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/

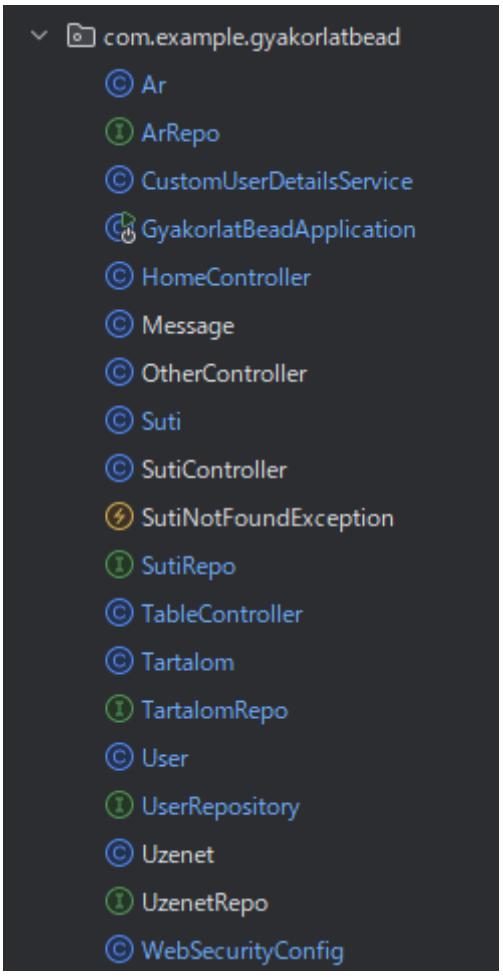
Alapértelmezett bejelentkezések:

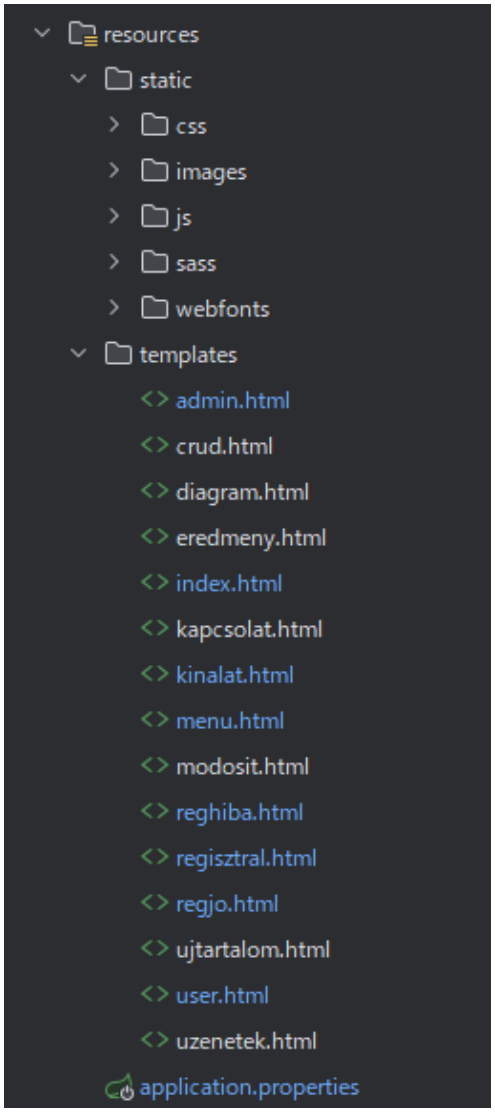
- **admin felhasználó:** admin@gmail.com, jelszó: jelszo1 (admin menüpontot csak admin láthatja)
- **felhasználó:** user@gmail.com, jelszó: jelszo2 (üzenet menüpontot csak regisztrált felhasználó láthatja)

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:





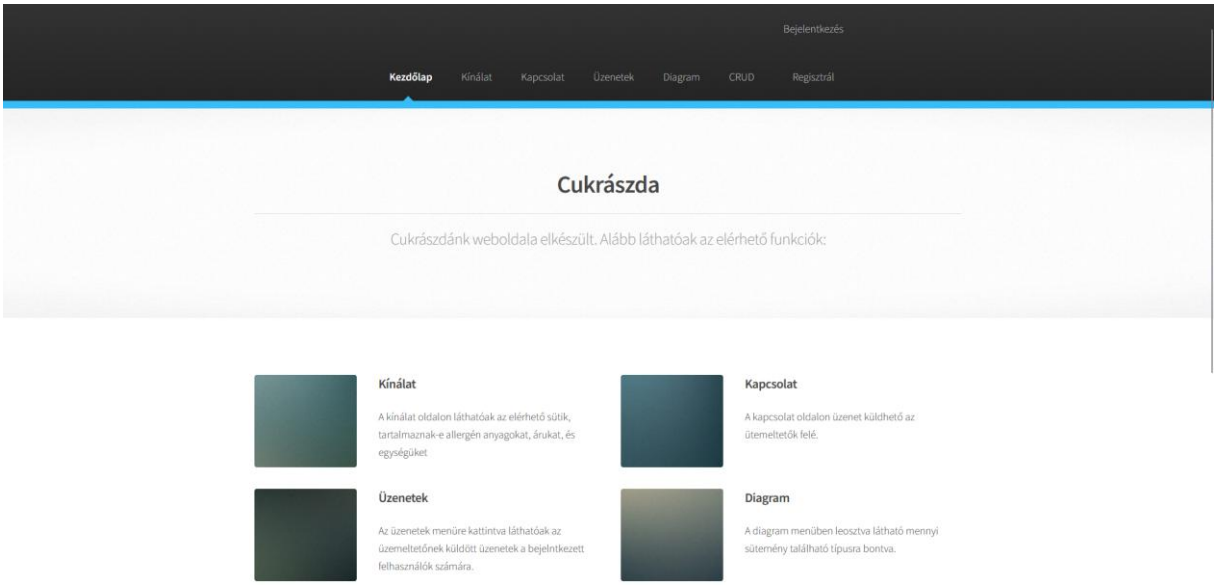
A com.example.gyakorlatbead-ban találhatóak az elkészített classok és interface-k, még a resources mappában található a static-on belül az összes formázással kapcsolatos fájl, még a Thymeleaf-et használó oldalak a templates mappában találhatóak.

