Java alkalmazások – Gyakorlat - Beadandó feladat -Dokumentáció

Beadandót készítette:
Borbély Bettina -mx4hyq
Slemmer Kevin - lzjiaq

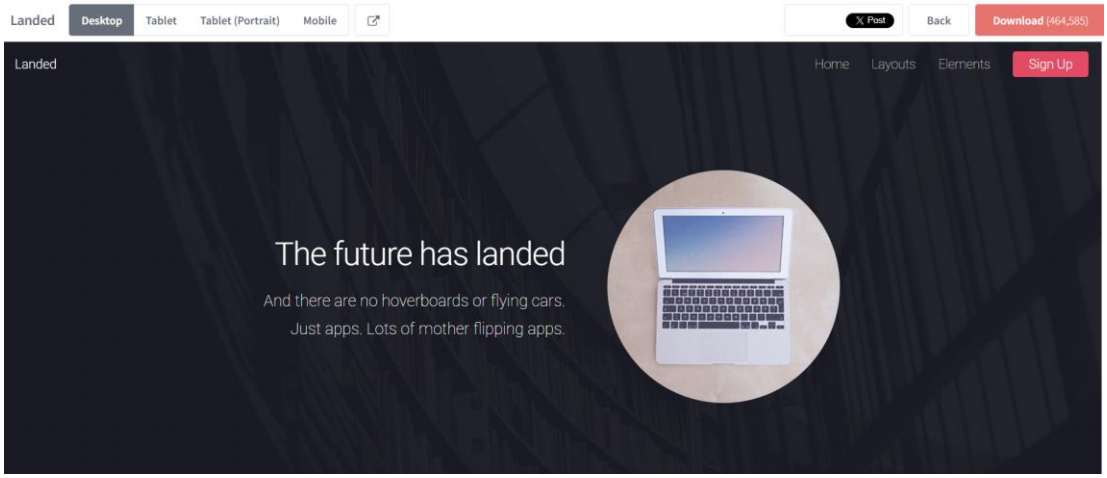
Feladatmegosztás:
Bettina: téma, kapcsolat menü, üzenetek menü, diagram menü, crud menü
Kevin: Autentikáció, főoldal menü, adatbázis menü, admin menü

Adatbázis: Városok

Weboldal: http://rivendell.nje.hu:9443/MX4HYQ-LZJIAQ-03/

1. Téma:

A weboldalak elkészítéséhez egy ingyenesen elérhető, modern és teljes mértékben reszponzív sablont választottunk. A feladathoz az HTML5 UP - Landed nevű témáját használtuk.
A téma elérhetősége: https://html5up.net/landed



2. Autentikáció:

Az alkalmazásban teljes autentikációs rendszer működik, amely a regisztrációt, bejelentkezést és kijelentkezést is tartalmazza. A felhasználók szerepkör alapján különböző jogosultságokat kapnak.

com.example.demo/security/SecurityConfig

```
@Configuration
public class SecurityConfig {

    private final UserRepository userRepository;  2 usages

    public SecurityConfig(UserRepository userRepository) { this.userRepository = userRepository; }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
                .authorizeHttpRequests( AuthorizationManagerRequestMat... auth -> auth
                        // Nyilvános oldalak
                        .requestMatchers("/", "/index", "/home", "/about").permitAll()
                        .requestMatchers("/contact").permitAll()    // csak a kapcsolat form nyilvános
                        .requestMatchers("/css/**", "/js/**", "/images/**", "/fonts/**").permitAll()
                        .requestMatchers("/register", "/login").permitAll()

                        // CSAK bejelentkezett felhasználó
                        .requestMatchers("/contact/all").hasAnyRole( ...roles: "REGISTERED", "ADMIN")
                        .requestMatchers("/uzenetek/**").hasAnyRole( ...roles: "REGISTERED", "ADMIN")

                        // CSAK admin
                        .requestMatchers("/admin/**").hasRole("ADMIN")
                       // minden más: szabadon marad
                        .anyRequest().permitAll()
                )
                .formLogin( FormLoginConfigurer<HttpSecurity> form -> form
                        .loginPage("/login")
                        .defaultSuccessUrl( defaultSuccessUrl: "/",  alwaysUse: true)
                        .permitAll()
                )
                .logout( LogoutConfigurer<HttpSecurity> logout -> logout
                        .logoutUrl("/logout")
                        .logoutSuccessUrl("/?logout")
                        .invalidateHttpSession(true)
                        .deleteCookies( ...cookieNamesToClear: "JSESSIONID")
                        .permitAll()
                )
                .exceptionHandling( ExceptionHandlingConfigurer<HttpSecurity> ex -> ex.accessDeniedHandler(accessDeniedHandler()));
```

```
        return http.build();
    }

    @Bean
    public AccessDeniedHandler accessDeniedHandler() {
        return ( HttpServletRequest request,  HttpServletResponse response,  AccessDeniedException accessDeniedException) ->
                response.sendRedirect( s: "/");
    }

    @Bean
    public UserDetailsService userDetailsService() {
        return  String email -> userRepository.findByEmail(email) Optional<User>
                .map( User user -> {
                    String role = user.getRole().name().toUpperCase();  // REGISTERED vagy ADMIN
                    return new org.springframework.security.core.userdetails.User(
                            user.getEmail(),
                            user.getPassword(),
                            Collections.singletonList(new SimpleGrantedAuthority( role: "ROLE_" + role))
                    );
                }) Optional<User>
                .orElseThrow(() -> new UsernameNotFoundException("Felhasználó nem található: " + email));
    }


    @Bean
    public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }
    @Bean
    public org.thymeleaf.extras.springsecurity6.dialect.SpringSecurityDialect springSecurityDialect() {
        return new org.thymeleaf.extras.springsecurity6.dialect.SpringSecurityDialect();
    }

}
```

dom.example.demo/repository/UserRepository

```
package com.example.demo.repository;

import com.example.demo.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.Optional;

public interface UserRepository extends JpaRepository<User, Long> {  7 usages
    Optional<User> findByEmail(String email);  2 usages
}
```

src/main/java/com/example/demo/entity/User.java

```
@Entity
@Table(name = "users")
@Data
public class User {

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

    @NotBlank(message = "Név megadása kötelező")
    private String name;

    @NotBlank(message = "Email megadása kötelező")
    @Email(message = "Érvénytelen email cím")
    @Column(unique = true)
    private String email;

    @NotBlank(message = "Jelszó megadása kötelező")
    @Size(min = 6, message = "Jelszó legalább 6 karakter")
    private String password;

    @Enumerated(EnumType.STRING)
    private Role role = Role.registered;

    private LocalDateTime emailVerifiedAt;
    private String rememberToken;

    @Column(name = "created_at")
    private LocalDateTime createdAt;

    @Column(name = "updated_at")
    private LocalDateTime updatedAt;

    public enum Role {  3 usages
        registered, admin
    }
```

A **User** entitás egy **Role** enumot tartalmaz, amely meghatározza a felhasználók jogosultságait:

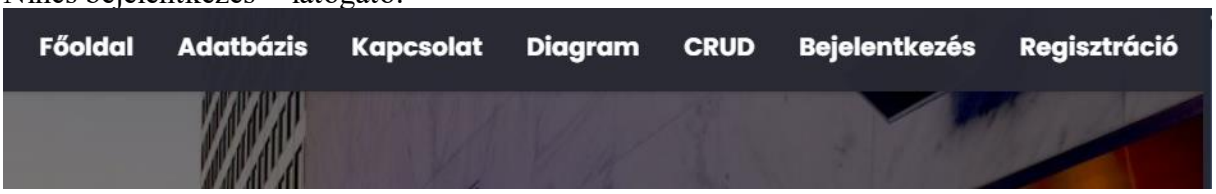
- **registered -** regisztrált látogató
- **admin -** adminisztrátor

A nem bejelentkezett felhasználók automatikusan a „látogató” szerepkörbe tartoznak.
A **UserRepository** az adatbázisból tölti be a felhasználókat e-mail cím alapján.
A jelszavak **BCrypt** titkosítással kerülnek mentésre.
A **SecurityConfig** egy saját **UserDetailsService**-t használ, amely az adatbázisban tárolt e-mail, jelszó és szerepkör alapján hozza létre a Spring Security által használt felhasználói objektumot.
A biztonsági konfiguráció URL-alapú védelemmel látja el az alkalmazást:

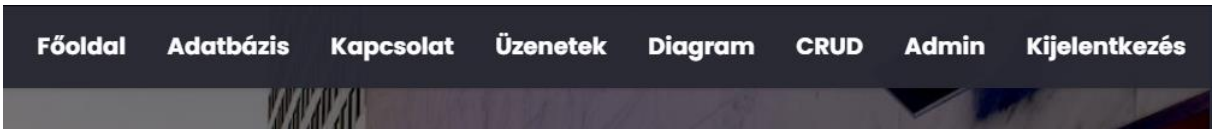
- **Nyilvános oldalak:** /, /home, /about, /contact, /login, /register
- **Regisztrált látogató** + admin: /contact/all, /uzenetek/**
- **Csak admin**: /admin/**

A rendszer egyedi bejelentkezési oldalt használ **(/login)**.
Sikeres bejelentkezés után a főoldalra irányítja a felhasználót.
A kijelentkezés a **/logout** végponton történik, amely törli a sessiont és a cookie-kat.

