Java Gyakorlat Dokumentáció

Github link: https://github.com/FLBence/Java_Gyakorlat_Beadando

Faragó Lajos Bence github felhasználó: FLBence

Faragó Lajos Bence NEPTUN kód: UZBB3N

Varga Bence github felhasználó: V-Bence

Varga Bence NEPTUN kód: OW0PUO

Oldal linkje: http://193.224.222.53:9443/uzbb3n-gy/

Teszteléshez: admin@gmail.com jelszo1

Tartalom:

Megoldott feladatok:

Faragó Lajos Bence:1,7,8,10

Varga Bence: 2,3,4,5,6

1. Feladat: Reszponzív téma

A feladatunkhoz mi a https://html5up.net/ oldalról a Dimension nevű reszponzív témát választottuk. A választás oka az, hogy ez egy letisztult dizájnt biztosít az oldalunknak.

2. Feladat: Autentikáció

User felhasználói belépés: user@gmail.com, jelszó: user123

Az AuthController felelős a felhasználói autentikációhoz kapcsolodó nézetek és műveletek kezeléséért. A /login végpont megjeleníti a bejelentkezési oldalt, míg a /register GET kérés előkészíti és betölti a regisztrációs űrlapot. A regisztrációt feldolgozó POST metódus ellenőrzi, hogy a megadott e-mail már létezik-e, majd elmenti az új felhasználót, és sikeres regisztráció után a bejelentkezési oldalra irányítja a felhasználót.

```java
@Controller  V-Bence
public class AuthController {

    private final UserRepository repo;  3 usages
    private final PasswordEncoder encoder;  2 usages

    public AuthController(UserRepository repo, PasswordEncoder encoder) {  V-Bence
        this.repo = repo;
        this.encoder = encoder;
    }

    @GetMapping("/login")  V-Bence
    public String loginPage() { return "login"; // templates/login.html }

    @GetMapping("/register")  V-Bence
    public String registerPage(Model model) {
        model.addAttribute( attributeName: "user", new User());
        return "register"; // templates/register.html
    }

    @PostMapping("/register")  V-Bence
    public String processRegister(@ModelAttribute("user") User user,
                                  Model model) {

        if (repo.findByEmail(user.getEmail()).isPresent()) {
            model.addAttribute( attributeName: "error",  attributeValue: "Ilyen email már létezik.");
            return "register";
        }

        user.setPassword(encoder.encode(user.getPassword()));
        user.setRole("REG"); // Minden új felhasználó REG lesz

        repo.save(user);
        model.addAttribute( attributeName: "success",  attributeValue: "Sikeres regisztráció!");
        return "login";
    }
}
```

login.html:

A bejelentkezés oldal egy egyszerű űrlapot jelenít meg, amelyben a felhasználó megadhatja e-mail címét és jelszavát.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Bejelentkezés</title>
    <link rel="stylesheet" th:href="@{/css/main.css}">
    <meta charset="UTF-8">
</head>
<body class="login-oldal">
<a href="/" class="back-btn">Vissza</a>

<div class="login-kontener">
    <h2>Bejelentkezés</h2>

    <form th:action="@{/login}" method="post">
        <label>Email:</label>
        <input type="text" name="email">

        <label>Jelszó:</label>
        <input type="password" name="password">
        <p></p>
        <button type="submit">Belépés</button>
    </form>

    <div class="regisztral">
        Nincs fiókod? <a th:href="@{/register}">Regisztrálj!</a>
    </div>

</div>

</body>
</html>
```

register.html:

A regisztrációs oldal egy űrlapot jelenít meg, amely Thymeleaf adatbinding segítségével a háttérben létrehozott User objektum mezőihez kapcsolja az e-mail- és jelszóbeviteli mezőket, miközben a felhasználó a /register végpontra küldi az adatokat POST metódussal.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Regisztráció</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="/css/main.css">
</head>
<body class="login-oldal">
<a href="/" class="back-btn">Vissza</a>

<div class="login-kontener">
    <h2>Regisztráció</h2>

    <p th:if="${error}" th:text="${error}" style="..."></p>
    <p th:if="${success}" th:text="${success}" style="..."></p>

    <form th:action="@{/register}" th:object="${user}" method="post">

        <label>Email:</label>
        <input type="text" th:field="*{email}">

        <label>Jelszó:</label>
        <input type="password" th:field="*{password}">
        <p></p>
        <button type="submit">Regisztráció</button>
    </form>
    <div class="regisztral">
        Már van fiókod? <a th:href="@{/login}">Jelentkezz be!</a>
    </div>
</div>

</body>
</html>
```