A bejelentkezési rendszerrel kapcsolatos controller-ek a HomeController-ben, egyéb bejelentkezéssel 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 class-okban.

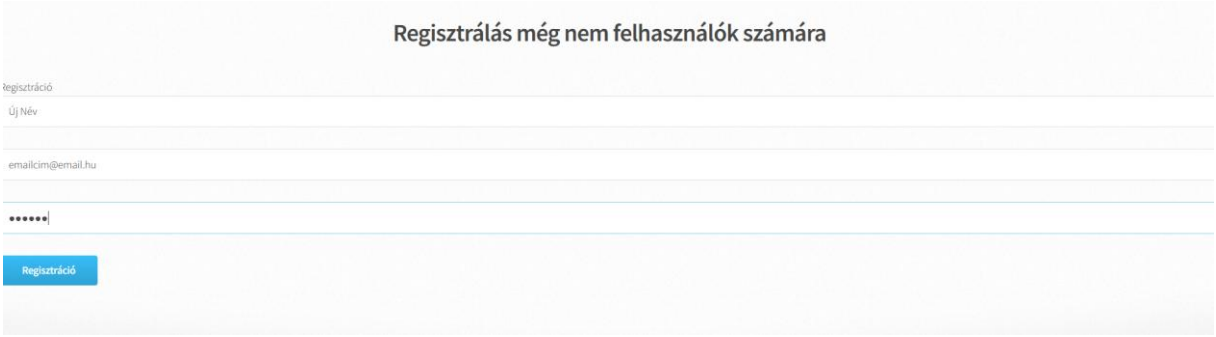
A githubon feltöltött beadando.sql fájl tartalmazza a beadando nevezetű adatbázisba importálandó táblákat, és adatokat amelyek szükségesek az alkalmazás futtatásához.

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



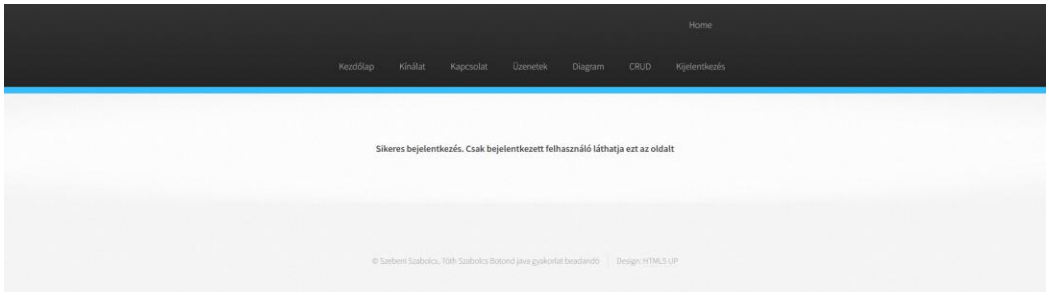
Elérhető a főoldalról a többi elkészített funkciót tartalmazott menüpont, és a regisztrálás, bejelentkezés gomb (megvalósítva az index.html-ben).

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

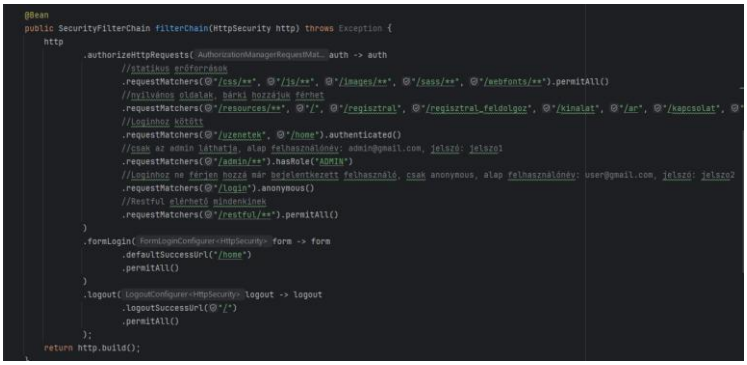


Regisztrálás után megjelenik a sikeres regisztráció, más esetben ha foglalt az email cím (regjo.html, ellenkezője reghiba.html).

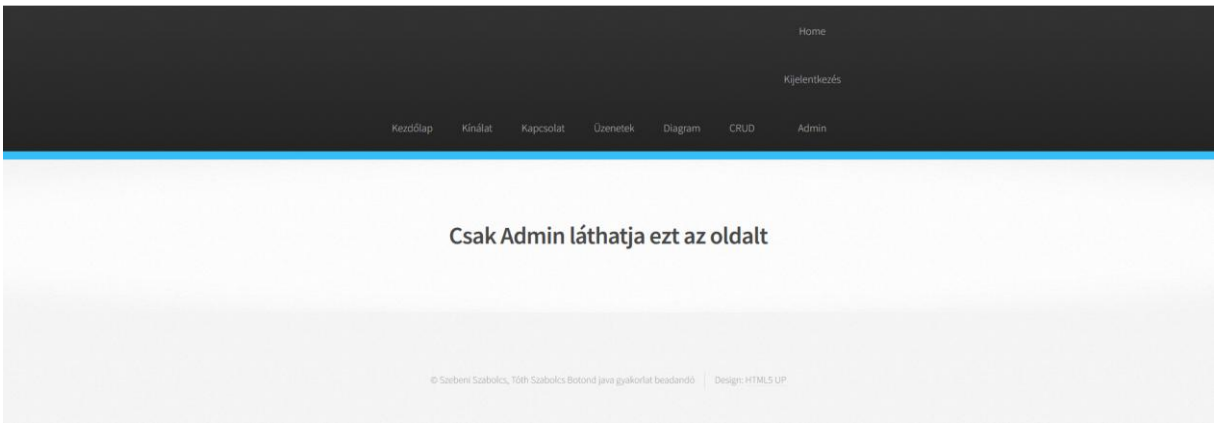
Ezután a felhasználó bejelentkezhet emaillel és jelszóval. Bejelentkezés után egy olyan oldalt lát, amelyet csak bejelentkezett felhasználó láthat (user.html).



Elérhető oldalakat kezelő kód (WebSecurityConfig class):



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 egységben vannak mérve. Az oldal megvalósítása a kinalat.html-ben.

