JAVA Alkalmazások
Gyakorlati beadandó

Hoki Attila – F2XVKV
Koltai Armand - GYBASN

2024

Tartalomjegyzék

1. GITHUB REPOSITORY ÉS BEÜZEMELÉS 2

1.1. APPLICATION.PROPERTIES 2

2. MUNKAMEGOSZTÁS 2

2.1. KOLTAI ARMAND (GYBASN) 2
2.2. HOKI ATTILA (F2XVKV) 3

3. FELÜLET BEMUTATÁSA 3

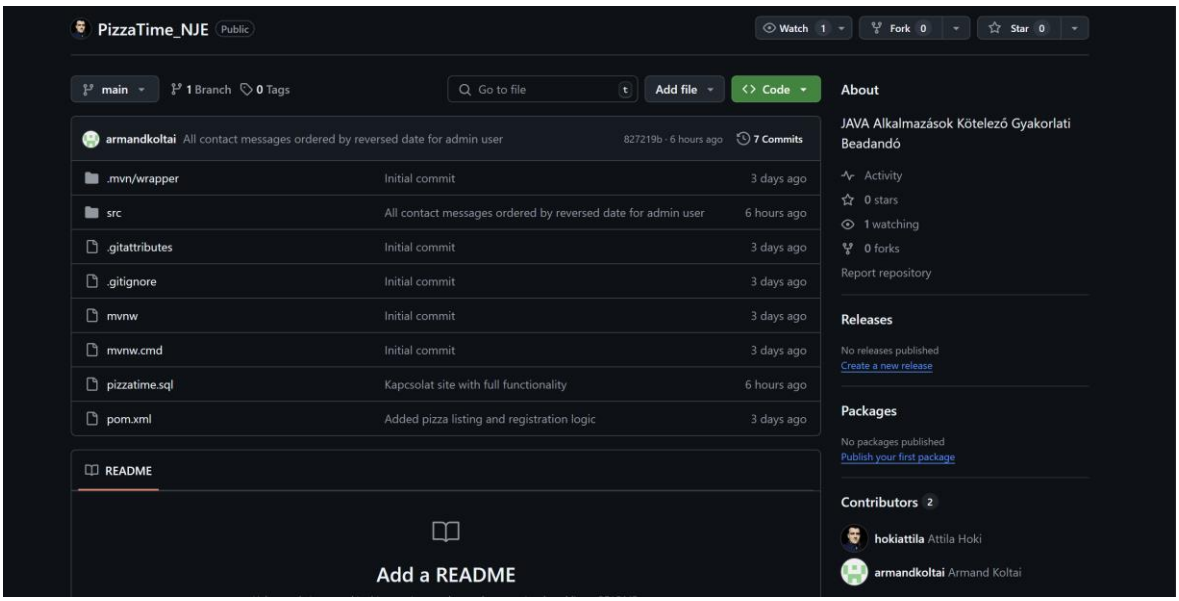
3.1. PROJEKT STRUKTÚRA 3
3.2. WEBOLDAL 3
3.2.1. Főoldal 3
3.2.2. Ajánlataink 4
3.2.3. Bejelentkezés 4
3.2.4. Regisztráció 5
3.2.5. Kapcsolat menüpont 6
3.2.6. Rendelés leadás 6
3.2.7. Admin menüpont 7
3.3. ADATBÁZIS 8

4. CONTROLLEREK 9

4.1. SECURITY KONFIGURÁCIÓK: 12

1. GitHub Repository és Beüzemelés

A projektet a https://github.com/hokiattila/PizzaTime_NJE/ linken lehetséges elérni



1.1. application.properties

```
spring.application.name=PizzaTime server.port=5555
spring.web.resources.static-locations=classpath:/static/
spring.thymeleaf.prefix=classpath:/templates/
spring.thymeleaf.suffix=.html
spring.datasource.url=jdbc:mysql://localhost:3306/pizzatime
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=validate
logging.level.root=INFO
logging.level.javagyakorlat.pizzatime.controller=DEBUG
```

Az oldalt az „**5555**”-ös porton kell elindítani.

2. Munkamegosztás

2.1. Koltai Armand (GYBASN)

- Frontend
- Kapcsolat űrlap
- Adatbázis módosítások, ContactForm tábla
- Admin menü - üzenetek kilistázása
- 3 különböző táblából adat megjelenítés