3. Feladat: Főoldal menü

A főoldal egy áttekintő felület, ami bemutatja a projekt funkcióját, adatbázisát, és a felhasznált technológiákat.

```
<body class="fooldal-oldal">
<a href="/" class="back-btn">Vissza</a>

<div class="fooldal-kontener">
    <div class="container-box">

        <h1 class="text-center mb-4">Üdvözlünk a Forma-1 Adatbázis Alkalmazásban!</h1>

        <p class="mb-4">
            Ez a webalkalmazás a Forma-1 világához kapcsolódó adatok kezelését és megtekintését teszi lehetővé.
            A rendszer különböző funkciókat biztosít a felhasználók és adminisztrátorok számára.
        </p>

        <h2>✓ Funkciók</h2>
        <ul>
            <li>Pilóták adatainak megtekintése</li>
            <li>Felhasználói regisztráció és bejelentkezés</li>
            <li>Üzenetek küldése regisztrált felhasználóknak</li>
            <li>Admin funkciók (adatkezelés, törlés, módosítás)</li>
        </ul>

        <h2 class="mt-4">🗄 Adatbázis</h2>
        <p>
            A projekt egy <strong>Forma-1 tematikájú adatbázist</strong> használ, amely az alábbi adatokat tartalmazza:
        </p>
        <ul>
            <li>Pilóták adatai</li>
            <li>Felhasználók és szerepköreik</li>
            <li>Üzenetek</li>
        </ul>

        <h2 class="mt-4">🖥 Felhasznált technológiák</h2>
        <ul>
            <li>Java 17</li>
            <li>Spring Boot, Spring MVC, Spring Security</li>
            <li>Thymeleaf templaterendszer</li>
            <li>MySQL adatbázis</li>
            <li>JPA / Hibernate</li>
            <li>HTML, CSS, Bootstrap</li>
        </ul>

    </div>
</div>
```

4. Feladat: Adatbázis menü

A DbController felelős a főoldal, az adatbázis-áttekintő oldal és az adminfelület kiszolgálásáért.

```java
package com.example.java_gyak_bead.controller;
import ...

@Controller  V-Bence +1
public class DbController {

    private final PilotaRepository pilotaRepository;  2 usages
    private final GpRepository gpRepository;  2 usages
    private final EredmenyRepository eredmenyRepository;  2 usages
    private final UserRepository userRepository;  2 usages

    public DbController(PilotaRepository pilotaRepository,  V-Bence +1
                        GpRepository gpRepository,
                        EredmenyRepository eredmenyRepository, UserRepository userRepository) {
        this.pilotaRepository = pilotaRepository;
        this.gpRepository = gpRepository;
        this.eredmenyRepository = eredmenyRepository;
        this.userRepository = userRepository;
    }

    @GetMapping("/")  V-Bence
    public String index() { return "index"; }
    @GetMapping("/fooldal")  V-Bence
    public String fooldal() { return "fooldal"; }


    @GetMapping("/adatbazis")  V-Bence
    public String adatbazis(Model model) {
        model.addAttribute( attributeName: "pilotak", pilotaRepository.findAll());
        model.addAttribute( attributeName: "gp", gpRepository.findAll());
        model.addAttribute( attributeName: "eredmenyek", eredmenyRepository.findAll());
        return "adatbazis";
    }

    @GetMapping("/admin")  FLBence +1
    public String admin(Model model) {
        model.addAttribute( attributeName: "users", userRepository.findAll());
        return "admin";
    }
}
```

Pilota model:

A Pilota osztály az adatbázis pilota tábláját reprezentálja, és olyan mezőket tartalmaz, mint a pilóta neve, születési dátuma, csapata és nemzetisége.

```java
package com.example.java_gyak_bead.model;

import ...

@Entity  V-Bence +1
@Table(name = "pilota")
public class Pilota {

    @Id
    @Column(name = "az")
    private Integer id;

    private String nev;  2 usages
    private Character nem;  2 usages
    @DateTimeFormat(pattern = "yyyy-MM-dd")  2 usages
    @Temporal(TemporalType.DATE)
    private Date szuldat;
    private String nemzet;  2 usages

    public Integer getId() { return id; }

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

    public String getNev() { return nev; }

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

    public Character getNem() { return nem; }

    public void setNem(Character nem) { this.nem = nem; }

    public Date getSzuldat() { return szuldat; }

    public void setSzuldat(Date szuldat) { this.szuldat = szuldat; }

    public String getNemzet() { return nemzet; }

    public void setNemzet(String nemzet) { this.nemzet = nemzet; }
}
```

Gp model:

A Gp osztáy a gp adatbázistáblát képviseli, amely a Forma-1 versenyhelyszínek adatait tartalmazza, például a nagydíj nevét, helyszínét és dátumát.

```java
package com.example.java_gyak_bead.model;
import jakarta.persistence.*;

import java.time.LocalDate;

@Entity  V-Bence
@Table(name = "gp")
public class Gp {

    @Id
    @Column(name = "datum")
    private LocalDate datum;

    @Column(name = "nev")  2 usages
    private String nev;

    @Column(name = "helyszin")  2 usages
    private String helyszin;

    public LocalDate getDatum() {  V-Bence
        return datum;
    }

    public void setDatum(LocalDate datum) {  V-Bence
        this.datum = datum;
    }

    public String getNev() {  no usages  V-Bence
        return nev;
    }

    public void setNev(String nev) {  no usages  V-Bence
        this.nev = nev;
    }

    public String getHelyszin() {  no usages  V-Bence
        return helyszin;
    }

    public void setHelyszin(String helyszin) {  no usages  V-Bence
        this.helyszin = helyszin;
    }
}
```

Eredmeny model:

Az Eredmeny model az eredmeny táblát írja le, amely egy futam eredményeit tárolja: többek között a verseny dátumát, a pilóta azonosítóját, az elért helyezést, a hibát, ami miatt nem ért el helyezést, a csapatot, az autó típusát és a motor típusát.

```java
@Entity  V-Bence
@Table(name = "eredmeny")
public class Eredmeny {

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

    @Column(name = "datum")  2 usages
    private LocalDate datum;

    private int pilotaaz;  2 usages
    private Integer helyezes;  2 usages
    private String hiba;  2 usages
    private String csapat;  2 usages
    private String tipus;  2 usages
    private String motor;  2 usages

    public Long getId() { return id; }  V-Bence
    public void setId(Long id) { this.id = id; }  V-Bence

    public LocalDate getDatum() { return datum; }  no usages  V-Bence
    public void setDatum(LocalDate datum) { this.datum = datum; }  no usages

    public int getPilotaaz() { return pilotaaz; }  no usages  V-Bence
    public void setPilotaaz(int pilotaaz) { this.pilotaaz = pilotaaz; }

    public Integer getHelyezes() { return helyezes; }  no usages  V-Bence
    public void setHelyezes(Integer helyezes) { this.helyezes = helyezes;

    public String getHiba() { return hiba; }  no usages  V-Bence
    public void setHiba(String hiba) { this.hiba = hiba; }  no usages  V-Ben

    public String getCsapat() { return csapat; }  no usages  V-Bence
    public void setCsapat(String csapat) { this.csapat = csapat; }  no usage

    public String getTipus() { return tipus; }  no usages  V-Bence
    public void setTipus(String tipus) { this.tipus = tipus; }  no usages

    public String getMotor() { return motor; }  no usages  V-Bence
    public void setMotor(String motor) { this.motor = motor; }  no usages
}
```

PilotaRepository:

A PilotaRepository a Spring Data JPA egyik alapvető repository interfésze, amely a Pilota entitásokon végez CRUD műveleteket. A JpaRepository<Pilota, Integer> kiterjesztésével automatikusan biztosítja az összes CRUD funkciót külön kód nélkül.