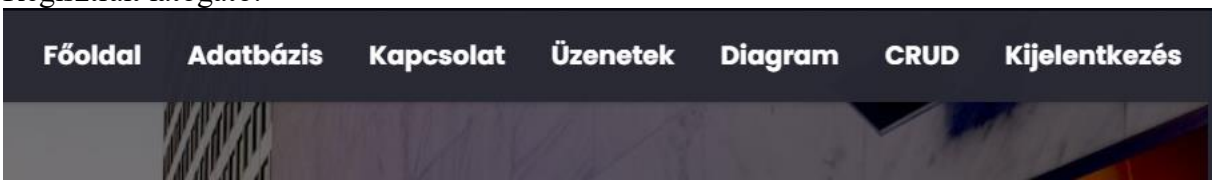
Nincs bejelentkezés – látogató:



Admin:



Regisztrált látogató:

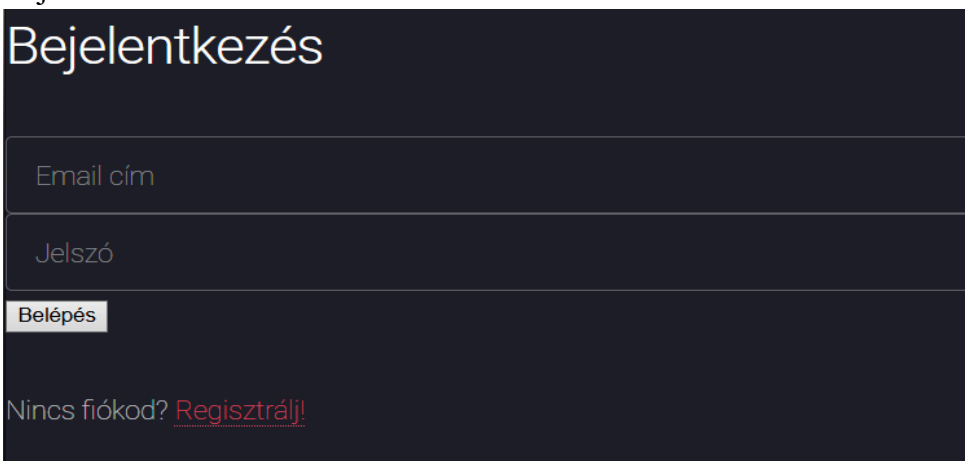


src/main/java/com/example/demo/controller/AuthController.java

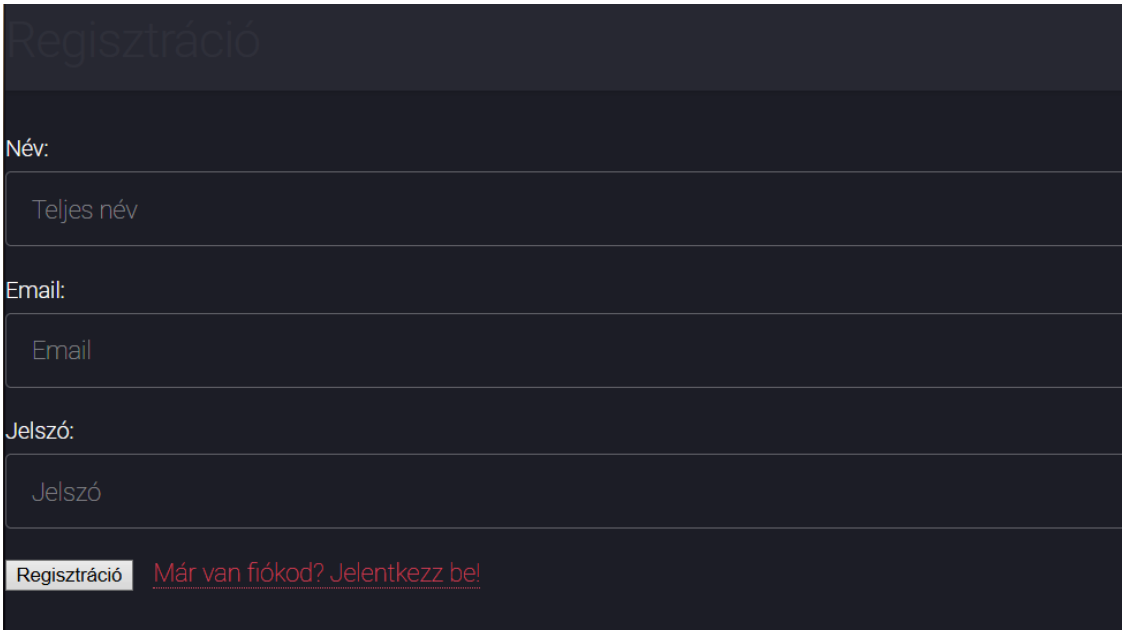
```java
@Controller
public class AuthController {

    @Autowired
    private UserRepository userRepo;

    @Autowired
    private PasswordEncoder passwordEncoder;

    @GetMapping("/login")
    public String login() { return "auth/login"; }

    @GetMapping("/register")
    public String showRegisterForm(Model model) {
        model.addAttribute( attributeName: "user", new User());
        return "auth/register";
    }

    @PostMapping("/register")
    public String register(@Valid @ModelAttribute("user") User user,
                           BindingResult result,
                           Model model) {

        if (result.hasErrors()) {
            return "auth/register";
        }

        if (userRepo.findByEmail(user.getEmail()).isPresent()) {
            model.addAttribute( attributeName: "error", attributeValue: "Ez az e-mail cím már foglalt!");
            return "auth/register";
        }

        user.setPassword(passwordEncoder.encode(user.getPassword()));
        user.setRole(User.Role.registered);

        userRepo.save(user); // mentés adatbázisba

        return "redirect:/login?registered";
    }
}
```

Az **AuthController** kezeli a felhasználói hitelesítéssel kapcsolatos funkciókat, mint a bejelentkezés és regisztráció.A **/login** végpont visszaadja a bejelentkezési oldalt. A **/register GET** végpont egy új regisztrációs űrlapot jelenít meg.A **/register POST** végpont ellenőrzi a felhasználó adatait, lekezeli a hibákat, ellenőrzi, hogy az e-mail cím már foglalt-e, és ha minden rendben van, a jelszót **titkosítja** (**PasswordEncoder**) és elmenti az adatbázisba.
A sikeres regisztráció után a felhasználó a bejelentkezési oldalra kerül átirányításra.

Bejelentkezési felület:



Regisztrációs felület:



3. Főoldal menü

A főoldal a HorizonTours cég bemutatkozó oldala, látványos hero szekcióval, szlogen és akciógombok megjelenítésével.
A legnépszerűbb úti célok kártyákon, képekkel, rövid leírással és árakkal láthatók.
A menü dinamikusan változik a felhasználó szerepkörétől függően: látogatók, regisztrált felhasználók és adminok különböző menüpontokat látnak.

