

Java alkalmazások gyakorlat beadandó feladat

Készítő: Szebeni Szabolcs (LNGZRW), Tóth Szabolcs Botond (TB8BGC)

Github projekt címe: <https://github.com/szabi9250/JavaGyakBead>

Választott téma: HTML5 UP Arcana
(<https://html5up.net/uploads/demos/arcana/>)

Választott adatbázis: Cukrászda

Linux tárhely weboldal elérhetősége: <http://rivendell.nje.hu:9443/LNGZRW-gy/>

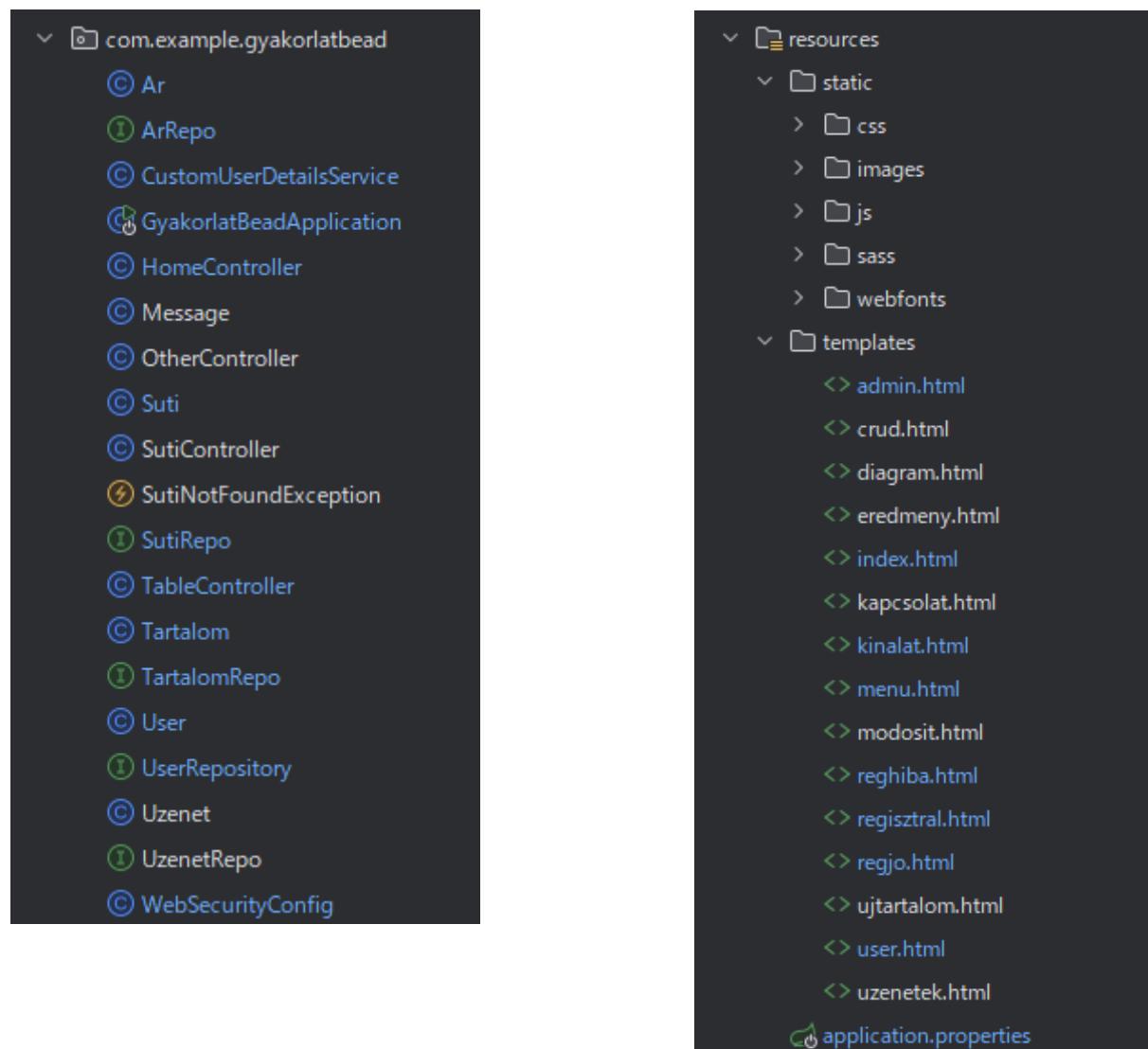
Alapértelmezett bejelentkezések:

- **admin felhasználó:** admin@gmail.com, jelszó: jelszo1 (admin menüpontot csak admin láthatja)
- **felhasználó:** user@gmail.com, jelszó: jelszo2 (üzenet menüpontot csak regisztrált felhasználó láthatja)

Elkészített funkciók:

- Autentikáció látogató, regisztrált látogató, és admin szerepkörrel elkülönítve
- Főoldal menü amely leírja az elkészített funkciókat
- Kínálat menüpont, amely az adatbázis 3 tábláját írja ki
- Kapcsolat menü, ahol üzenetet lehet küldeni az adatbázisba
- Üzenet menü, ahol meglehet nézni a beküldött üzeneteket (csak regisztrált felhasználó láthatja)
- Diagram, amely kiírja a süteményeket típusok alapján ChartJS segítségével
- CRUD menü, amelyben lehetségesek a CRUD funkciók az adatbázis egyik táblájában
- RESTful
- Admin menü, amelyet csak admin láthat