GpRepository:

A GpRepository hasonló módon működik, mivel a JpaRepository<Gp, Integer> kiterjesztés lehetővé teszi a nagydíj-adatok teljes körű adatbázis-kezelését.

EredmenyRepository:

AZ EredmenyRepository a JpaRepository<Eredmeny, Integer> interfész bővíti ki, így az eredmények adatbázisban történő kezelésére szolgál.

5. Feladat: Kapcsolat menü

A KapcsolatController kezeli a kapcsolatfelvételi űrlap megjelenítését és feldolgozását: a /kapcsolat GET végpont egy új Uzenet objektumot ad a modellhez, amelyet a felhasználó kitölthet, míg a POST metódus a beküldött adatokat érvényesíti, és hiba esetén visszairányítja a felhasználót az űrlaphoz. Sikeres validáció esetén a rendszer automatikusan beállítja az üzenet elküldésének időpontját, és elmenti azt az adatbázisba.

```java
package com.example.java_gyak_bead.controller;

import com.example.java_gyak_bead.repository.UzenetRepository;
import com.example.java_gyak_bead.model.Uzenet;
import jakarta.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

import java.time.LocalDateTime;

@Controller  V-Bence
public class KapcsolatController {

    @Autowired
    private UzenetRepository uzenetRepository;

    @GetMapping("/kapcsolat")  V-Bence
    public String kapcsolatForm(Model model) {
        model.addAttribute( attributeName: "form", new Uzenet());
        return "kapcsolat";
    }

    @PostMapping("/kapcsolat")  V-Bence
    public String kuldes(
            @Valid @ModelAttribute("form") Uzenet uzenet,
            BindingResult bindingResult) {

        if (bindingResult.hasErrors()) {
            return "kapcsolat";
        }

        uzenet.setKuldve(LocalDateTime.now());
        uzenetRepository.save(uzenet);

        return "redirect:/uzenetek";
    }
}
```

kapcsolat.html:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Kapcsolat</title>
    <link rel="stylesheet" th:href="@{/css/main.css}">
</head>
<body class="kapcsolat-oldal">
<a href="/" class="back-btn">Vissza</a>

<div class="kapcsolat-kontener">

    <h1>Kapcsolat</h1>

    <form th:action="@{/kapcsolat}" th:object="${form}" method="post">

        <label>Név:</label>
        <input type="text" th:field="*{nev}">
        <p class="error" th:if="${#fields.hasErrors('nev')}" th:errors="*{nev}"></p>

        <label>Email:</label>
        <input type="email" th:field="*{email}">
        <p class="error" th:if="${#fields.hasErrors('email')}" th:errors="*{email}"></p>

        <label>Üzenet:</label>
        <textarea rows="5" th:field="*{uzenet}"></textarea>
        <p class="error" th:if="${#fields.hasErrors('uzenet')}" th:errors="*{uzenet}"></p>