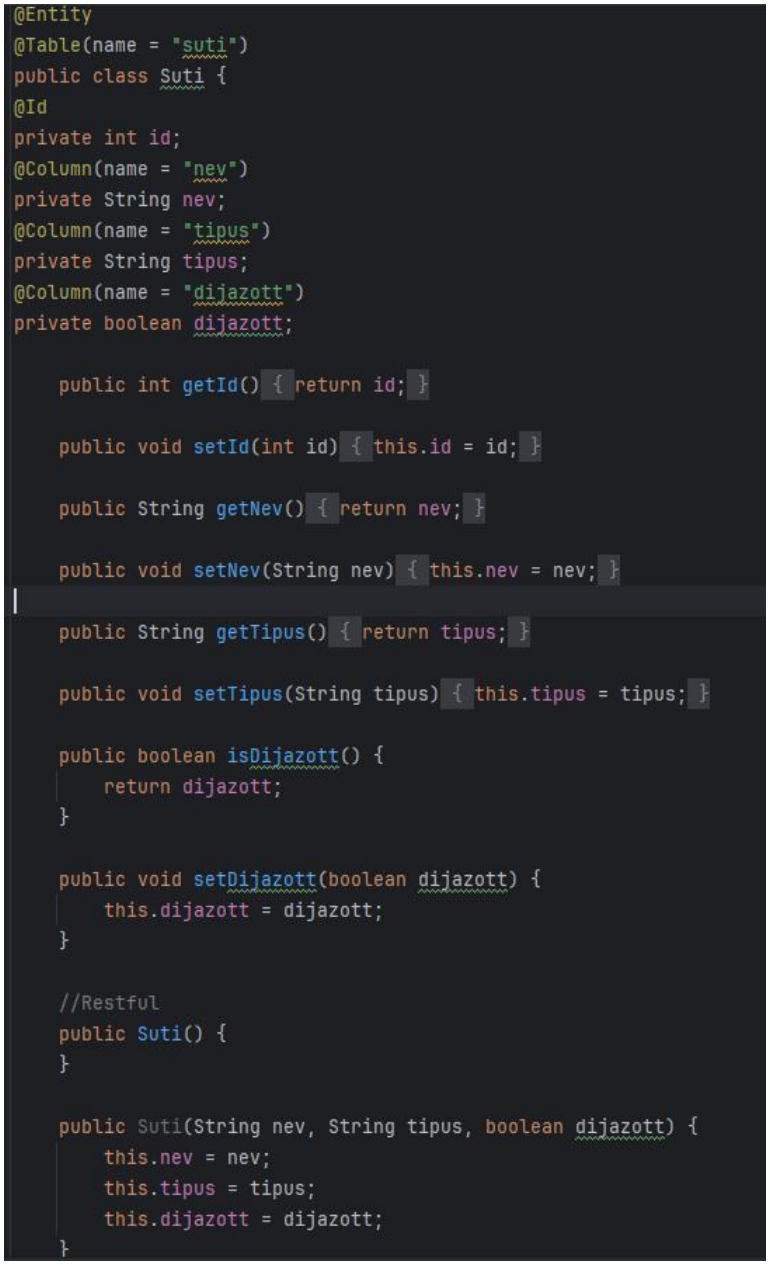
Süti	Tartalom	Ár	Egység
Süni		300	db
Gesztenyealagút		500	db
Sajtos pogácsa	sd	3300	kg
Diós-mákos		3500	koszorú
Módosított süti		9000	12 szeletes
Citrom	L	3900	8 szeletes
Citrom	L	7400	16 szeletes
Citrom	L	11400	24 szeletes
Eszterházy		490	db
Rákóczi-túrós		400	db
Meggyes kocka		450	db
Legényfogó		12100	24 szeletes
Legényfogó		4300	8 szeletes
Legényfogó		8200	16 szeletes

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

SutiReposity:

```
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();
}
```

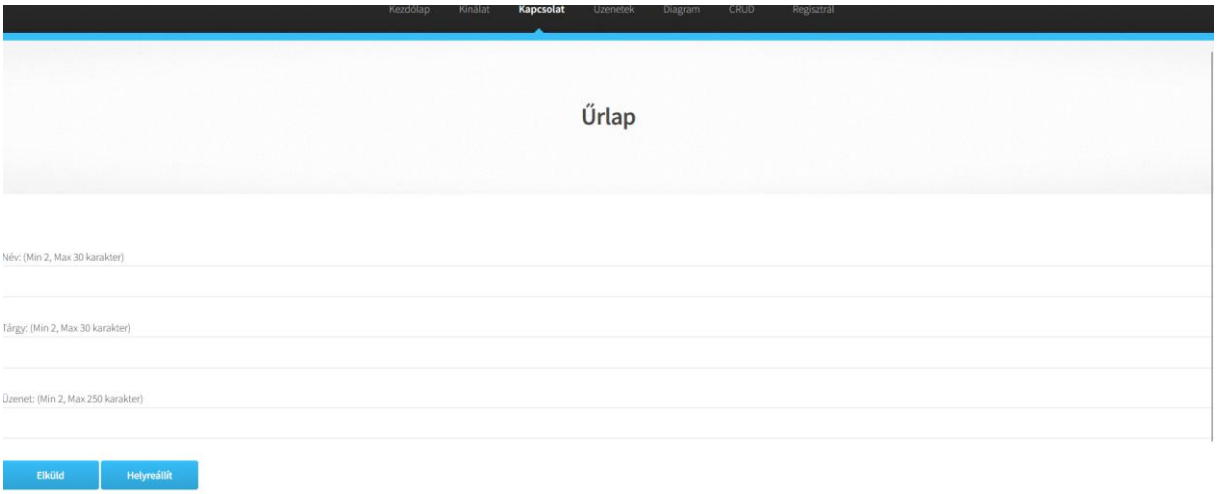
Suti class:



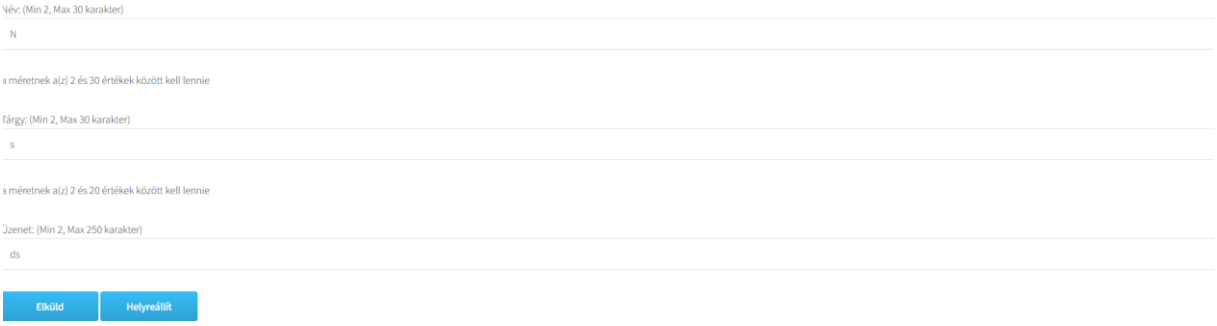
```java
@Entity
@Table(name = "suti")
public class Suti {
@Id
private int id;
@Column(name = "nev")
private String nev;
@Column(name = "tipus")
private String tipus;
@Column(name = "dijazott")
private boolean dijazott;

    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 UserRepo-ba, és az Uzenet és UzenetRepo-ban.

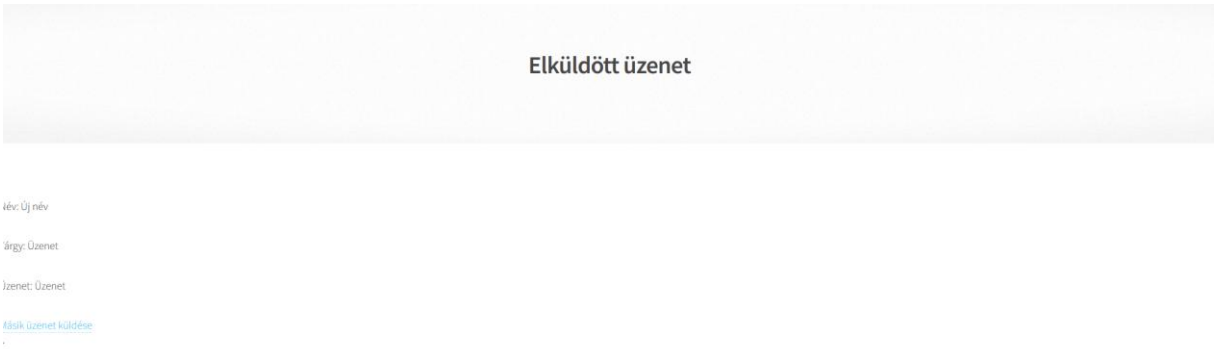
Kapcsolat menüpont (megvalósítva a kapcsolat.html-ben):



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



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



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

Uzenet class:

```java
@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; }
}
```

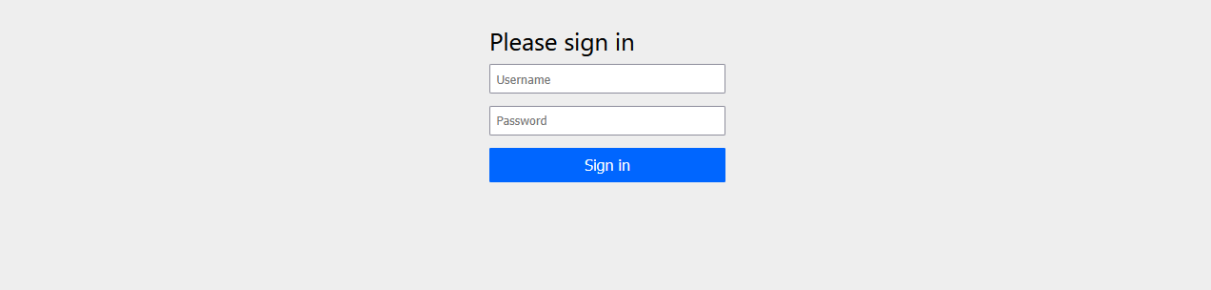
UzenetRepository:

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

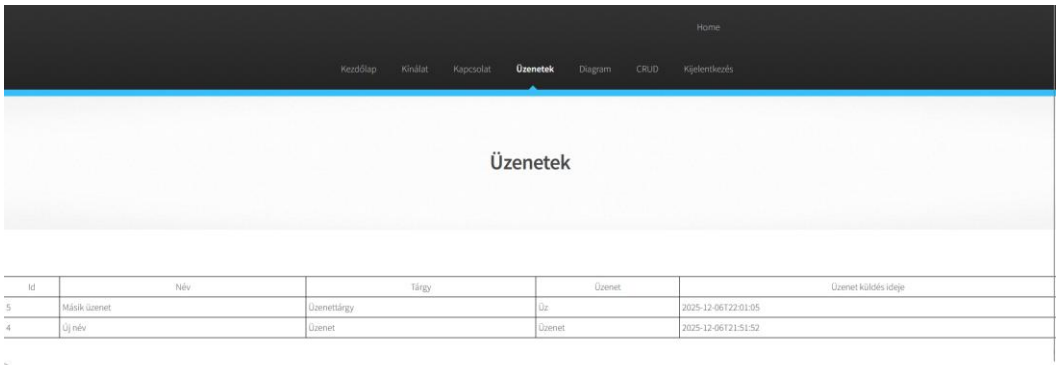
Kapcsolat.html:

```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):