A projekt könyvtára:



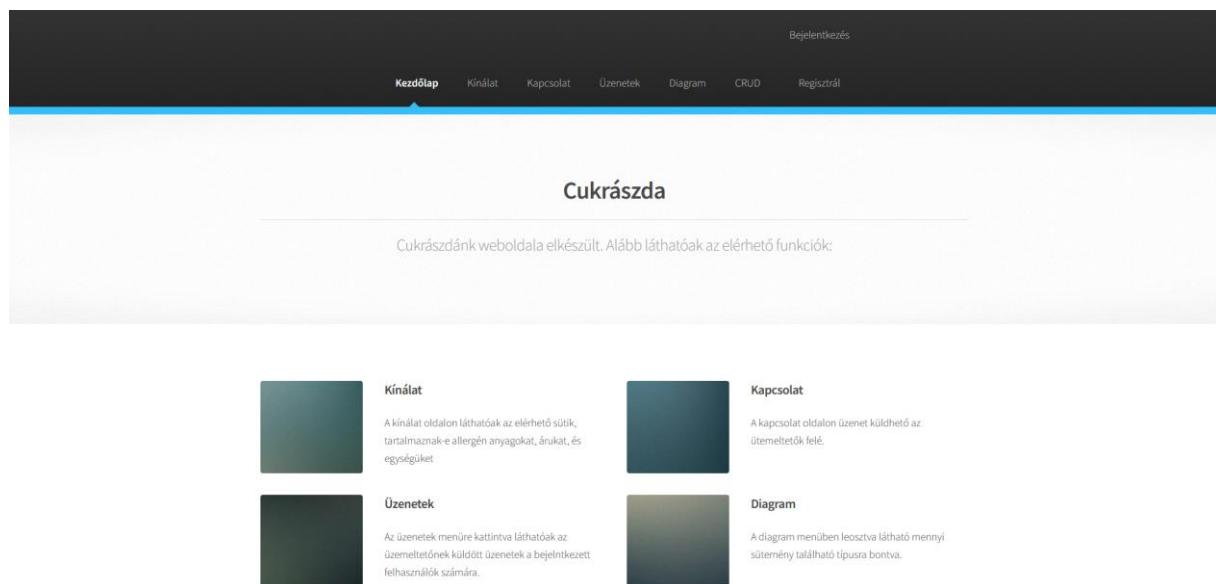
A com.example.gyakorlatbead-ban találhatóak az elkészített classok és interface-k, még a resources mappában található a static-on belül az összes formázással kapcsolatos fájl, még a Thymeleaf-et használó oldalak a templates mappában találhatóak.

A bejelentkezési rendszerrel kapcsolatos controller-ek a HomeController-ben, egyéb bejelentkezássel kapcsolatos kód a WebSecurityConfig és CustomUserDetailsService-ben van megvalósítva, míg az OtherController-ben található a legtöbb egyéb GetMapping és PostMapping függvény, a SutiController-ben pedig a Restful API-ért felelősek. A TableController-ben található a táblák kiíratásáért felelős controller függvény.

Táblák és adatbázis kapcsolata leírva a Suti, Tartalom, Ar, Uzenet, Users classokban.

A githubon feltöltött beadando.sql fájl tartalmazza a beadando nevezetű adatbázisba importálandó táblákat, és adatokat amelyek szükségesek az alkalmazás futtatásához.

Az alkalmazás elindítása után a localhost:8080-on elérhetővé válik, és a következő kezdőlap jön be, amely leírja mi van megvalósítva az oldalon:



Elérhető a főoldalról a többi elkészített funkciót tartalmazott menüpont, és a regisztrálás, bejelentkezés gomb (megvalósítva az index.html-ben).

A regisztrálás gombra kattintva a felhasználó regisztrálhat egy név, jelszó, és nem foglalt email cím felhasználásával (regisztral.html).

Regisztrálás még nem felhasználók számára

Regisztráció

Új Név

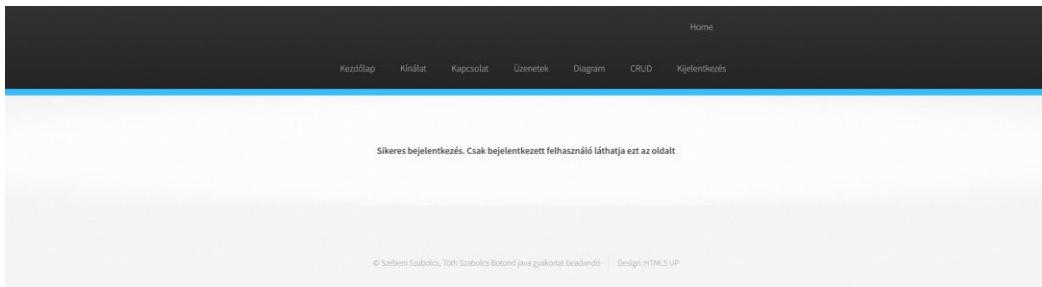
emailcim@email.hu

Regisztráció

Regisztrálás után megjelenik a sikeres regisztráció, más esetben ha foglalt az email cím (regjo.html, ellenkezője reghiba.html).