        <button type="submit">Küldés</button>
    </form>
</div>

</body>
</html>
```

6.Feladat: Üzenet menü

Az UzenetekController a kapcsolatfelvételi űrlapon keresztül beküldött üzenetek megjelenítéséért felel, mivel a /uzenetek végponton keresztül lekéri az összes üzenetet az UzenetRepository segítségével, majd ezeket a modellhez adja, hogy a nézetben teljes listaként jelenhessenek meg.

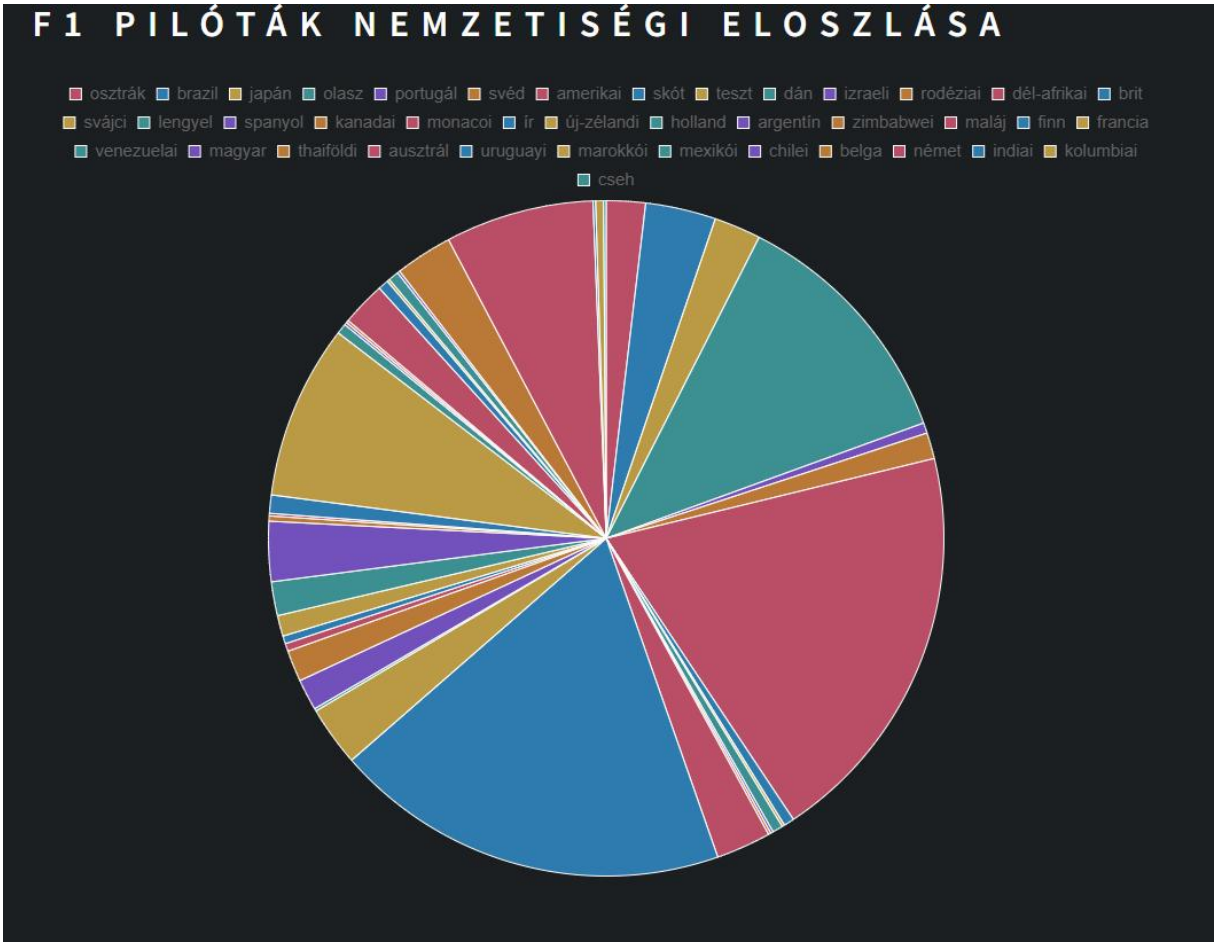
```java
package com.example.java_gyak_bead.controller;

import com.example.java_gyak_bead.repository.UzenetRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller  V-Bence
public class UzenetekController {

    @Autowired
    private UzenetRepository uzenetRepository;