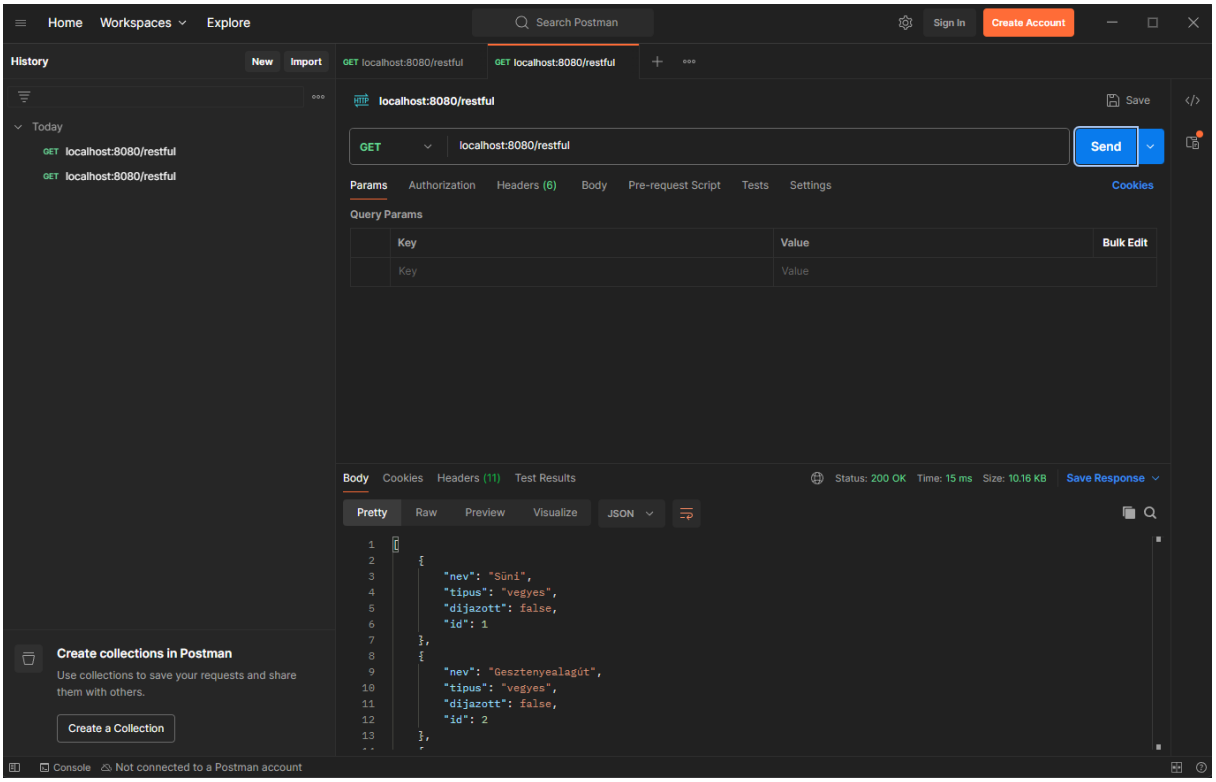
uzenetek.html:

```
<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>
```