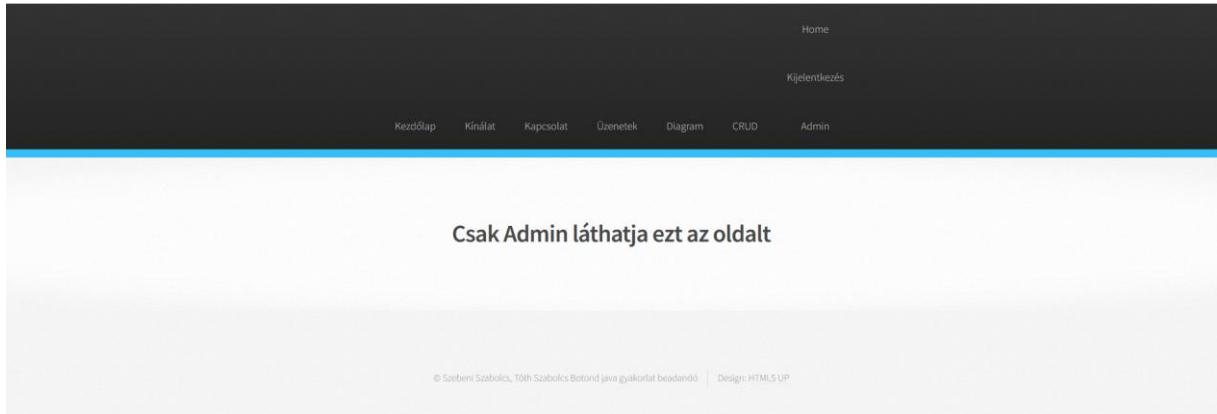
Ezután a felhasználó bejelentkezhet emaillel és jelszóval. Bejelentkezés után egy olyan oldalt lát, amelyet csak bejelentkezett felhasználó láthat (user.html).



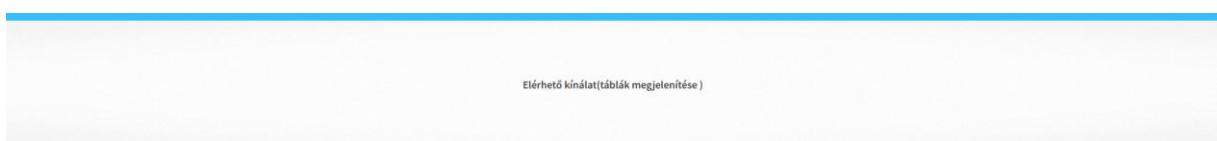
Elérhető oldalakat kezelő kód (WebSecurityConfig class):

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http
        .authorizeHttpRequests(AuthorizationManagerRequestMatcher::auth)
        .requestMatchers("*.css/*", "*/js/*", "*/images/**", "*/sass/**", "*/webfonts/**").permitAll()
        .requestMatchers("*/javascrip/*", "*/css/*").hasRole("FELHASZNALO")
        .requestMatchers("*/resources/*", "*/'", "*/registrat*", "*/regisztral_feldolgoz*", "*/kinalat*", "*/an", "*/kapcsolat", "*/loginh*", "*/uzenetek", "*/user*").authenticated()
        .requestMatchers("*/uzenetek", "*/user*").authenticated()
        .formLogin().loginPage("*/login*").username("admin@gmail.com").password("jelszo1")
        .requestMatchers("*/admin*").hasRole("ADMIN")
        .logout().logoutPage("*/logout*").logoutSuccessUrl("*/").permitAll()
        .requestMatchers("*/login*").anonymous()
        .requestMatchers("*/uzenetek").anonymous()
        .requestMatchers("*/regisztral*").anonymous()
        .requestMatchers("*/regisztral/*").permitAll()
    )
    .formLogin().formLoginConfigurer((mpSecurity) -> mpSecurity.form())
    .defaultSuccessUrl("*/home")
    .permitAll()
);
    return http.build();
}
```

Ha adminisztrátor lép be elérheti az admin menüpontot is, amely csak számára érhető el (admin.html):



A kínálat gombra nyomva elérhető a táblákat egybe kiíró menü, amely kiírja a sütemények nevét, allergén tartalmát, ha van, árát, és milyen egységen vannak mérve. Az oldal megvalósítása a kinalat.html-ben.



Suti	Tartalom	Ar	Egység
Süni		300	db
Gesztenyealagút		500	db
Sajtos pogácsa	sd	3300	kg
Diós-mákos		3500	koszonú
Módosított süti		9000	12 szeletes
Citrom	L	3900	8 szeletes
Citrom	L	7400	16 szeletes
Citrom	L	11400	24 szeletes
Ezterházy		490	db
Rákóczi-túró		400	db
Meggyes kocka		450	db
Legényfogó		12100	24 szeletes
Legényfogó		4300	8 szeletes
Legényfogó		8200	16 szeletes

A táblák class-ba vannak felvéve, és emellett Repository-kkal rendelkeznek.

SutiRepository:

```
public interface SutiRepo extends CrudRepository<Suti, Integer>{
    @Query("SELECT i.tipus, COUNT(i) FROM Suti i GROUP BY i.tipus")
    List<Object[]> countByTipus();
    @Query("SELECT s, t, a.ertek, a.egyseg FROM Suti s " +
           "LEFT JOIN Tartalom t ON t.sutiid = s.id " +
           "LEFT JOIN Ar a ON a.sutiid = s.id")
    List<Object[]> fullData();
}
```

Suti class:

```
@Entity
@Table(name = "suti")
public class Suti {
    @Id
    private int id;
    @Column(name = "nev")
    private String nev;
    @Column(name = "tipus")
    private String tipus;
    @Column(name = "dijazott")
    private boolean dijazott;

    public int getId() { return id; }

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

    public String getNev() { return nev; }

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

    public String getTipus() { return tipus; }

    public void setTipus(String tipus) { this.tipus = tipus; }

    public boolean isDijazott() {
        return dijazott;
    }

    public void setDijazott(boolean dijazott) {
        this.dijazott = dijazott;
    }

    //Restful
    public Suti() {
    }

    public Suti(String nev, String tipus, boolean dijazott) {
        this.nev = nev;
        this.tipus = tipus;
        this.dijazott = dijazott;
    }
}
```

Megvalósítják a MySQL-be felvett táblával lévő kapcsolatot. Többi tábla megvalósítása az Ar, ArRepo-ban, a Tartalom, TartalomRepo-ban, User, és UserRepository-ba, és az Uzenet és UzenetRepo-ban.

Kapcsolat menüpont (megvalósítva a kapcsolat.html-ben):

The screenshot shows a dark-themed navigation bar with several menu items: 'Kezdőlap', 'Rendel', 'Kapcsolat' (which is highlighted in blue), 'Üzenek', 'Diagram', 'CRUD', and 'Regisztráció'. Below the bar, the word 'Ürlap' is centered on the page.

Név: (Min 2, Max 30 karakter)

Tárgy: (Min 2, Max 30 karakter)

Üzenet: (Min 2, Max 250 karakter)

Elküld **Helyreállít**

A kapcsolat menüpont segítségével lehetséges üzenet küldése az adatbázisba. Ellenőrzött bevitel történik minden mezőnél. Az elküld gomb segítségével elküldhető, a helyreállítás segítségével kitörölhetőek az eddig beírtak. Rossz bevitel esetén kiírja a hibát.

Név: (Min 2, Max 30 karakter)
N

az méretnek a(z) 2 és 30 értékek között kell lennie

Fárgy: (Min 2, Max 30 karakter)
S

az méretnek a(z) 2 és 20 értékek között kell lennie

Üzenet: (Min 2, Max 250 karakter)
ds

Elküld **Helyreállít**

Elküldés után kiírja az elküldött üzenetet egy új megjelenített oldalon (kapcsolatk.html)



Iév: Új név
Árgy: Üzenet
Üzenet: Üzenet
[Másik üzenet küldése](#)

Megvalósítva az Uzenet class-ban, UzenetRepo-ban, és az controller az OtherController-ben.

Uzenet class:

```
@Entity
@Table(name = "uzenetek")
public class Uzenet {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nev;
    private String targy;
    private String uzenet;

    @CreationTimestamp
    @Column(name = "ido", nullable = false, updatable = false)
    private LocalDateTime ido;

    public Uzenet() {}

    public Uzenet(String nev, String targy, String uzenet) {}

    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 getTargy() { return targy; }

    public void setTargy(String targy) { this.targy = targy; }

    public String getUzenet() { return uzenet; }

    public void setUzenet(String uzenet) { this.uzenet = uzenet; }

    public LocalDateTime getIdo() { return ido; }

    public void setIdo(LocalDateTime ido) { this.ido = ido; }
}
```

UzenetRepository:

```
public interface UserRepository extends CrudRepository<User, Integer>{  
    Optional<User> findByEmail(String email);  
}
```

Kapcsolat.html:

```
<section class="wrapper style1">  
    <div>  
        <form action="#" th:action="@{/kapcsolatk}" th:object="${message}" method="post">  
            <p>Név: (Min 2, Max 30 karakter) <input type="text" th:field="*{name}" /></p>  
            <p th:if="#{fields.hasErrors('name')}" th:errors="*{name}"></p>  
            <p>Tárgy: (Min 2, Max 30 karakter) <input type="text" th:field="*{topic}" /></p>  
            <p th:if="#{fields.hasErrors('topic')}" th:errors="*{topic}"></p>  
            <p>Üzenet: (Min 2, Max 250 karakter) <input type="text" th:field="*{content}" /></p>  
            <p th:if="#{fields.hasErrors('content')}" th:errors="*{content}"></p>  
  
            <p><input type="submit" value="Elküld" /> <input type="reset" value="Helyreállít" /></p>  
        </form>  
    </div>  
</section>
```

Az üzenetek gombra rányomva, ha a felhasználó nincs bejelentkezve, akkor a bejelentkezési képernyő nyílik meg:



Bejelentkezés után a felhasználó láthatja a beküldött üzeneteket időben a legújabbat felülről látva (megvalósítva az uzenetek.html-ben):

Id	Név	Tárgy	Üzenet	Üzenet küldés ideje
5	Másik üzenet	Üzenettárgy	Üz	2025-12-06T22:01:05
4	Új név	Üzenet	Üzenet	2025-12-06T21:51:52

uzenetek.html:

```
<section class="wrapper style1">
    <div>
        <table>
            <thead>
                <tr>
                    <th>Id</th>
                    <th>Név</th>
                    <th>Tárgy</th>
                    <th>Üzenet</th>
                    <th>Üzenet küldés ideje</th>
                </tr>
            </thead>
            <tbody>
                <tr th:each="u : ${uzenetek}">
                    <td th:text="${u.id}"></td>
                    <td th:text="${u.nev}"></td>
                    <td th:text="${u.targy}"></td>
                    <td th:text="${u.uzenet}"></td>
                    <td th:text="${u.ido}"></td>
                </tr>
            </tbody>
        </table>
    </div>>
</section>
```

Restful menü: A Postman segítségével elvégzett kérések controllere a SutiController-ben vannak elkészítve:

```
@RestController
public class SutiController {
    private final SutiRepo sutirepo1;

    SutiController(SutiRepo sutirepo1) {
        this.sutirepo1 = sutirepo1;
    }

    //Restful oldala
    @GetMapping("/{restful}")
    Iterable<Suti> readAll() { return sutirepo1.findAll(); }

    @GetMapping("/{restful}/{id}")
    Suti readOne(@PathVariable int id) {
        return sutirepo1.findById(id)
            .orElseThrow(() -> new SutiNotFoundException(id));
    }

    @PostMapping("/{restful}")
    Suti sutiload(@RequestBody Suti ujSuti) { return sutirepo1.save(ujSuti); }

    @PutMapping("/{restful}/{id}")
    Suti sutiEdit(@RequestBody Suti adatSuti, @PathVariable int id) {
        return sutirepo1.findById(id)
            .map(suti -> {
                adatSuti.getNeve();
                adatSuti.getTipus();
                adatSuti.isDijazott();
                sutirepo1.save(a);
            })
            .orElseGet(() -> {
                adatSuti.setId(id);
                return sutirepo1.save(adatSuti);
            });
    }

    @DeleteMapping("/{restful}/{id}")
    void deletesuti (@PathVariable int id){
        sutirepo1.deleteById(id);
    }
}
```

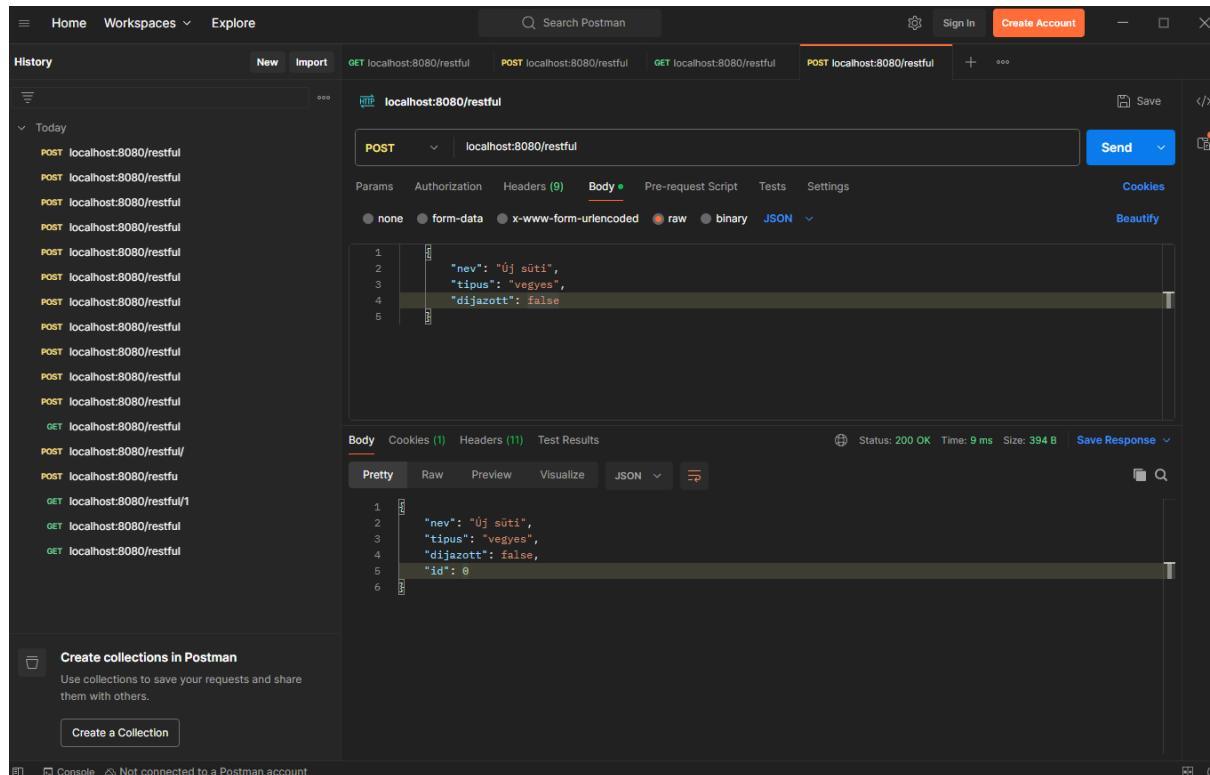
Postman segítségével get:

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Home', 'Workspaces', 'Explore', and a search bar. Below the navigation is a 'History' section showing two previous requests to 'localhost:8080/restful'. The main area displays a request configuration for a 'GET' method to 'localhost:8080/restful'. The 'Params' tab is selected, showing a single query parameter 'Key' with an empty value. On the right, the 'Body' tab shows a JSON response with two objects. The first object has 'nev': 'Süni', 'tipus': 'vegyes', 'dijazott': false, and 'id': 1. The second object has 'nev': 'Gesztenyealagút', 'tipus': 'vegyes', 'dijazott': false, and 'id': 2. The status bar at the bottom indicates a 200 OK response with a time of 15 ms and a size of 10.16 KB.

Postman segítségével 1-es id-jú süti get:

This screenshot is similar to the previous one but shows a specific request for 'localhost:8080/restful/1'. The 'Params' tab in the request configuration shows a 'Key' parameter with the value '1'. The response body shows a single object with 'nev': 'Süni', 'tipus': 'vegyes', 'dijazott': false, and 'id': 1. The status bar at the bottom indicates a 200 OK response with a time of 19 ms and a size of 390 B.

Postman segítségével új süti hozzáadása:



The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for Home, Workspaces, Explore, New, Import, and Create Account. Below the navigation, the History section shows several previous requests. A new POST request is being prepared for the URL `localhost:8080/restful`. The Body tab is selected, and the JSON content is:

```
1 "nev": "Új süti",
2 "tipus": "vegyes",
3 "dijazott": false
```

Below the request, the response pane shows the result of the POST request:

```
1 "nev": "Új süti",
2 "tipus": "vegyes",
3 "dijazott": false,
4 "id": 0
```

The status of the response is 200 OK, time is 9 ms, and size is 394 B.

A MySQL adatbázis táblájában megjelenik az új süti:

	Módosítás	Másolás	Törlés	id	nev	tipus	dijazott
<input type="checkbox"/>				126	Gesztenyés-karamellás	bejgli	0
<input type="checkbox"/>				127	Gesztenyés szív	édes teasütemény	0
<input type="checkbox"/>				128	Ropi	sós teasütemény	0
<input type="checkbox"/>				129	Paleolit étcokoládé	különleges torta	0
<input type="checkbox"/>				130	Túró	pite	0
<input type="checkbox"/>				131	Ishler	vegyes	0
<input type="checkbox"/>				132	Lúdláb	tortaszelet	0
<input type="checkbox"/>				133	Csokoládémousse	tortaszelet	0
<input type="checkbox"/>				134	Dió	torta	0
<input type="checkbox"/>				135	Krémes	krémes	0
<input type="checkbox"/>				136	Mini ischler	édes teasütemény	0
<input type="checkbox"/>				137	Paleolit étcokoládé	tortaszelet	0
<input type="checkbox"/>				138	Tejfölös túró hajtогatott	sós teasütemény	0
<input type="checkbox"/>				139	Mákos guba	torta	0
<input type="checkbox"/>				302	Új süti	vegyes	0

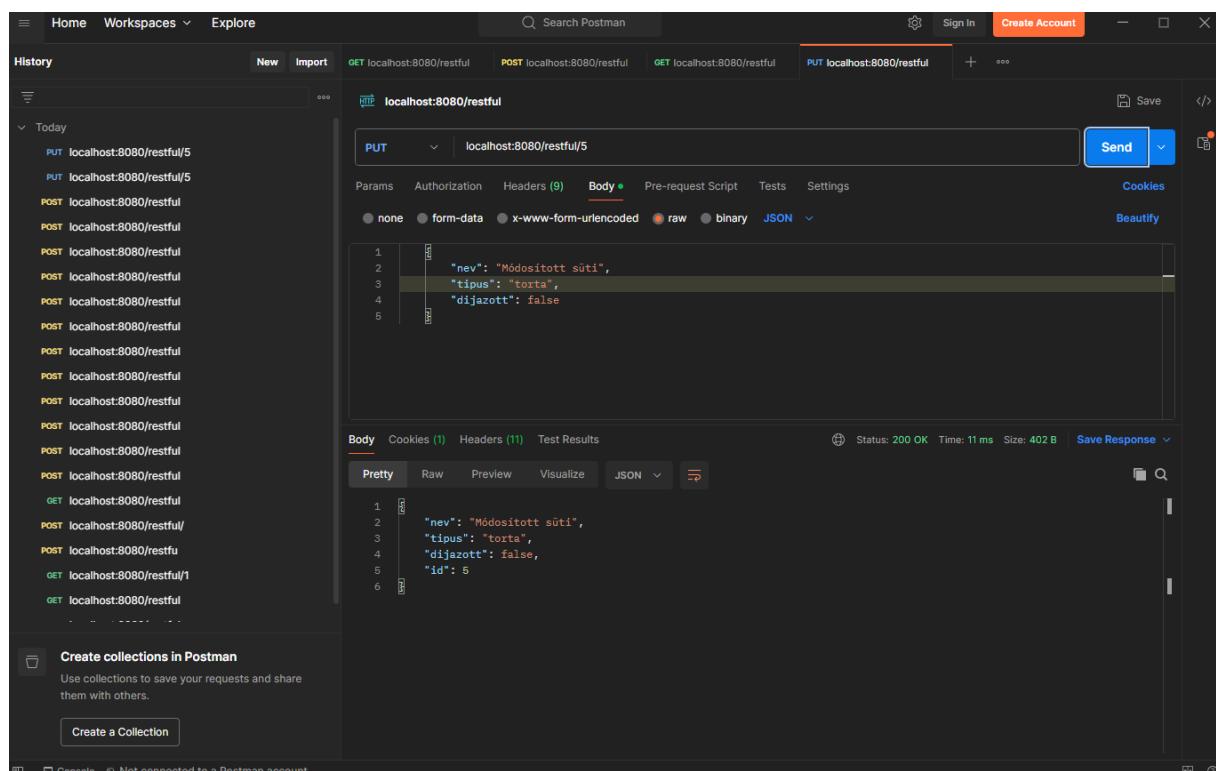
Postman segítségével süti törlése:

The screenshot shows the Postman application interface. On the left, the 'History' panel lists several requests, including a recent DELETE request to 'localhost:8080/restful/302'. The main workspace displays a DELETE request to 'localhost:8080/restful/302' with the body set to '1'. The status bar at the bottom indicates a successful response: 'Status: 200 OK Time: 17 ms Size: 293 B'.

Törlés után a megadott id-jú süti törlődik a táblából

	Módosítás	Másolás	Törlés	id	nev	típus	díjjazott
<input type="checkbox"/>				126	Gesztenyés-karamellás	bejgli	0
<input type="checkbox"/>				127	Gesztenyés szív	édes teasütemény	0
<input type="checkbox"/>				128	Ropi	sós teasütemény	0
<input type="checkbox"/>				129	Paleolit étcsonkotád	különleges torta	0
<input type="checkbox"/>				130	Túró	pite	0
<input type="checkbox"/>				131	Ishler	vegyes	0
<input type="checkbox"/>				132	Lúdláb	tortaszelet	0
<input type="checkbox"/>				133	Csokoládémoszsa	tortaszelet	0
<input type="checkbox"/>				134	Dió	torta	0
<input type="checkbox"/>				135	Krémes	krémes	0
<input type="checkbox"/>				136	Mini ischler	édes teasütemény	0
<input type="checkbox"/>				137	Paleolit étcsonkotád	tortaszelet	0
<input type="checkbox"/>				138	Tejfölös túró hajtogatott	sós teasütemény	0
<input type="checkbox"/>				139	Mákost guba	torta	0

Postman segítségével süti módosítása:



The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for Home, Workspaces, and Explore. The main area displays a history of requests made today, including various GET and POST requests to the 'restful' endpoint on localhost:8080. A specific PUT request to 'localhost:8080/restful/5' is selected. The 'Body' tab is active, showing a JSON payload:

```
1  "nev": "Módosított süti",
2  "tipus": "torta",
3  "dijazott": false
```

Below the body, the response is shown in 'Pretty' format:

```
1  "nev": "Módosított süti",
2  "tipus": "torta",
3  "dijazott": false,
4  "id": 5
```

The status bar at the bottom indicates a successful 200 OK response with a time of 11 ms and a size of 402 B.

Ezután módosítódik a MySQL adatbázis táblájában az adott süti:

	Módosítás	Másolás	Törlés	id	nev	tipus	dijazott
<input type="checkbox"/>				1	Süni	vegyes	0
<input type="checkbox"/>				2	Gesztenyealagút	vegyes	0
<input type="checkbox"/>				3	Sajtos pogácsa	sós teasütemény	0
<input type="checkbox"/>				4	Diós-mákos	bejgli	0
<input type="checkbox"/>				5	Módosított süti	torta	0
<input type="checkbox"/>				6	Citrom	torta	0

A diagram menüpont segítségével elérhető a ChartJS diagram, amely a suti táblából jeleníti meg lebontva milyen süti típusból mennyi van (diagram.html-ben megvalósítva):



Az OtherControllerben lévő kód:

```
//Diagram oldala
@GetMapping("/diagram")
public String showChart(Model model) {
    List<Object[]> results = sutiRepo.countByTipus();
    List<String> labels = new ArrayList<>();
    List<Long> data = new ArrayList<>();

    for(Object[] row : results) {
        labels.add((String) row[0]);
        data.add((Long) row[1]);
    }

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

    return "diagram";
}
```

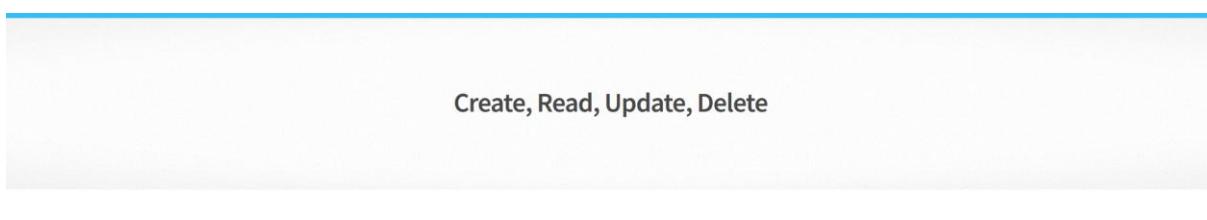
Diagram.html-ben ezért felelős kód:

```
<section class="wrapper style1">
    <div style="width: 900px; height: 500px; margin: 0 auto;">
        <canvas id="tipusChart" width="600" height="400"></canvas>
        <script th:inline="javascript">
            var labels = [[${labels}]];
            var data = [[${data}]];
```