    @GetMapping("/uzenetek")  V-Bence
    public String uzenetek(Model model) {
        model.addAttribute( attributeName: "uzenetek", uzenetRepository.findAll());
        return "uzenetek";
    }
}
```

uzenetek.html:

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Beérkező üzenetek</title>
    <link rel="stylesheet" href="/css/main.css">
</head>
<body class="uzenetek-oldal">
<a href="/" class="back-btn">Vissza</a>
<h1>Beérkező üzenetek</h1>

<table border="1">
    <tr>
        <th>Név</th>
        <th>Email</th>
        <th>Üzenet</th>
        <th>Küldve</th>
    </tr>

    <tr th:each="u : ${uzenetek}">
        <td th:text="${u.nev}"></td>
        <td th:text="${u.email}"></td>
        <td th:text="${u.uzenet}"></td>
        <td th:text="${u.kuldve}"></td>
    </tr>
</table>

</body>
</html>
```

7.Feladat: Diagram menü

A ChartController lekéri az összes pilóta adatát, ezeket csoportosítja nemzetiség alapján, amit aztán átad a chart.html-nek amely a kapott adatokból canvas segítségével diagrammot készít.




```java
@Controller FLBence
public class ChartController {

    @Autowired
    private PilotaRepository pilotaRepository;

    @GetMapping("/chart") FLBence
    public String chartPage(Model model) {
        List<Pilota> pilotak = pilotaRepository.findAll();

        Map<String, Long> nationalityCount = pilotak.stream()
                .collect(Collectors.groupingBy(Pilota::getNemzet, Collectors.counting()));

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

        return "chart";
    }
}
```

```
<div style="width: 1000px; height: 700px; margin: 0 auto;">
    <h2>F1 Pilóták Nemzetiségi Eloszlása</h2>
    <canvas id="f1Chart" width="1000" height="600"></canvas>

    <script th:inline="javascript">
        const labels = /*[[${labels}]]*/ [];
        const data = /*[[${data}]]*/ [];

        const ctx = document.getElementById('f1Chart').getContext('2d');
        const f1Chart = new Chart(ctx, {
            type: 'pie',
            data: {
                labels: labels,
                datasets: [{
                    label: 'Pilóták száma',
                    data: data,
                    backgroundColor: [
                        'rgba(255, 99, 132, 0.7)',
                        'rgba(54, 162, 235, 0.7)',
                        'rgba(255, 206, 86, 0.7)',
                        'rgba(75, 192, 192, 0.7)',
                        'rgba(153, 102, 255, 0.7)',
                        'rgba(255, 159, 64, 0.7)'
                    ],
                    borderColor: 'rgba(255, 255, 255, 1)',
                    borderWidth: 1
                }]
            },
            options: {
                responsive: true,
                maintainAspectRatio: false,
                plugins: {
                    legend: {
                        position: 'top',
                        labels: {
                            boxWidth: 10,
                            boxHeight: 10,
                            font: {
                                size: 15
                            }
```