src/main/java/com/example/demo/controller/HomeController.java

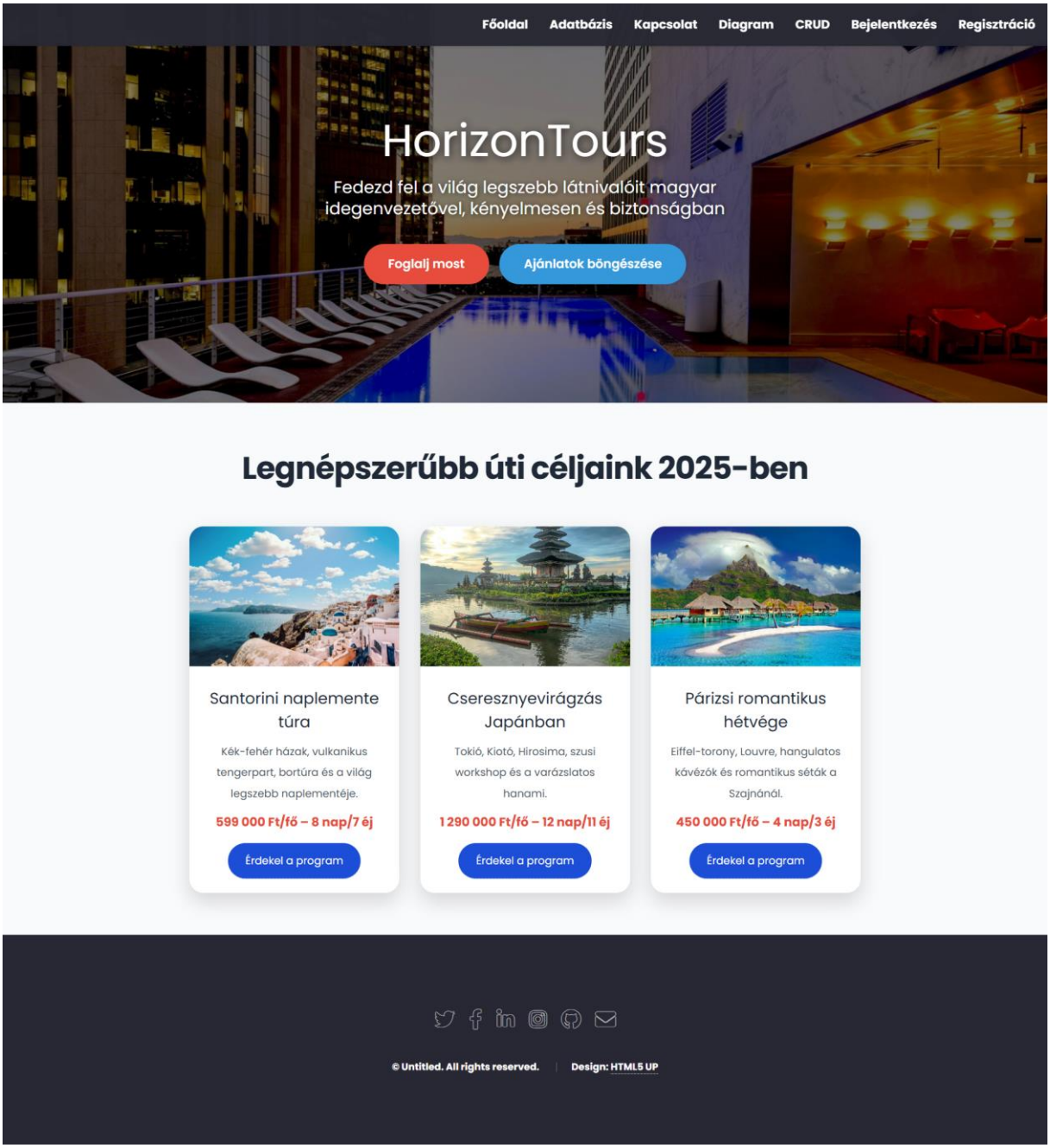
```java
@Controller
public class HomeController {

    @GetMapping("/")
    public String home(Model model) {
        model.addAttribute( attributeName: "companyName",  attributeValue: "HorizonTours");
        model.addAttribute( attributeName: "slogan",  attributeValue: "Fedezd fel a világ legszebb látnivalóit velünk!");
        return "fooldal/index"; // → templates/index.html
    }
}
```

src/main/resources/templates/index.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:sec="http://www.thymeleaf.org/extras/spring-security">

<head th:replace="fragments/head :: head('HorizonTours - Fedezd fel a világot')"></head>

<body>
<!-- Header fragment -->
<header th:replace="fragments/header :: header"></header>

<main>

    <!-- Hero szekció -->
    <section class="hero">
        <div class="hero-content">
            <h1>HorizonTours</h1>
            <p>Fedezd fel a világ legszebb látnivalóit magyar idegenvezetővel, kényelmesen és biztonságban</p>
            <div class="hero-buttons">
                <a href="#" class="btn primary">Foglalj most</a>
                <a href="#" class="btn secondary">Ajánlatok böngészése</a>
            </div>
        </div>
    </section>

    <!-- Legnépszerűbb úti célok -->
    <section class="section" id="destinations">
        <h2>Legnépszerűbb úti céljaink 2025-ben</h2>
        <div class="destinations-container">

            <div class="card">
                <img src="https://images.pexels.com/photos/1285625/pexels-photo-1285625.jpeg?auto=compress&cs=tinysrgb&w=1600" alt="Santorini">
                <div class="card-body">
                    <h3>Santorini naplemente túra</h3>
                    <p>Kék-fehér házak, vulkanikus tengerpart, bortúra és a világ legszebb naplementéje.</p>
                    <span class="price">599 000 Ft/fő - 8 nap/7 éj</span>
                    <a href="#" class="btn card-btn">Érdekel a program</a>
                </div>
            </div>

            <div class="card">
                <img src="https://images.pexels.com/photos/2166559/pexels-photo-2166559.jpeg?auto=compress&cs=tinysrgb&w=1600" alt="Japán">
                <div class="card-body">
                    <h3>Cseresznyevirágzás Japánban</h3>
                    <p>Tokió, Kiotó, Hirosima, szusi workshop és a varázslatos hanami.</p>
                    <span class="price">1 290 000 Ft/fő - 12 nap/11 éj</span>
                    <a href="#" class="btn card-btn">Érdekel a program</a>
                </div>
            </div>

            <div class="card">
                <img src="https://images.pexels.com/photos/753626/pexels-photo-753626.jpeg?auto=compress&cs=tinysrgb&w=1600" alt="Párizs">
                <div class="card-body">
                    <h3>Párizsi romantikus hétvége</h3>
                    <p>Eiffel-torony, Louvre, hangulatos kávézók és romantikus séták a Szajnánál.</p>
                    <span class="price">450 000 Ft/fő - 4 nap/3 éj</span>
                    <a href="#" class="btn card-btn">Érdekel a program</a>
                </div>
            </div>

        </div>
    </section>

</main>

<!-- Footer fragment -->
<footer th:replace="fragments/footer :: footer"></footer>
```

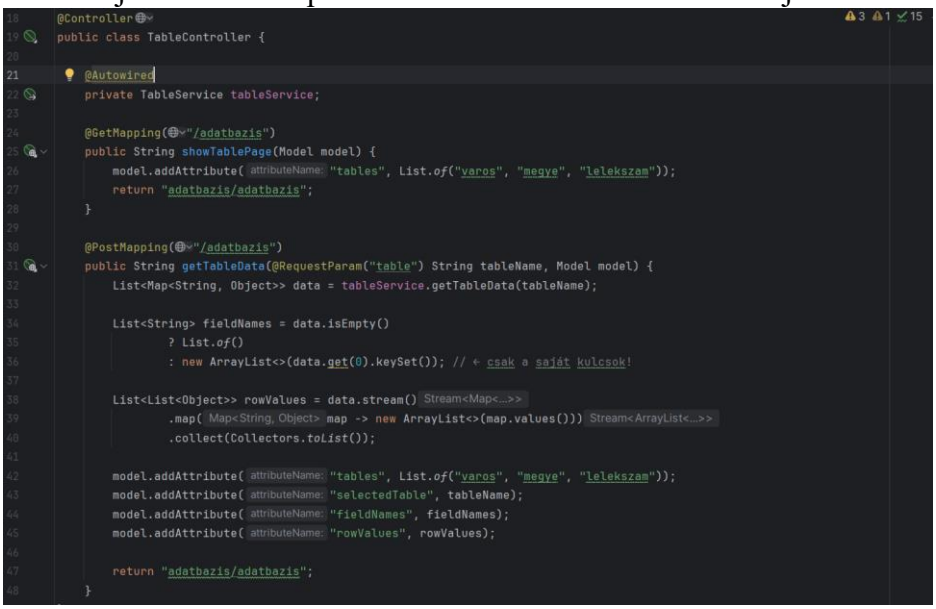
69. sortól stílusok

Ez a HomeController kezeli a főoldal megjelenítését a webalkalmazásban.
A / URL-re érkező kérések esetén a home() metódus fut le.
A metódus a modelhez hozzáadja a cégnév (companyName) és a szlogen (slogan) adatait.
Ezután visszaadja a fooldal/index sablont, amely a főoldal HTML-jét jeleníti meg.


Főoldal
Adatbázis
Kapcsolat
Diagram
CRUD
Bejelentkezés
Regisztráció
HorizonTours
Fedezd fel a világ legszebb látnivalóit magyar idegenvezetővel, kényelmesen és biztonságban
Foglalj most
Ajánlatok böngészése
Legnépszerűbb úti céljaink 2025-ben
Santorini naplemente túra
Kék-fehér házak, vulkanikus tengerpart, bortúra és a világ legszebb naplementéje.
599 000 Ft/fő – 8 nap/7 éj
Érdekel a program
Cseresznyevirágzás Japánban
Tokió, Kiotó, Hirosima, szusi workshop és a varázslatos hanami.
1 290 000 Ft/fő – 12 nap/11 éj
Érdekel a program
Párizsi romantikus hétvége
Eiffel-torony, Louvre, hangulatos kávézók és romantikus séták a Szajnánál.
450 000 Ft/fő – 4 nap/3 éj
Érdekel a program
© Untitled. All rights reserved.
Design: HTML5 UP

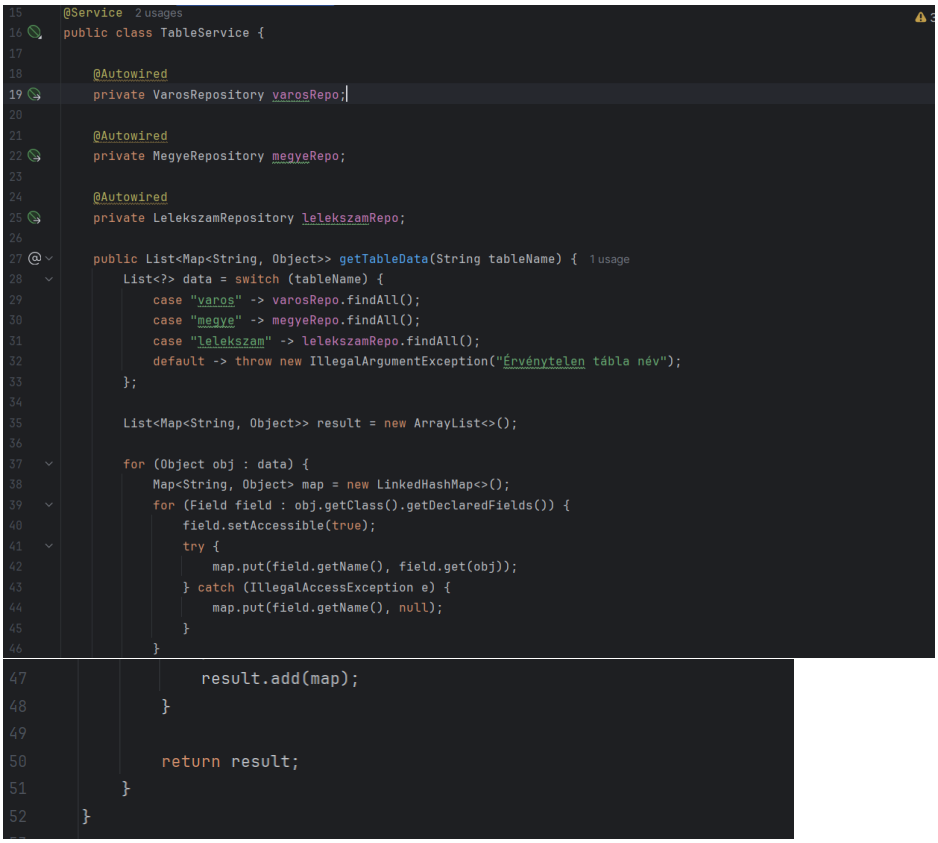
4. Adatbázis menü

src/main/java/com/example/demo/controller/TableController.java