- Dokumentáció

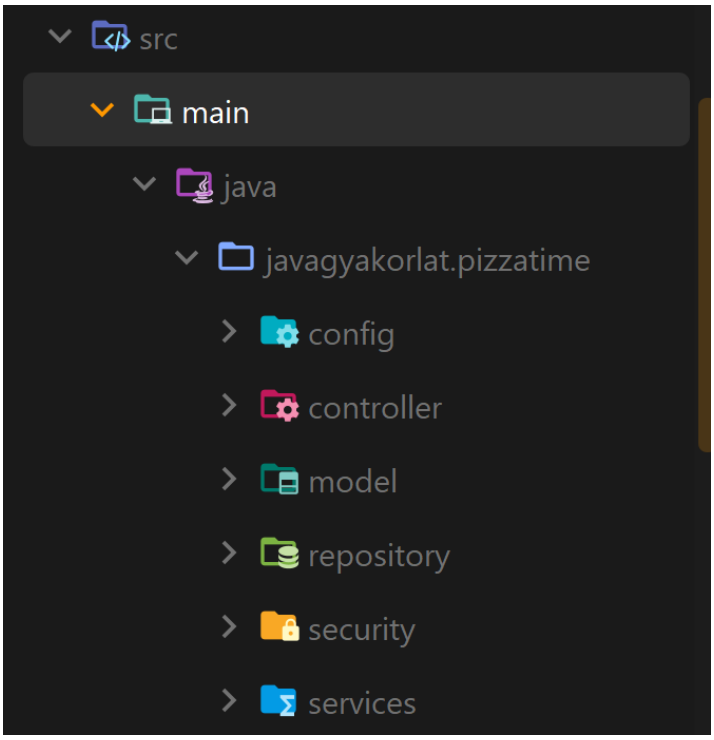
2.2. Hoki Attila (F2XVKV)

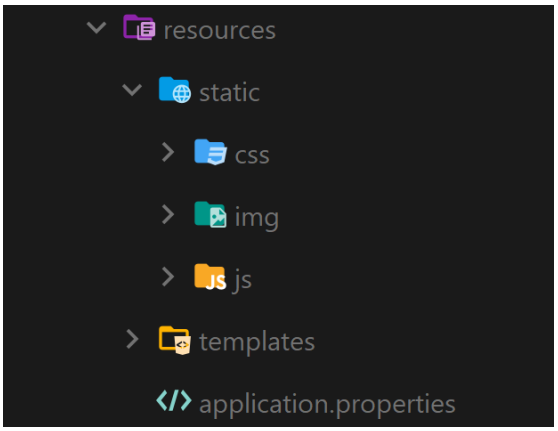
- Projekt inicializálás
- Login / Regisztráció / Authentikáció
- Projekt struktúra megalkotása – Alap Modellek, Kontrollerek
- 3 különböző felhasználó, és azokhoz tartozó jogosultságok
- Navigációs menü és login logika
- Teszt adatok a táblában

3. Felület bemutatása

3.1. Projekt struktúra

Minden fájl megtalálható az elnevezésnek megfelelő mappában. A projekt követi az MVC struktúrát .

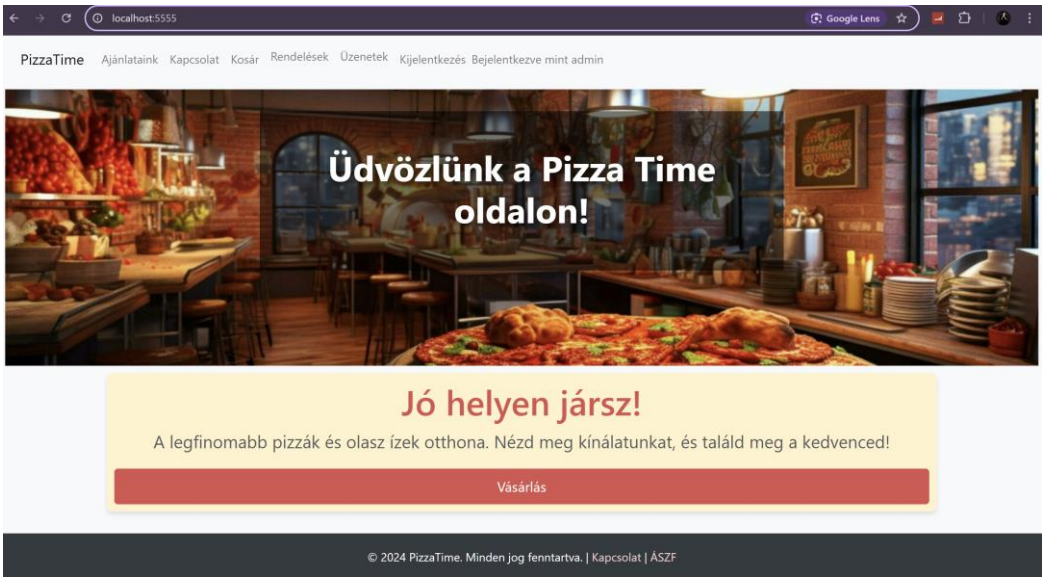




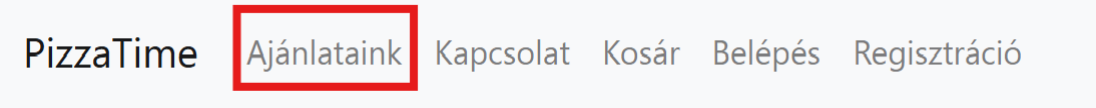
3.2. Weboldal

3.2.1. Főoldal

A főoldalon a látogatókat egy hívogató oldal várja. Az oldal egyetlen funkciója a felhasználók üdvözlése, funkciója egyedül a vásárlás oldalra való átirányítás.