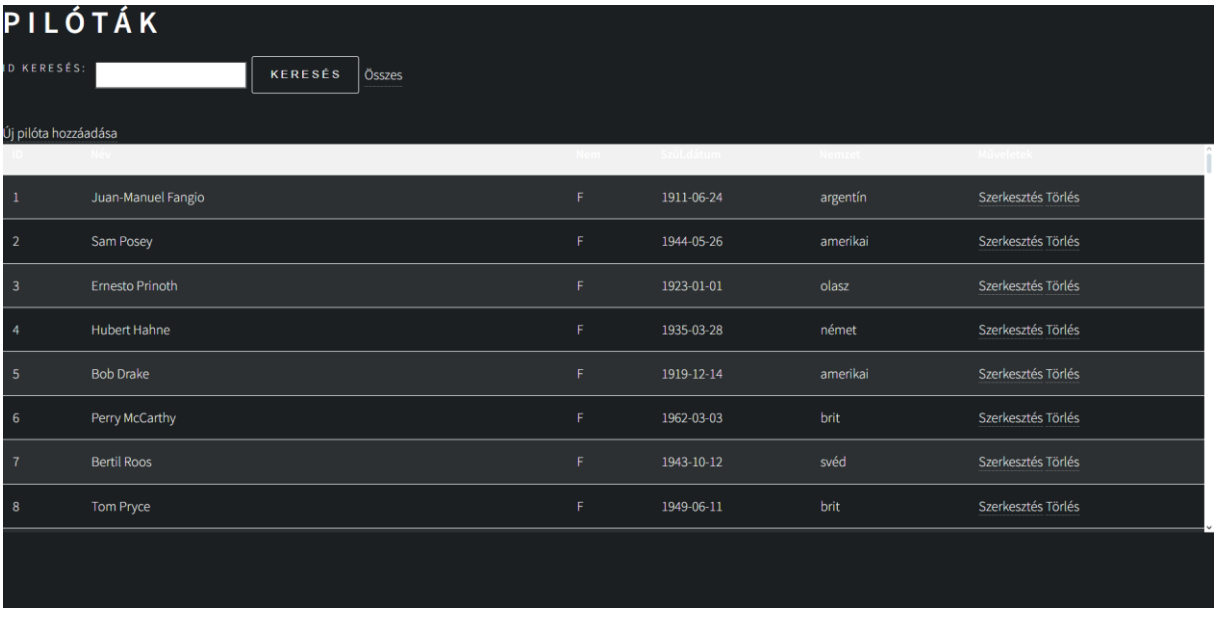
8.Feladat: CRUD menü

A CRUD menüben a CRUD funkciókat valósítjuk meg (Create, Read, Update, Delete). A CRUD kontrollert beágyaztuk a PilotaControllerbe mivel ezen a táblán hajtjuk végre a műveleteket.

```
private final CRUDService service;  7 usages

public PilotaController(CRUDService service) {  FLBence
    this.service = service;
}

@GetMapping  FLBence
public String list(Model model) {
    model.addAttribute( attributeName: "pilotak", service.getall());
    model.addAttribute( attributeName: "ujPilota", new Pilota());
    return "crud";
}

@GetMapping("/add")  FLBence
public String showAddForm(Model model) {
    model.addAttribute( attributeName: "pilota", new Pilota());
    return "crud-add";
}
@PostMapping("/add")  FLBence
public String add(@ModelAttribute("pilota") Pilota pilota, Model model) {
    try {
        service.save(pilota);
        return "redirect:/crud";
    } catch (IllegalArgumentException ex) {
        model.addAttribute( attributeName: "error", ex.getMessage());
        return "crud-add";
    }
}
@GetMapping("/search")  FLBence
public String search(@RequestParam Integer id, Model model) {
    model.addAttribute( attributeName: "pilotak", service.getone(id).map(List::of).orElse(List.
    model.addAttribute( attributeName: "ujPilota", new Pilota());
    return "crud";
}
```

A CRUDService-ben pedig a szükséges service-eket végzi el, mint például az összes pilóta lekérdezése, vagy id alapján való lekérdezés, létrehozáskor figyeli van-e azonos id, ha van akkor szól a felhasználónak arról, hogy az id nem megfelelő, frissítést pedig csak akkor végez el, ha megtalálja az általunk keresett id-t.

```java
@Service  3 usages  FLBence
public class CRUDService {

    private final PilotaRepository repo;  8 usages

    public CRUDService(PilotaRepository repo) { this.repo = repo; }

    public List<Pilota> getall() {  1 usage  FLBence
        return repo.findAll();
    }

    public Optional<Pilota> getone(Integer id) {  2 usages  FLBence
        return repo.findById(id);
    }

    public Pilota save(Pilota pilota) {  FLBence
        if (repo.existsById(pilota.getId())) {
            throw new IllegalArgumentException("Már létezik ilyen ID: " + pilota.getId());
        }
        return repo.save(pilota);
    }

    public void delete(Integer id) { repo.deleteById(id); }

    public void update(Integer id, Pilota mod) {  1 usage  FLBence
        repo.findById(id).ifPresent( Pilota p -> {
            p.setNev(mod.getNev());
            p.setNem(mod.getNem());
            p.setSzuldat(mod.getSzuldat());
            p.setNemzet(mod.getNemzet());
            repo.save(p);
        });
    }
```



10.Feladat: Admin menü

Az admin menü nevéből adódóan is csak az admin jogosultságú felhasználók számára érhető el. Az admin menün az admin felhasználó láthatja az összes regisztrált felhasználót és azok adatait.