```java
@Controller
public class TableController {

    @Autowired
    private TableService tableService;

    @GetMapping("/adatbazis")
    public String showTablePage(Model model) {
        model.addAttribute( attributeName: "tables", List.of("varos", "megye", "lelekszam"));
        return "adatbazis/adatbazis";
    }

    @PostMapping("/adatbazis")
    public String getTableData(@RequestParam("table") String tableName, Model model) {
        List<Map<String, Object>> data = tableService.getTableData(tableName);

        List<String> fieldNames = data.isEmpty()
                ? List.of()
                : new ArrayList<>(data.get(0).keySet()); // ← csak a saját kulcsok!

        List<List<Object>> rowValues = data.stream() Stream<Map<...>>
                .map( Map<String, Object> map -> new ArrayList<>(map.values())) Stream<ArrayList<...>>
                .collect(Collectors.toList());

        model.addAttribute( attributeName: "tables", List.of("varos", "megye", "lelekszam"));
        model.addAttribute( attributeName: "selectedTable", tableName);
        model.addAttribute( attributeName: "fieldNames", fieldNames);
        model.addAttribute( attributeName: "rowValues", rowValues);

        return "adatbazis/adatbazis";
    }
```

// src/main/java/com/example/demo/service/TableService.java

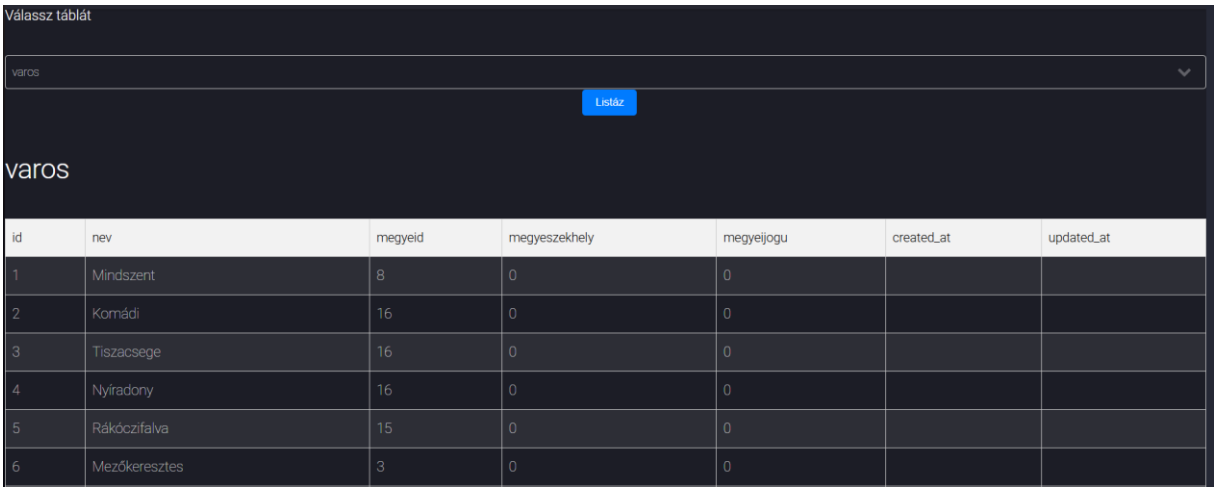


```java
@Service  2 usages
public class TableService {

    @Autowired
    private VarosRepository varosRepo;

    @Autowired
    private MegyeRepository megyeRepo;

    @Autowired
    private LelekszamRepository lelekszamRepo;

    public List<Map<String, Object>> getTableData(String tableName) {  1 usage
        List<?> data = switch (tableName) {
            case "varos" -> varosRepo.findAll();
            case "megye" -> megyeRepo.findAll();
            case "lelekszam" -> lelekszamRepo.findAll();
            default -> throw new IllegalArgumentException("Érvénytelen tábla név");
        };

        List<Map<String, Object>> result = new ArrayList<>();

        for (Object obj : data) {
            Map<String, Object> map = new LinkedHashMap<>();
            for (Field field : obj.getClass().getDeclaredFields()) {
                field.setAccessible(true);
                try {
                    map.put(field.getName(), field.get(obj));
                } catch (IllegalAccessException e) {
                    map.put(field.getName(), null);
                }
            }
```



```java
            result.add(map);
        }

        return result;
    }
}
```

A kód egy dinamikus adatbázis-megjelenítő felületet valósít meg, ahol a felhasználó kiválaszthatja, melyik tábla (varos, megye, lelekszam) adatait szeretné látni. A **TableController** kezeli az oldal megjelenítését és a kiválasztott tábla adatainak lekérését. A **TableService** a választott tábla összes rekordját lekéri az adatbázisból.



5. Kapcsolat menü:

Lehetővé teszi a felhasználók számára üzenetek küldését az oldal tulajdonosának. Az űrlap adatainak **szerver oldali validációja** biztosítja a helyes és biztonságos adatbevitelét, míg az **adatbázisba mentés** lehetővé teszi az üzenetek nyomon követését és későbbi feldolgozását.

src/main/java/com/example/demo/controller/ContactController.java

```java
@Controller
@RequestMapping("/contact")
public class ContactController {

    private final MessageService messageService;  3 usages

    public ContactController(MessageService messageService) {
        this.messageService = messageService;
    }

    // Kapcsolat űrlap megjelenítése
    @GetMapping
    public String showForm(Model model) {
        model.addAttribute( attributeName: "message", new Message());
        return "contact/form";
    }

    // Űrlap elküldése
    @PostMapping
    public String submitForm(@ModelAttribute Message message, Model model) {
        messageService.save(message);
        model.addAttribute( attributeName: "success",  attributeValue: "Üzenet sikeresen elküldve!");
        model.addAttribute( attributeName: "message", new Message()); // Ürítjük az űrlapot
        return "contact/form";
    }
     @GetMapping("/all")
     public String listMessages(Model model) {
         List<Message> messages = messageService.findAllMessagesOrdered();
         model.addAttribute( attributeName: "messages", messages);
         return "contact/list";
     }
```

src/main/java/com/example/demo/service /MessageService.java