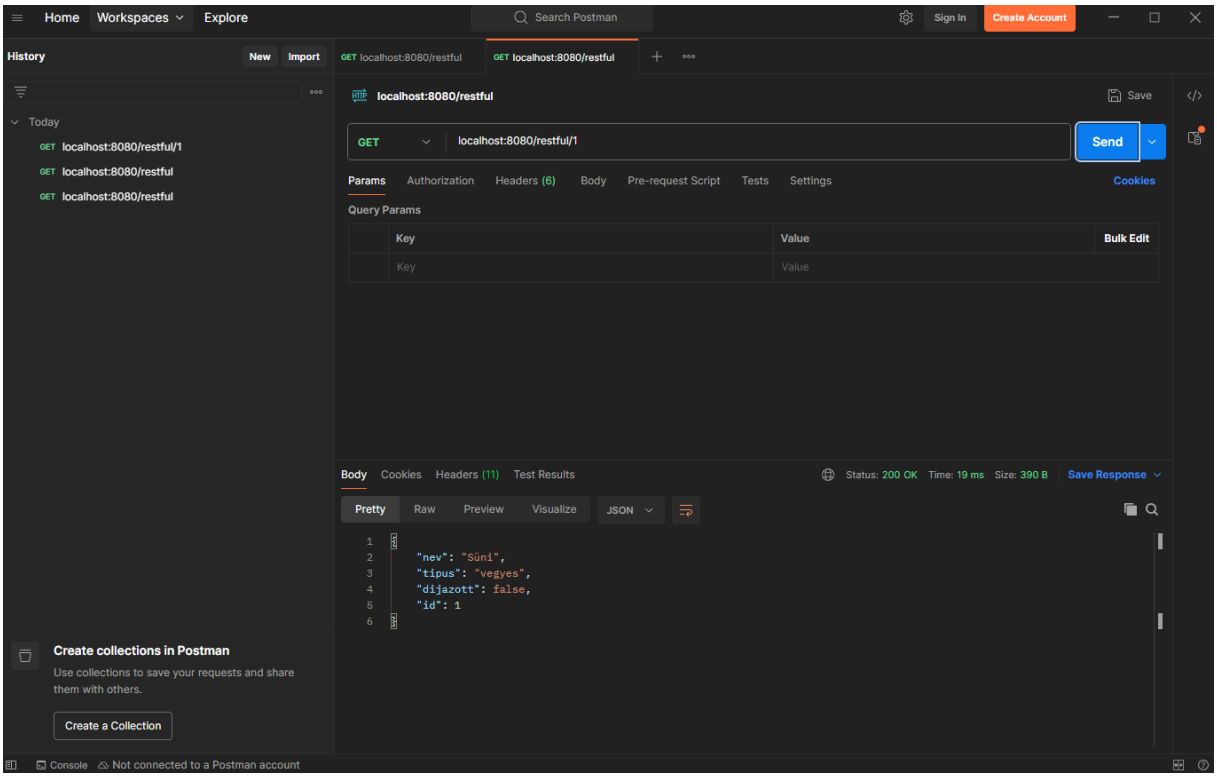
Restful menü: A Postman segítségével elvégzett kérések controllere a SutiController-ben vannak elkészítve:

```
@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 a -> {
                    a.setNev(adatSuti.getNev());
                    a.setTipus(adatSuti.getTipus());
                    a.setDijazott(adatSuti.isDijazott());
                    return sutirepo1.save(a);
                })
                .orElseGet(() -> {
                    adatSuti.setId(id);
                    return sutirepo1.save(adatSuti);
                });
    }

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

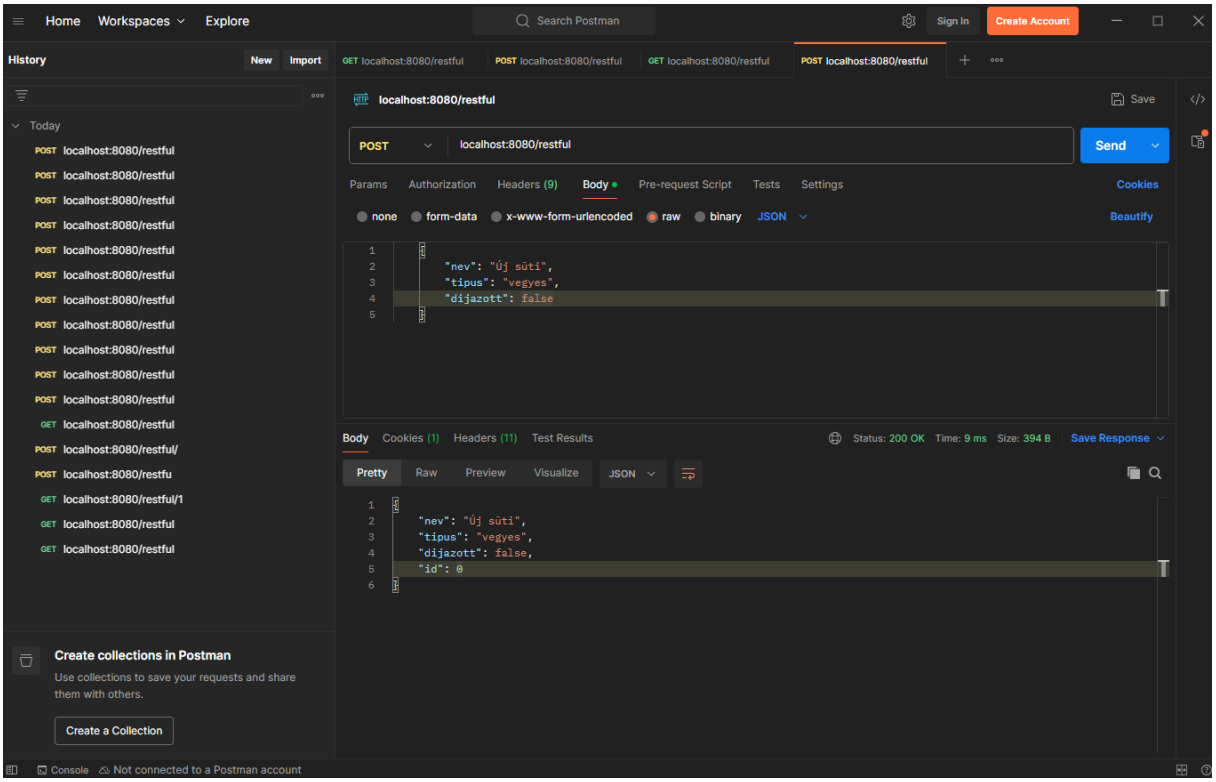
Postman segítségével get:



Postman segítségével 1-es id-jú süti get:



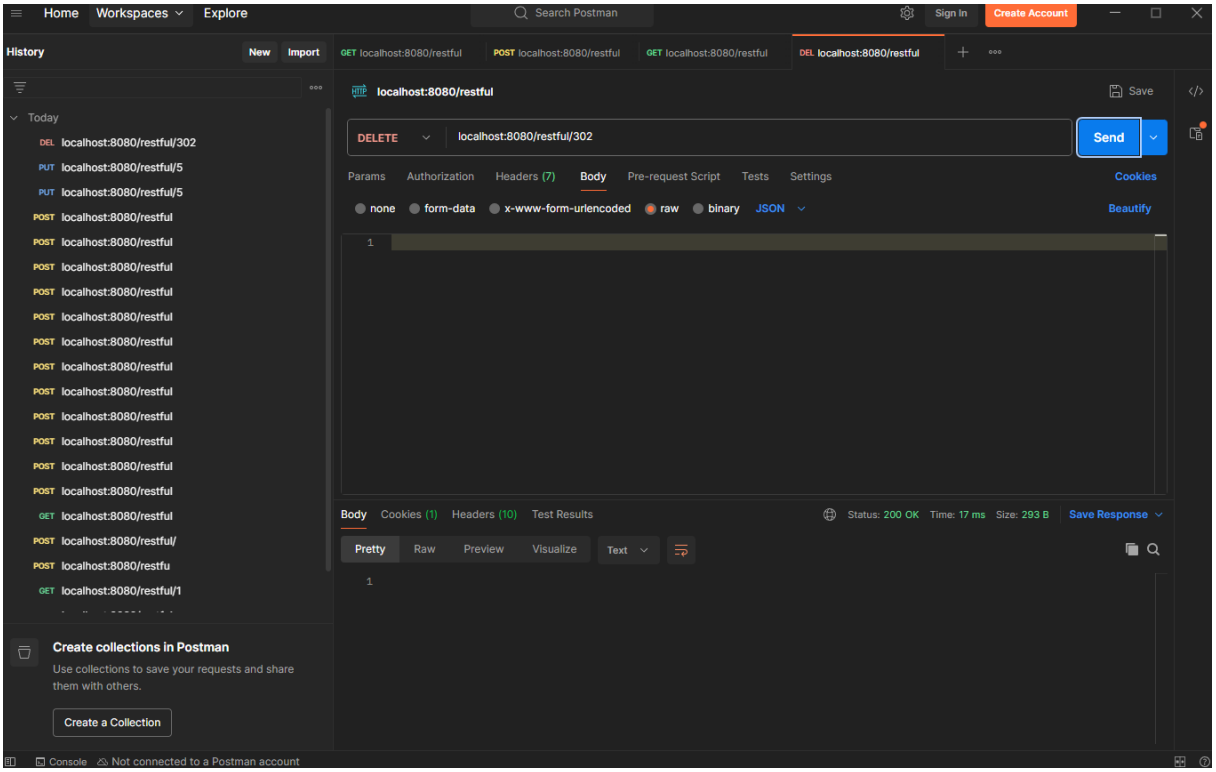
Postman segítségével új süti hozzáadása:



A MySQL adatbázis táblájában megjelenik az új süti:

	id	nev	tipus	dijazott
Módosítás Másolás Törlés	126	Gesztenyés-karamellás	bejgli	0
Módosítás Másolás Törlés	127	Gesztenyés szív	édes teasütemény	0
Módosítás Másolás Törlés	128	Ropi	sós teasütemény	0
Módosítás Másolás Törlés	129	Paleolit étcsokoládé	különleges torta	0
Módosítás Másolás Törlés	130	Túrós	pite	0
Módosítás Másolás Törlés	131	Ischler	vegyes	0
Módosítás Másolás Törlés	132	Lúdláb	tortaszelet	0
Módosítás Másolás Törlés	133	Csokoládémousse	tortaszelet	0
Módosítás Másolás Törlés	134	Dió	torta	0
Módosítás Másolás Törlés	135	Krémes	krémes	0
Módosítás Másolás Törlés	136	Mini ischler	édes teasütemény	0
Módosítás Másolás Törlés	137	Paleolit étcsokoládé	tortaszelet	0
Módosítás Másolás Törlés	138	Tejfölös túrós hajtogatott	sós teasütemény	0
Módosítás Másolás Törlés	139	Mákos guba	torta	0
Módosítás Másolás Törlés	302	Új süti	vegyes	0

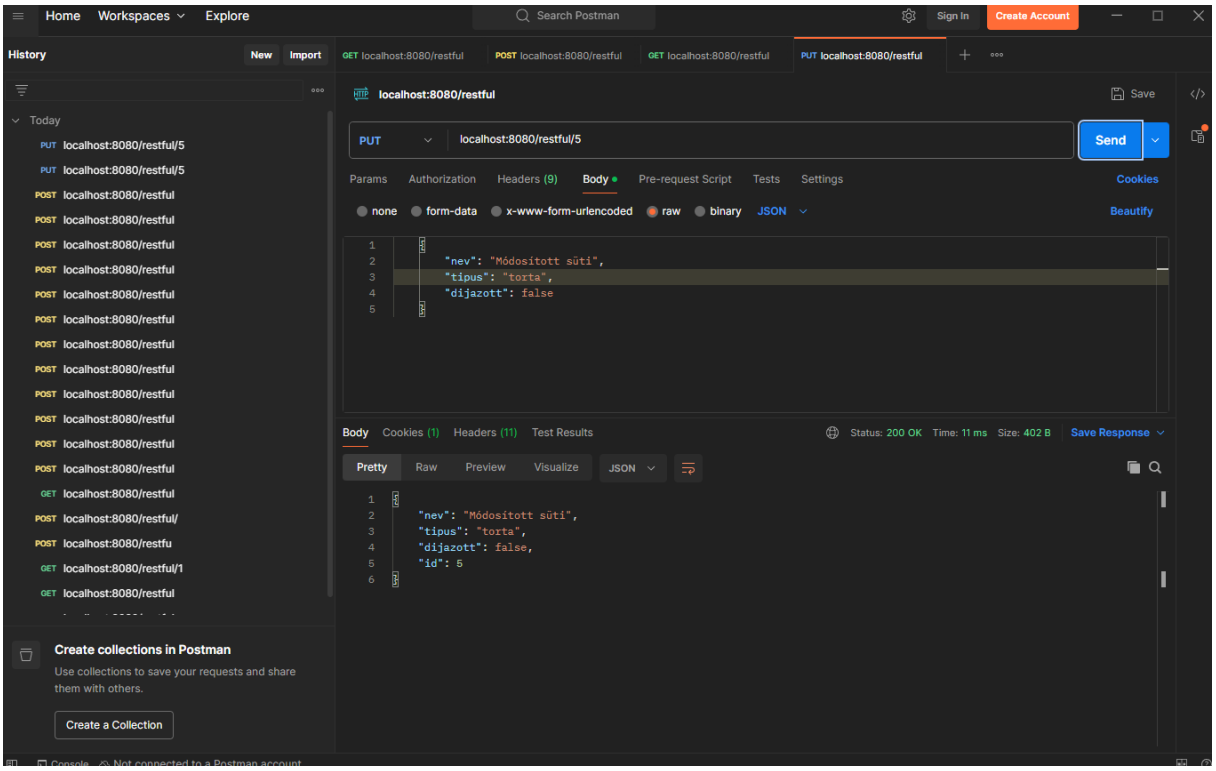
Postman segítségével süti törlése:



Törlés után a megadott id-jú süti törlődik a táblából

		id	nev	tipus	dijazott
☐	Módosítás Másolás Törlés	126	Gesztenyés-karamellás	bejgli	0
☐	Módosítás Másolás Törlés	127	Gesztenyés szív	édes teasütemény	0
☐	Módosítás Másolás Törlés	128	Ropi	sós teasütemény	0
☐	Módosítás Másolás Törlés	129	Paleolit étcsokoládé	különleges torta	0
☐	Módosítás Másolás Törlés	130	Túrós	pite	0
☐	Módosítás Másolás Törlés	131	Ischler	vegyes	0
☐	Módosítás Másolás Törlés	132	Lúdláb	tortaszelet	0
☐	Módosítás Másolás Törlés	133	Csokoládémousse	tortaszelet	0
☐	Módosítás Másolás Törlés	134	Dió	torta	0
☐	Módosítás Másolás Törlés	135	Krémes	krémes	0
☐	Módosítás Másolás Törlés	136	Mini ischler	édes teasütemény	0
☐	Módosítás Másolás Törlés	137	Paleolit étcsokoládé	tortaszelet	0
☐	Módosítás Másolás Törlés	138	Tejfölös túrós hajtogatott	sós teasütemény	0
☐	Módosítás Másolás Törlés	139	Mákos guba	torta	0

Postman segítségével süti módosítása:



Ezután módosítódik a MySQL adatbázis táblájában az adott süti:

		id	nev	tipus	dijazott
☐	Módosítás Másolás Törlés	1	Süni	vegyes	0
☐	Módosítás Másolás Törlés	2	Gesztenyealagút	vegyes	0
☐	Módosítás Másolás Törlés	3	Sajtos pogácsa	sós teasütemény	0
☐	Módosítás Másolás Törlés	4	Diós-mákos	bejgli	0
☐	Módosítás Másolás Törlés	5	Módosított süti	torta	0
☐	Módosítás Másolás Törlés	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):

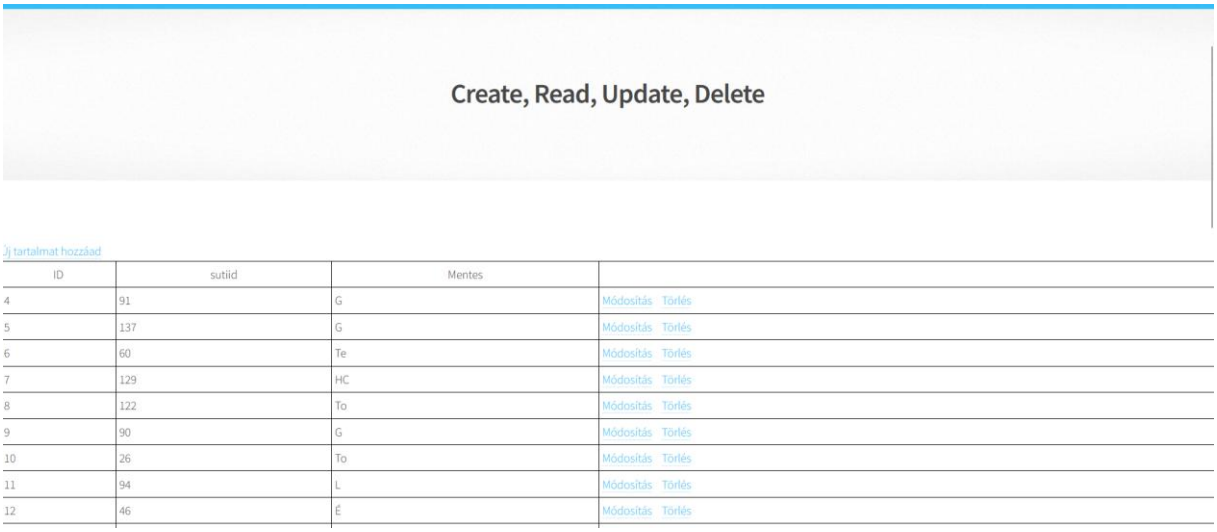
Az OtherControllerben lévő kód:

```
//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";
}
```

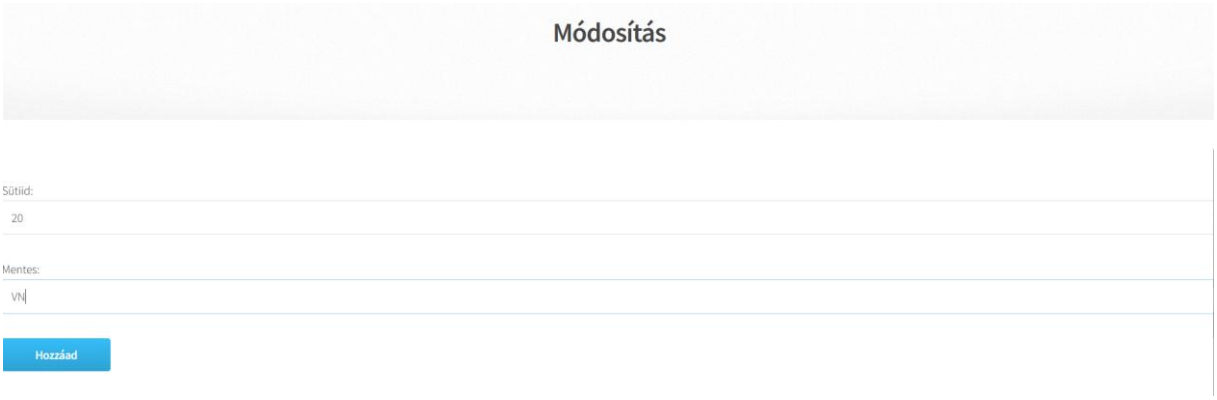
Diagram.html-ben ezért felelős kód:

