk nem fog működni.

## Példák:

```
Olvashatóra formázás
{
  "createdAt": "2025-11-25T19:15:45",
  "csapat": "Új Csapat",
  "ertek": 1000,
  "id": 7,
  "kor": 25,
  "mezzam": 99,
  "nev": "Módosított Név",
  "poszt": "Középpályás",
  "updatedAt": "2025-11-25T19:15:45"
},
{
  "createdAt": "2025-11-25T19:31:10",
  "csapat": "Teszt FC",
  "ertek": 1000,
  "id": 9,
  "kor": 24,
  "mezzam": 10,
  "nev": "Teszt Játékos",
  "poszt": "Csatár",
  "updatedAt": "2025-11-25T19:31:10"
},
{
  "createdAt": "2025-11-25T19:35:07",
  "csapat": "Fradi",
  "ertek": 1500,
  "id": 10,
  "kor": 27,
  "mezzam": 14,
  "nev": "Nagy Péter",
  "poszt": "Középpályás",
  "updatedAt": "2025-11-25T19:35:07"
},
{
  "createdAt": "2025-11-27T18:09:43",
  "csapat": "Teszt FC",
  "ertek": 1000,
  "id": 11,
  "kor": 24,
  "mezzam": 10,
  "nev": "Teszt Játékos",
  "poszt": "Csatár",
  "updatedAt": "2025-11-27T18:09:43"
}
}
```

Microsoft Windows [Version 10.0.26200.7171]

(c) Microsoft Corporation. Minden jog fenntartva.

```
C:\Users\Samyb>curl -X POST -H "Content-Type: application/json" ^
More? -d '{"nev":"Teszt Játékos","mezzam":10,"poszt":"Csatár","csapat":"Teszt FC","ertek":1000,"kor":24}' ^
More? http://localhost:8080/api/crud
{"createdAt":"2025-11-27T18:09:43.701171","csapat":"Teszt FC","ertek":1000,"id":11,"kor":24,"mezzam":10,"nev":"Teszt Játékos","poszt":"Csatár","updatedAt":"2025-11-27T18:09:43.701171"}
C:\Users\Samyb>
```

## 10. Admin menü

### Cél

Az üzenetek kezelése. Minden felhasználó üzenetét megkapja és egyesével tudja törölni is. A megvalósítás hasonló, mint az üzenetek oldal. Az oldalon szintűgy idősorrendbe van lekérve az üzenetek és az admin felületen az üzenetek oldalhoz képest a felhasználó neve is megjelenik, hogy beazonosítható legyenek az üzenetek.

A megvalósításért az **UzenetController** és az **UzenetSerive** felel.

The screenshot displays the 'ELKÜLDÖTT ÜZENETEK' (Sent Messages) section of an admin interface. The top navigation bar includes links for FŐOLDAL, ADATBÁZIS, KAPCSOLAT, ÜZENETEK, DIAGRAM, CRUD, ADMIN, and a KIJELENTKEZÉS button. The main content area is divided into four cards, each representing a sent message. Each card shows a timestamp, a sender name, a subject, a brief description, and a 'TÖRLÉS' (Delete) button.

Timestamp	Sender	Subject	Description	Action
2025.11.30 15:47	TESZTELŐ BÉLA	TESZTELÉS	Ez egy teszt üzenet egy másik felhasználótól	TÖRLÉS
2025.11.30 15:30	TOMI	TESZT2	Még egy teszt üzenet	TÖRLÉS
2025.11.30 15:29	TOMI	TESZT1	Ez egy teszt üzenet	TÖRLÉS
2025.11.28 20:23	KOVÁCS BÉLA	ASDASRF	safdfhdfhd	TÖRLÉS



```

11 @Service 2 usages 2 FanTomi2 *
12 public class UzenetService {
13     @Autowired private UzenetRepository UzenetRepo;
14
15     public void save(Uzenet uzenet) { 2 FanTomi2
16         UzenetRepo.save(uzenet);
17     }
18
19     public List<Uzenet> getUserUzenetek(User user) { 1 usage 2 FanTomi2
20         return UzenetRepo.findByUserOrderByLetrehozasc(user);
21     }
22
23     public List<Uzenet> getAllUzenetek() { 1 usage 2 FanTomi2
24         return UzenetRepo.findAllByOrderByLetrehozasc();
25     }
26
27     public Optional<Uzenet> findById(int id) { 1 usage 2 FanTomi2
28         return UzenetRepo.findById(id);
29     }
30
31
32     public void delete(int id) { 2 usages 2 FanTomi2
33         UzenetRepo.deleteById(id);
34     }
35 }

```

```

57 @GetMapping("/admin") 2 FanTomi2
58 public String adminUzenetek(Model model) {
59     model.addAttribute("uzenetek", uzenetService.getAllUzenetek());
60     return "admin";
61 }
62
63 @PostMapping("/admin/{id}/delete") 2 FanTomi2
64 public String adminDelete(@PathVariable int id) {
65     uzenetService.delete(id);
66     return "redirect:/admin?torolve";
67 }
68 }
69

```

```

40 <div id="main">
41     <h2>Elküldött üzenetek</h2>
42
43     <section class="posts">
44         <article th:each="uzenet : ${uzenetek}">
45             <header>
46                 <span class="date"
47                     th:text="${#temporals.format(uzenet.letrehozasc, 'yyyy.MM.dd HH:mm')}">
48                 </span>
49                 <h4 th:text="${uzenet.user.nev}"></h4>
50                 <h2 th:text="${uzenet.cim}"></h2>
51             </header>
52
53             <p th:text="${uzenet.uzenet}"></p>
54
55             <form th:action="@{admin/{id}/delete(id=${uzenet.id})}"
56                 method="post"
57                 onsubmit="return confirm('Biztosan törölni szeretnéd?')">
58                 <button type="submit" class="btn">Törlés</button>
59             </form>
60         </article>
61     </section>
62 </div>

```