```
            var ctx = document.getElementById('tipusChart').getContext('2d');
            var myChart = new Chart(ctx, {
                type: 'bar',
                data: {
                    labels: labels,
                    datasets: [{
                        label: 'Tipus',
                        data: data,
                        backgroundColor: 'magenta',
                        borderColor: 'blue',
                        borderWidth: 1
                    }]
                },
                options: {
                    responsive: true,
                    scales: {
                        y: { beginAtZero: true }
                    }
                }
            });
        </script>
    </div>>
</section>
```

CRUD funkciók a CRUD menüpontban kerültek megvalósításra, amely a tartalom táblát írja ki, amely tartalmazza milyen süti id-val milyen mentes az adott süti (crud.html, mentéshez modosit.html, új elemhez ujtartalom.html).



The screenshot shows a web page with a light gray header containing the text "Create, Read, Update, Delete". Below the header is a table with 12 rows of data. The table has four columns: "ID", "sutiid", "Mentes", and a column containing two buttons, "Módosítás" and "Törölés".

Íj tartalmat hozzáad			
ID	sutiid	Mentes	
4	91	G	Módosítás Törölés
5	137	G	Módosítás Törölés
6	60	Te	Módosítás Törölés
7	129	HC	Módosítás Törölés
8	122	To	Módosítás Törölés
9	90	G	Módosítás Törölés
10	26	To	Módosítás Törölés
11	94	L	Módosítás Törölés
12	46	É	Módosítás Törölés

Íj tartalmat hozzáad			
ID	sutiid	Mentes	
4	91	G	Módosítás Törölés
5	137	G	Módosítás Törölés
6	60	Te	Módosítás Törölés
7	129	HC	Módosítás Törölés
8	122	To	Módosítás Törölés
9	90	G	Módosítás Törölés
10	26	To	Módosítás Törölés
11	94	L	Módosítás Törölés
12	46	É	Módosítás Törölés

Új hozzáadásnál meg kell adni egy létező süti id-ját, és milyen legyen a mentes menü:

Módosítás

Sütid: 20

Mentes: VN

Hozzáad

Ezután hozzáadásra kerül az adott új elem:

42	63	G	Módosítás Törölés
44	129	L	Módosítás Törölés
45	15	É	Módosítás Törölés
46	21	JR	Módosítás Törölés
50	20	VN	Módosítás Törölés

>

A módosításra nyomva módosítható:

Módosít

Sütid: 20

Mentes: VNC

Módosít

Ezután megjelenik a módosítás a táblázatban:

41	20	HC	Módosítás Törölés
42	63	G	Módosítás Törölés
44	129	L	Módosítás Törölés
45	15	É	Módosítás Törölés
46	21	JR	Módosítás Törölés
50	20	VNG	Módosítás Törölés

>

A törlés gombra kattintva törölhető az elem, ezután eltűnik a táblából:

40	90	To	Módosítás Törlés
41	20	HC	Módosítás Törlés
42	63	G	Módosítás Törlés
44	129	L	Módosítás Törlés
45	15	É	Módosítás Törlés
46	21	JR	Módosítás Törlés

Crud.html tartalma:

```
<section class="wrapper style1">
  <div>
    <a href="/uj">Új tartalmat hozzáad</a>
    <table>
      <thead>
        <tr>
          <th>ID</th>
          <th>sutiid</th>
          <th>Mentes</th>
        </tr>
      </thead>
      <tbody>
        <tr th:each="t : ${tartalmak}">
          <td th:text="${t.id}"></td>
          <td th:text="${t.sutiid}"></td>
          <td th:text="${t.mentes}"></td>
          <td>
            <a th:href="@{/modosit/' + ${t.id} }">Módosítás</a>
            <a th:href="@{/torles/' + ${t.id} }">Törlés</a>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</section>
```

Ujtartalom.html:

```
<section class="wrapper style1">
  <div>
    <form action="#" th:action="@{ment}" th:object="${tartalom}" method="post">
      <p>Sutiid:<input type="text" th:field="*{sutiid}" placeholder="Süti id" required/></p>
      <p>Mentes:<input type="text" th:field="*{mentes}" placeholder="Mentes" required/></p>
      <p><input type="submit" value="Hozzáad" /></p>
    </div>
  </section>
```

Modosit.html:

```
<section class="wrapper style1">
  <div>
    <form action="#" th:action="@{/modosit}" th:object="${tartalom}" method="post">
      <input type="hidden" th:field="*{id}" />
      <p>Sutiid:<input type="text" th:field="*{sutiid}" value= "${tartalom.sutiid}" required/></p>
      <p>Mentes:<input type="text" th:field="*{mentes}" value= "${tartalom.mentes}" required/></p>
      <p><input type="submit" value="Módosít" /></p>
    </form>
  </div>
</section>
```

OtherController-ben lévő CRUD-ért felelős controller tartalmi része:

```
//CRUD oldala
@GetMapping(@RequestMapping("/crud"))
public String crud(Model model) {
    model.addAttribute("tartalmak", tartalomRepo.findAll());
    return "crud";
}

@GetMapping(@RequestMapping("/ui"))
public String newTartalom(Model model) {
    model.addAttribute("tartalom", new Tartalom());
    return "ujtartalom";
}

@PostMapping(value = "/ment")
public String saveTartalom(@ModelAttribute Tartalom tartalom) {
    tartalomRepo.save(tartalom);
    return "redirect:/";
}

@GetMapping(@RequestMapping("/modosit/{id}"))
public String editTartalom(@PathVariable(name = "id") int id, Model model) {
    model.addAttribute("tartalom", tartalomRepo.findById(id));
    return "modosit";
}

@PostMapping(@RequestMapping("/modosit"))
public String updateTartalom(@ModelAttribute Tartalom tartalom) {
    tartalomRepo.save(tartalom);
    return "redirect:/crud";
}

@GetMapping(@RequestMapping("/torles/{id}"))
public String deleteTartalom(@PathVariable(name = "id") int id) {
    tartalomRepo.delete(tartalomRepo.findById(id).get());
    return "redirect:/";
}
```