```java
@Service  3 usages
public class MessageService {

    private final MessageRepository messageRepository;  4 usages

    public MessageService(MessageRepository messageRepository) {
        this.messageRepository = messageRepository;
    }

    public Message save(Message message) {
        return messageRepository.save(message);
    }

    public List<Message> findAll() {  no usages
        return messageRepository.findAll();
    }
    public List<Message> findAllMessagesOrdered() {  1 usage
        List<Message> messages = messageRepository.findAll();
        messages.sort(( Message m1,  Message m2) -> m2.getCreatedAt().compareTo(m1.getCreatedAt()));
        return messages;
    }
}
```

src/main/java/com/example/demo/repository /MessageRepository.java

```java
public interface MessageRepository extends JpaRepository<Message, Long> {  3 usages
    List<Message> findAllByOrderByCreatedAtDesc();  no usages
}
```

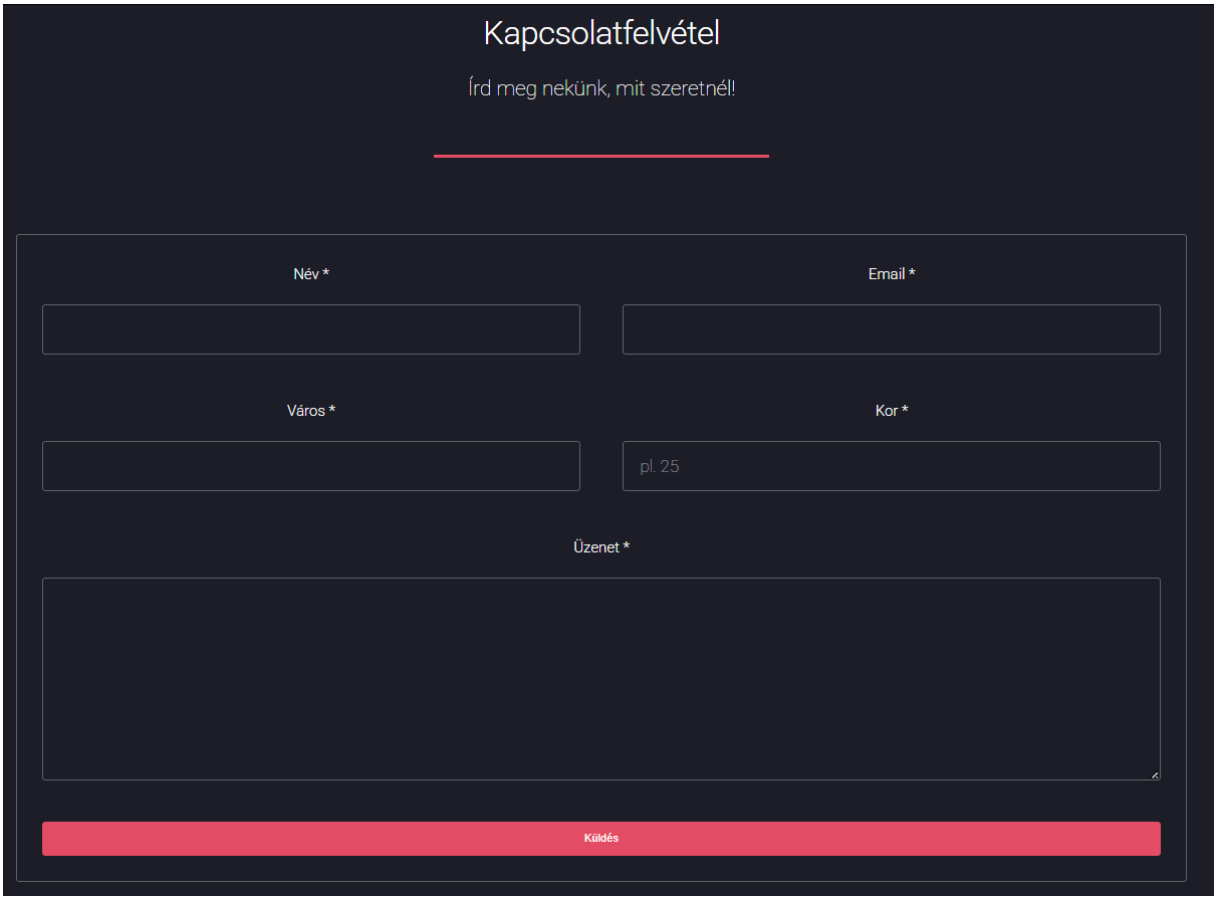
src/main/java/com/example/demo/model /Message.java

```java
@Entity
@Table(name = "uzenetek")
public class Message {

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

    private String nev;  2 usages

    private String email;  2 usages

    private String varos;  2 usages

    private String kor;  2 usages

    private String uzenet;  2 usages

    private LocalDateTime createdAt;  3 usages

    private LocalDateTime updatedAt;  2 usages

    // Konstruktor
    public Message() {
        this.createdAt = LocalDateTime.now();
    }

    // Getterek és setterek
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

    public String getNev() { return nev; }
    public void setNev(String nev) { this.nev = nev; }

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    public String getVaros() { return varos; }  no usages
    public void setVaros(String varos) { this.varos = varos; }  no usages

    public String getKor() { return kor; }
    public void setKor(String kor) { this.kor = kor; }

    public String getUzenet() { return uzenet; }  no usages
    public void setUzenet(String uzenet) { this.uzenet = uzenet; }  no usages

    public LocalDateTime getCreatedAt() { return createdAt; }  2 usages
    public void setCreatedAt(LocalDateTime createdAt) { this.createdAt = createdAt; }  no usages

    public LocalDateTime getUpdatedAt() { return updatedAt; }  no usages
    public void setUpdatedAt(LocalDateTime updatedAt) { this.updatedAt = updatedAt; }  no usages

}
```

src/main/resources/templates/contact/form.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:sec="http://www.thymeleaf.org/extras/spring-security">

<head th:replace="~{fragments/head :: head('Kapcsolatfelvétel')}">
    <style>
    </style>
</head>

<body class="is-preload landing">
<div id="page-wrapper">

    <!-- HEADER (menü + login/logout) -->
    <th:block th:replace="~{fragments/header :: header}"></th:block>

    <!-- FŐ TARTALOM -->
    <section id="one" class="wrapper style1 special">
        <div class="container">
            <header class="major">
                <h2>Kapcsolatfelvétel</h2>
                <p>Írd meg nekünk, mit szeretnél!</p>
            </header>

            <!-- SIKERES ÜZENET -->
            <div th:if="${success}" class="success">
                <p th:text="${success}"></p>
            </div>

            <!-- ŰRLAP -->
            <div class="box">
                <form th:action="@{/contact}" th:object="${message}" method="post">
                    <div class="row gtr-uniform">

                        <div class="col-6 col-12-xsmall">
                            <label for="nev">Név *</label>
                            <input type="text" th:field="*{nev}" id="nev" required />
                        </div>

                        <div class="col-6 col-12-xsmall">
                            <label for="email">Email *</label>
                            <input type="email" th:field="*{email}" id="email" required />
                        </div>

                        <div class="col-6 col-12-xsmall">
                            <label for="varos">Város *</label>
                            <input type="text" th:field="*{varos}" id="varos" required />
                        </div>

                        <div class="col-6 col-12-xsmall">
                            <label for="kor">Kor *</label>
                            <input type="text" th:field="*{kor}" id="kor" placeholder="pl. 25" required />
                        </div>

                        <div class="col-12">
                            <label for="uzenet">Üzenet *</label>
                            <textarea th:field="*{uzenet}" id="uzenet" rows="6" required></textarea>
                        </div>

                        <div class="col-12">
                            <button type="submit" class="button primary fit">Küldés</button>
                        </div>

                    </div>
                </form>
            </div>
        </div>
    </section>

    <!-- FOOTER -->
    <th:block th:replace="~{fragments/footer :: footer}"></th:block>
```

A **ContactController** kezeli az űrlap megjelenítését, az üzenetek elküldését és az összes üzenet listázását. Az üzeneteket a **Message** entitás tárolja az adatbázisban, beleértve a felhasználó nevét, email címét, városát, korát és az üzenet tartalmát. A **MessageService** és a **MessageRepository** biztosítják az üzenetek mentését és lekérdezését, valamint az időrendi sorrend szerinti megjelenítést. A szerver oldali validáció bevezetésével az űrlapadatok biztonságosan ellenőrizhetők és menthetők az adatbázisba.