```
<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).



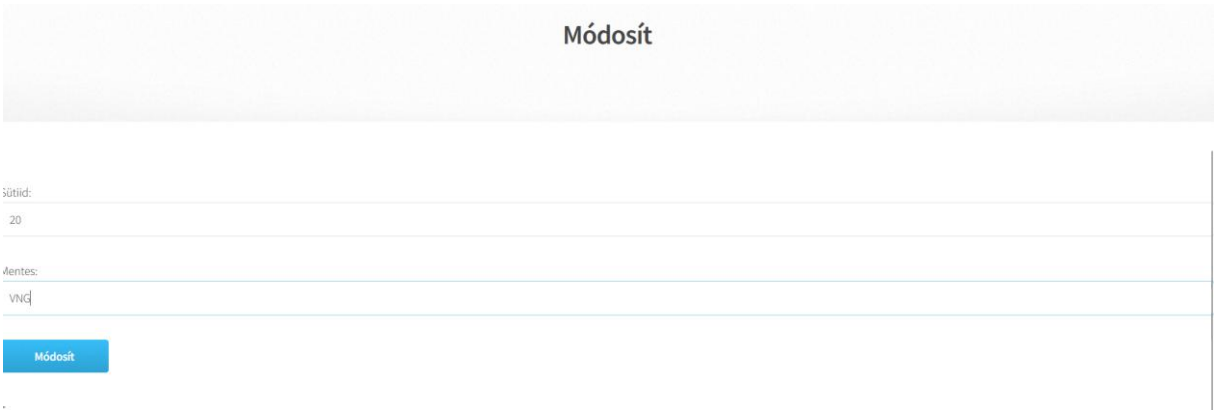
Új hozzáadásnál meg kell adni egy létező süti id-ját, és milyen legyen a mentes menü:



Ezután hozzáadásra kerül az adott új elem:

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

A módosításra nyomva módosítható:



Ezután megjelenik a módosítás a táblázatban:

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
50	20	VNG	Módosítás Törlés

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

10	90	To	Módosítás Törlés
11	20	HC	Módosítás Törlés
12	63	G	Módosítás Törlés
14	129	L	Módosítás Törlés
15	15	É	Módosítás Törlés
16	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>Sütiid:<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="${tartalom}" method="post">
            <input type="hidden" th:field="*{id}"/>
            <p>Sütiid:<input type="text" th:field="*{sutiid}" value= "${tartalom.sutiid}" required/></p>
            <p>Mentes:<input type="text" th:field="*{mentes}" value= "${tartalom.mentes}" required/></p>
            <p><input type="submit" value="Módosit" /></p>
        </form>
    </div>>
</section>
```

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

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

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

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

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

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

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