6. Üzenetek menü

A felhasználók táblázatban tekinthetik meg az adatbázisba mentett üzeneteket, fordított időrendben, a legfrissebb üzenet elöl. Minden üzenet mellett megjelenik a küldés időpontja is. A menü csak bejelentkezett felhasználók számára érhető el, így az üzenetek biztonságosan kezelhetők.

src/main/java/com/example/demo/controller/ContactController.java

```
37      @GetMapping("/all")
38      public String listMessages(Model model) {
39          List<Message> messages = messageService.findAllMessagesOrdered();
40          model.addAttribute( attributeName: "messages", messages);
41          return "contact/list";
42      }
```

src/main/java/com/example/demo/repository/MessageRepository.java

```java
public interface MessageRepository extends JpaRepository<Message, Long> {  3 usages
    List<Message> findAllByOrderByCreatedAtDesc();  no usages
}
```

src/main/resources/templates/contact/list.html

```html
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:sec="http://www.thymeleaf.org/extras/spring-security">

<head th:replace="~{fragments/head :: head('Beérkezett üzenetek')}">
    <style>
    </style>
</head>

<body class="is-preload landing">
<div id="page-wrapper">

    <th:block th:replace="~{fragments/header :: header}"></th:block>

    <div class="card">
        <div class="card-header">
            <h1>Beérkezett üzenetek</h1>
        </div>

        <div th:if="${messages == null or messages.isEmpty()}" class="empty">
            Még nincs beérkezett üzenet
        </div>

        <div style="..." th:if="${messages != null and !messages.isEmpty()}">
            <table>
                <thead>
                <tr>
                  <th>Név</th>
                  <th>Email</th>
                  <th>Város</th>
                  <th>Kor</th>
                  <th>Üzenet</th>
                  <th class="timestamp">Elküldve</th>
                </tr>
                </thead>
                <tbody>
                <tr th:each="m : ${messages}">
                    <td th:text="${m.nev}">Név</td>
                    <td th:text="${m.email}">email@pelda.hu</td>
                    <td th:text="${m.varos}">Város</td>
                    <td th:text="${m.kor}">18</td>
                    <td th:text="${m.uzenet}">Üzenet...</td>
                    <td class="timestamp" th:text="${#temporals.format(m.createdAt, 'yyyy.MM.dd HH:mm')}">
                        2025.12.08 14:20
                    </td>
                </tr>
                </tbody>
            </table>
        </div>
    </div>

    <th:block th:replace="~{fragments/footer :: footer}"></th:block>

</div>
</body>
</html>
```

A ContactController GET **/contact/all** útvonalán lekéri az összes üzenetet az adatbázisból a **MessageService** segítségével, időrend szerint (legfrissebb elöl), és a **contact/list** nézetnek továbbítja. A **Message** entitás tárolja a felhasználó nevét, email címét, városát, korát, az üzenet tartalmát, valamint a létrehozás és frissítés idejét. Egy táblázatban jeleníti meg az üzeneteket, minden sorhoz kiírva az üzenet küldésének időpontját. Ha még nincs üzenet, a felhasználó egy „Még nincs beérkezett üzenet” üzenetet lát.

Név	Email	Város	Kor	Üzenet	Elküldve
Teszt	tesztelek@gmail.com	Iregszemcse	21	Teszt betti felé	2025.12.08 01:49

7. Diagram menü

A diagram a magyarországi megyékhez tartozó városok számát ábrázolja.

src/main/java/com/example/demo/controller/ChartController.java

```java
public class ChartController {

    private final CountyRepository countyRepository;  2 usages
    private final CityRepository cityRepository;  2 usages

    public ChartController(CountyRepository countyRepository, CityRepository cityRepository) {
        this.countyRepository = countyRepository;
        this.cityRepository = cityRepository;
    }

    @GetMapping("/chart")
    public String chart(Model model) {

        List<County> counties = countyRepository.findAll();

        List<CountyData> countyDataList = counties.stream() Stream<County>
                .map( County c -> new CountyData(c.getNev(), cityRepository.findAllByMegye_Id(c.getId().intValue()).size())) Strea
                .sorted(( CountyData a,  CountyData b) -> Integer.compare(b.getCount(), a.getCount())) // csökkenő sorrend
                .toList();

        List<String> labels = countyDataList.stream() Stream<CountyData>
                .map(CountyData::getName) Stream<String>
                .toList();

        List<Integer> data = countyDataList.stream() Stream<CountyData>
                .map(CountyData::getCount) Stream<Integer>
                .toList();

        model.addAttribute( attributeName: "labels", labels);
        model.addAttribute( attributeName: "data", data);

        return "diagram/cities";
    }

    // Segédosztály a párosításra
    public static class CountyData {  4 usages
        private final String name;  2 usages
        private final int count;  2 usages

        public CountyData(String name, int count) {  1 usage
            this.name = name;
            this.count = count;
        }

        public String getName() {
            return name;
        }

        public int getCount() {  3 usages
            return count;
        }
    }

}
```

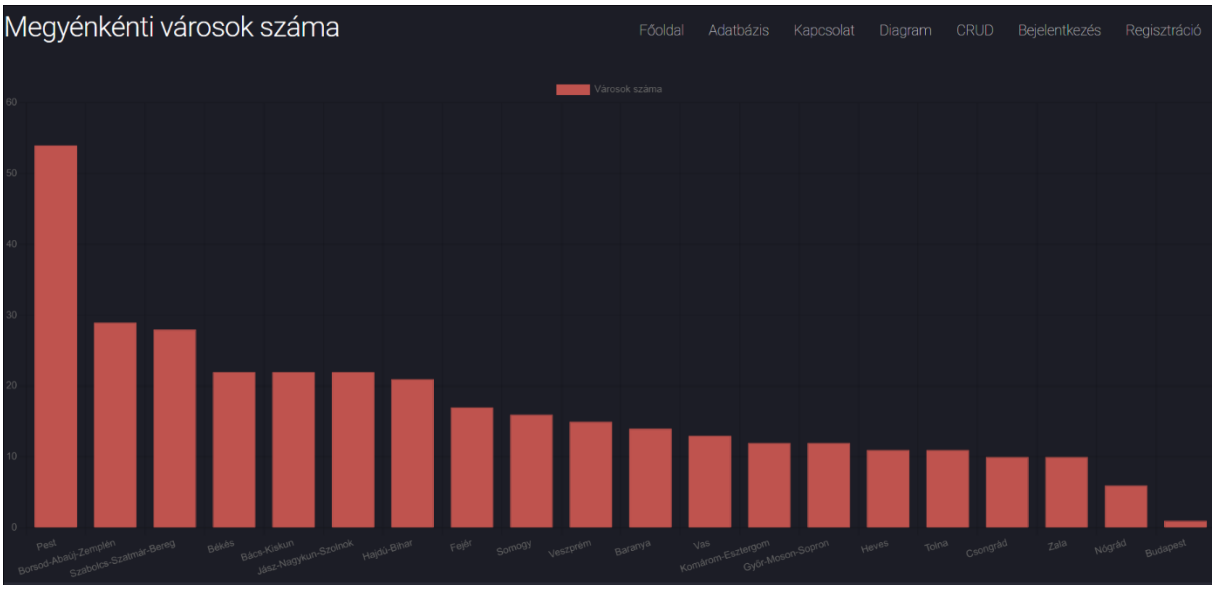
// src/main/java/com/example/demo/repository/CountyRepository.java

```java
import ...

public interface CountyRepository extends JpaRepository<County, Long> {  7 usages
    Long id(Long id);
}
```

// src/main/java/com/example/demo/repository/CityRepository.java

```java
import ...

@Repository  7 usages
public interface CityRepository extends JpaRepository<City, Long> {
    List<City> findAllByMegye_Id(int megyeId);  1 usage
}
```

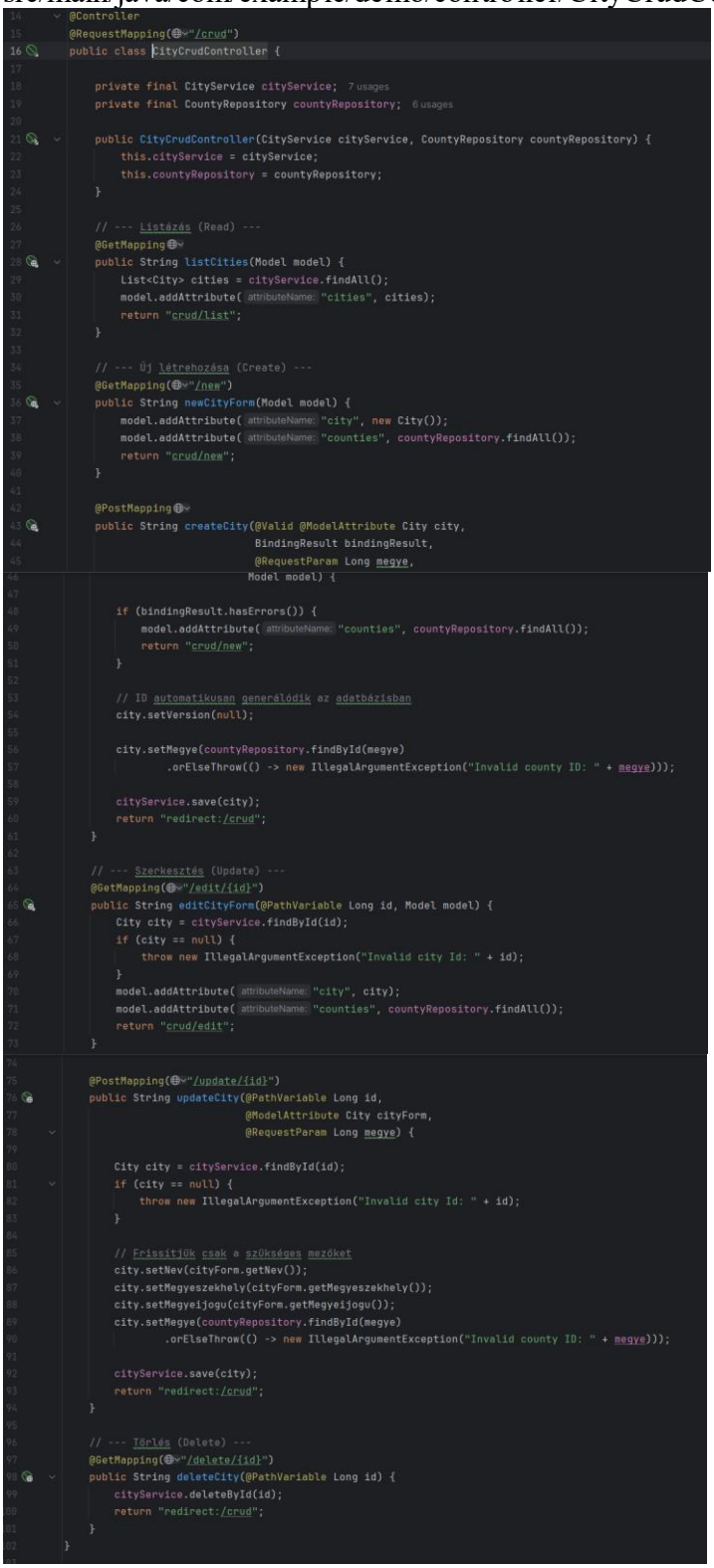
A vezérlő lekéri az összes megyét, majd mindegyikhez meghatározza, hány város tartozik hozzá. Az adatokat egy segédosztályba rendezi és a városok száma szerint csökkenő sorrendbe rakja. Ezután külön listába gyűjti a megyék neveit és a hozzájuk tartozó városszámokat, hogy diagramhoz felhasználhatók legyenek. A lekészített adatokat hozzáadja a modellhez, majd visszatér a „**diagram/cities**” nézethez, ahol a grafikon megjelenik. A **CountyRepository** egy JPA adattároló, a **CityRepository** pedig a megyékhez tartozó városok lekérdezését végzi.



8. CRUD menü

Az oldalon egy teljes CRUD funkciót valósítottunk meg az adatbázis város táblájához, amely lehetővé teszi az adatok megjelenítését, új rekordok felvitelét, valamint a meglévő rekordok módosítását és törlését.

src/main/java/com/example/demo/controller/CityCrudController.java



```java
@Controller
@RequestMapping("/crud")
public class CityCrudController {

    private final CityService cityService;  7 usages
    private final CountyRepository countyRepository;  6 usages

    public CityCrudController(CityService cityService, CountyRepository countyRepository) {
        this.cityService = cityService;
        this.countyRepository = countyRepository;
    }

    // --- Listázás (Read) ---
    @GetMapping
    public String listCities(Model model) {
        List<City> cities = cityService.findAll();
        model.addAttribute( attributeName: "cities", cities);
        return "crud/list";
    }

    // --- Új létrehozása (Create) ---
    @GetMapping("/new")
    public String newCityForm(Model model) {
        model.addAttribute( attributeName: "city", new City());
        model.addAttribute( attributeName: "counties", countyRepository.findAll());
        return "crud/new";
    }

    @PostMapping
    public String createCity(@Valid @ModelAttribute City city,
                             BindingResult bindingResult,
                             @RequestParam Long megye,
                             Model model) {

        if (bindingResult.hasErrors()) {
            model.addAttribute( attributeName: "counties", countyRepository.findAll());
            return "crud/new";
        }

        // ID automatikusan generálódik az adatbázisban
        city.setVersion(null);

        city.setMegye(countyRepository.findById(megye)
                .orElseThrow(() -> new IllegalArgumentException("Invalid county ID: " + megye)));

        cityService.save(city);
        return "redirect:/crud";
    }

    // --- Szerkesztés (Update) ---
    @GetMapping("/edit/{id}")
    public String editCityForm(@PathVariable Long id, Model model) {
        City city = cityService.findById(id);
        if (city == null) {
            throw new IllegalArgumentException("Invalid city Id: " + id);
        }
        model.addAttribute( attributeName: "city", city);
        model.addAttribute( attributeName: "counties", countyRepository.findAll());
        return "crud/edit";
    }

    @PostMapping("/update/{id}")
    public String updateCity(@PathVariable Long id,
                             @ModelAttribute City cityForm,
                             @RequestParam Long megye) {

        City city = cityService.findById(id);
        if (city == null) {
            throw new IllegalArgumentException("Invalid city Id: " + id);
        }

        // Frissítjük csak a szükséges mezőket
        city.setNev(cityForm.getNev());
        city.setMegyeszekhely(cityForm.getMegyeszekhely());
        city.setMegyeijogu(cityForm.getMegyeijogu());
        city.setMegye(countyRepository.findById(megye)
                .orElseThrow(() -> new IllegalArgumentException("Invalid county ID: " + megye)));

        cityService.save(city);
        return "redirect:/crud";
    }

    // --- Törlés (Delete) ---
    @GetMapping("/delete/{id}")
    public String deleteCity(@PathVariable Long id) {
        cityService.deleteById(id);
        return "redirect:/crud";
    }
}
```

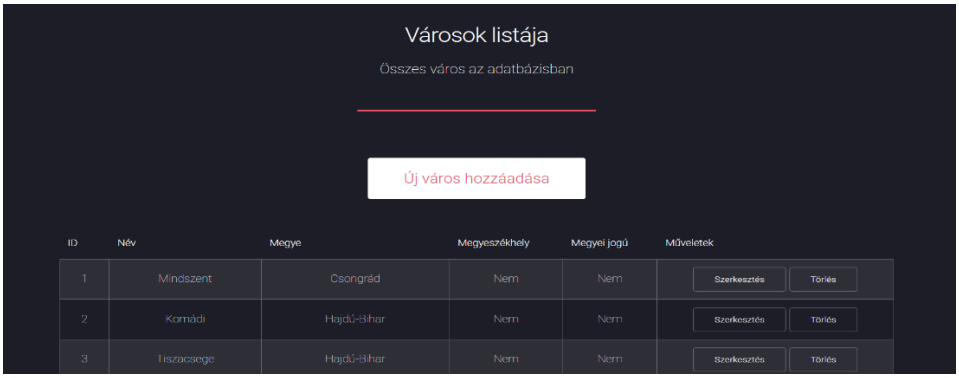
src/main/java/com/example/demo/service/CityService.java

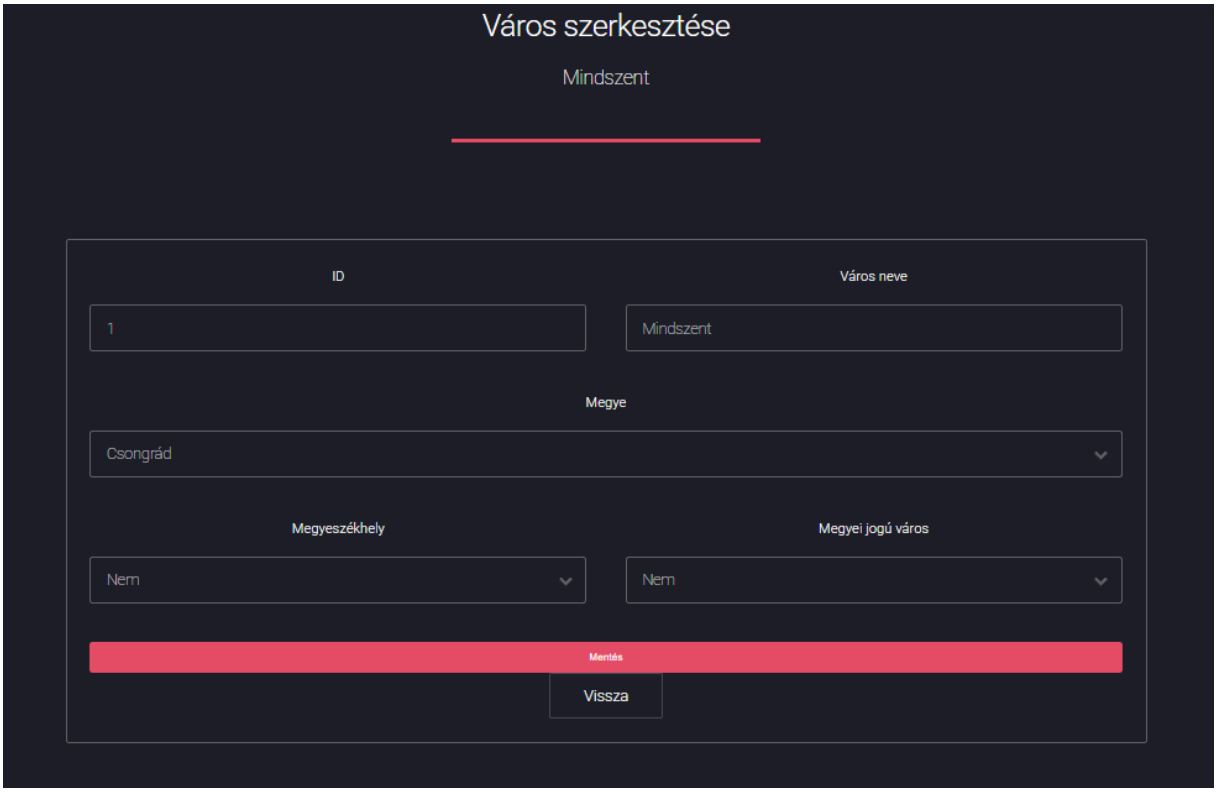
```java
public interface CityService {  4 usages  1 implementation
    List<City> findAll();  1 usage  1 implementation
    City findById(Long id);  2 usages  1 implementation
    City save(City city);  1 implementation
    void deleteById(Long id);  1 usage  1 implementation
    boolean existsById(Long id);  no usages  1 implementation
    Map<Long, Long> getCountByCounty();  no usages  1 implementation
}
```

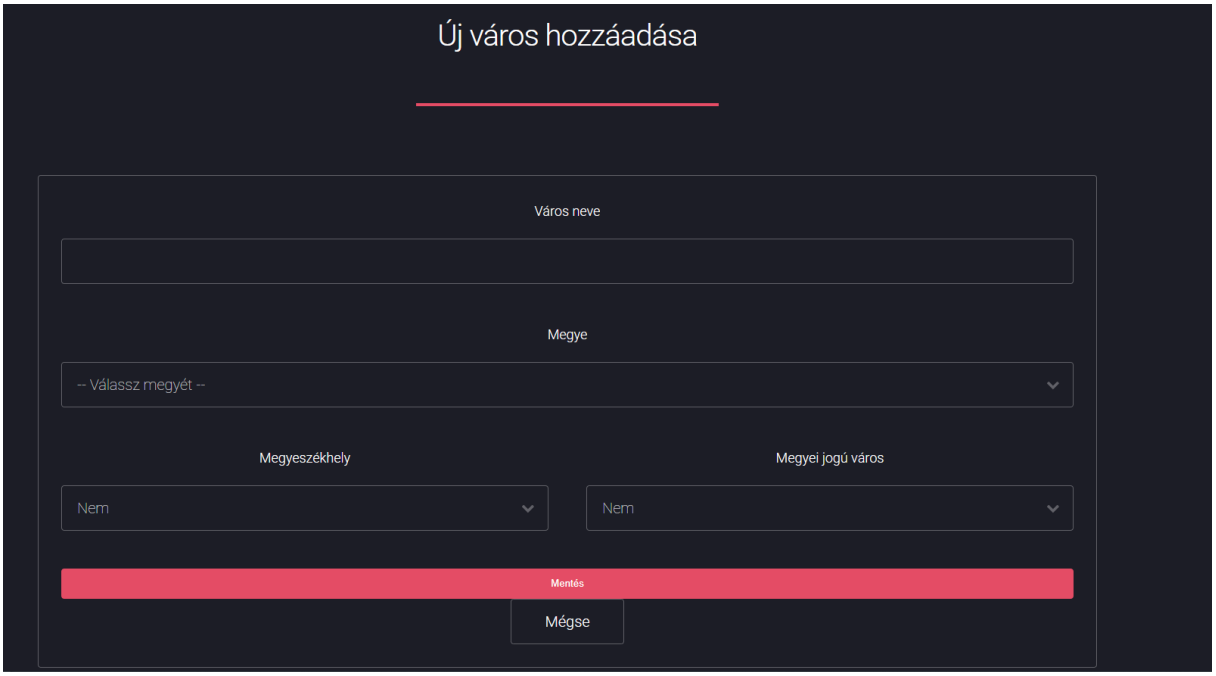
src/main/java/com/example/demo/service/CityServiceImpl.java

```java
@Service
public class CityServiceImpl implements CityService {

    private final CityRepository cityRepository;  7 usages

    public CityServiceImpl(CityRepository cityRepository) { this.cityRepository = cityRepository; }

    @Override  1 usage
    public List<City> findAll() { return cityRepository.findAll(); }

    @Override  2 usages
    public City findById(Long id) { return cityRepository.findById(id).orElse( other: null); }

    @Override
    public City save(City city) { return cityRepository.save(city); }

    @Override  1 usage
    public void deleteById(Long id) { cityRepository.deleteById(id); }

    @Override  no usages
    public boolean existsById(Long id) { return cityRepository.existsById(id); }
    @Override  no usages
    public Map<Long, Long> getCountByCounty() {
        return cityRepository.findAll().stream()
                .collect(Collectors.groupingBy(
                        City city -> city.getId(),
                        Collectors.counting()
                ));
    }
```

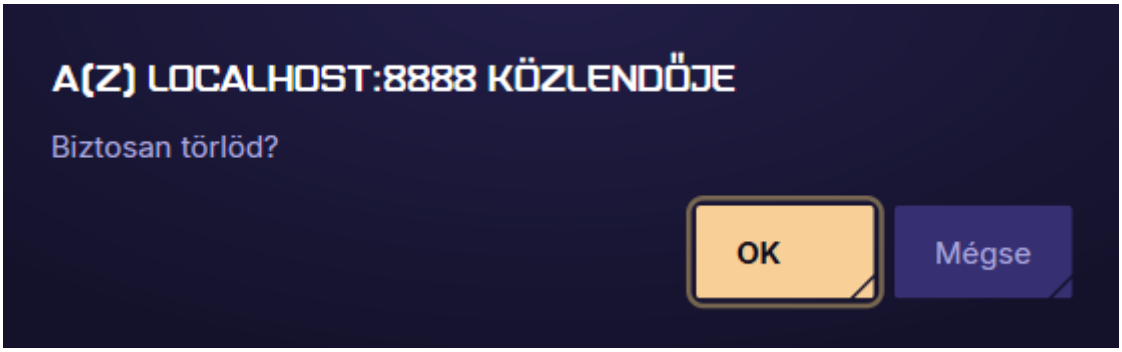
A **CityCrudController** kezeli a városokhoz tartozó összes **CRUD** műveletet, vagyis a városok listázását, létrehozását, szerkesztését és törlését. A **CityService** és annak implementációja biztosítja az adatbázissal való kommunikációt, például a lekérdezést, mentést és törlést. A vezérlő a megyeadatokat is betölti a megfelelő űrlapokhoz, így a városok megyéhez rendelhetők




Város szerkesztése
Mindszent
ID
1
Város neve
Mindszent
Megye
Csongrád
Megyeszékhely
Nem
Megyei jogú város
Nem
Mentés
Vissza


Új város hozzáadása
Város neve
Megye
-- Válassz megyét --
Megyeszékhely
Nem
Megyei jogú város
Nem
Mentés
Mégse

Város törlése:



9. RESTful menü

-

10. Admin menü:

Az Admin menü egy olyan külön felület, amely csak bejelentkezett adminisztrátor számára érhető el, így a normál felhasználók nem látják

src/main/java/com/example/demo/controller/AdminController.java

```java
import ...

@Controller
@RequestMapping("/admin")
public class AdminController {

    @Autowired
    private UserRepository userRepository;

    @GetMapping({"", "/", "/users"})
    public String showUsers(Model model) {
        List<User> users = userRepository.findAll();
        model.addAttribute( attributeName: "users", users);
        model.addAttribute( attributeName: "pageTitle", attributeValue: "Admin - Felhasználók kezelése");
        return "admin/users";
    }
}
```

Az **AdminController** egy vezérlő, amely az /admin útvonalon elérhető admin felületet kezeli.
A **showUsers** metódus lekéri az összes felhasználót a **UserRepository** segítségével.
A lekért felhasználókat és az oldal címét hozzáadja a **Model**-hez.
Visszatér az **admin/users** nézethez, ahol a felhasználók listája megjelenik az **admin** felületen.
A menü és a funkciók csak az adminok számára láthatók.

11	Registered	registered@gmail.com	registered
12	admin	Admin@gmail.com	admin