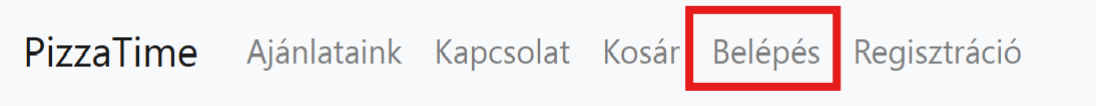


3.2.2. Ajánlataink



Ezen az oldalon lehetséges pizzákat tenni a kosárba, amiket később ezzel meg lehet vásárolni

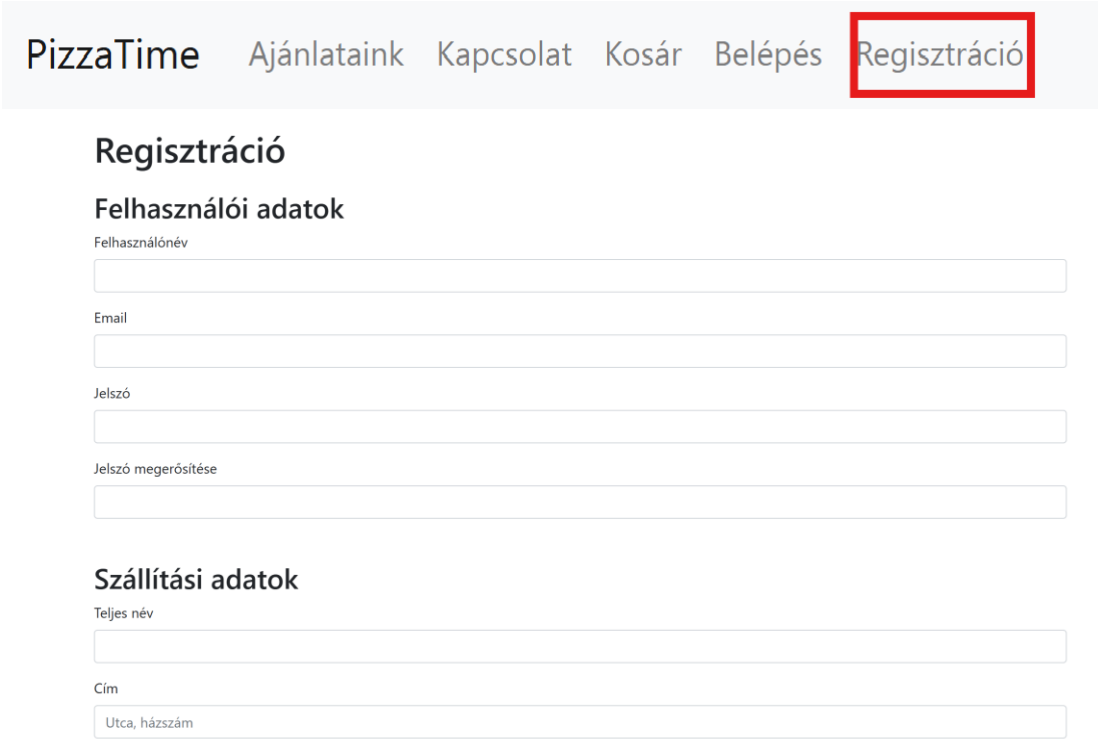
3.2.3. Bejelentkezés

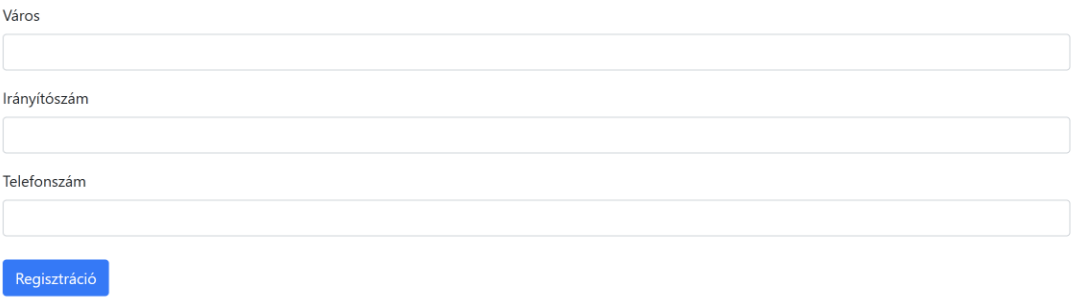


Frontend és Backend ellenőrzések is vannak az adatok validálására. A bejelentkezés oldal nyílik meg minden alkalommal, ha a felhasználó olyan helyre szeretne navigálni, amire nincs jogosultsága.

3.2.4. Regisztráció

Itt tudja a felhasználó megadni az adatait, és regisztrálni az oldalra. Az oldal ezeket az adatokat fogja később használni az adatbázisból, így nem kell minden egyes rendelésnél megadni a pontos lakcímet, és a felhasználó egyéb adatait.



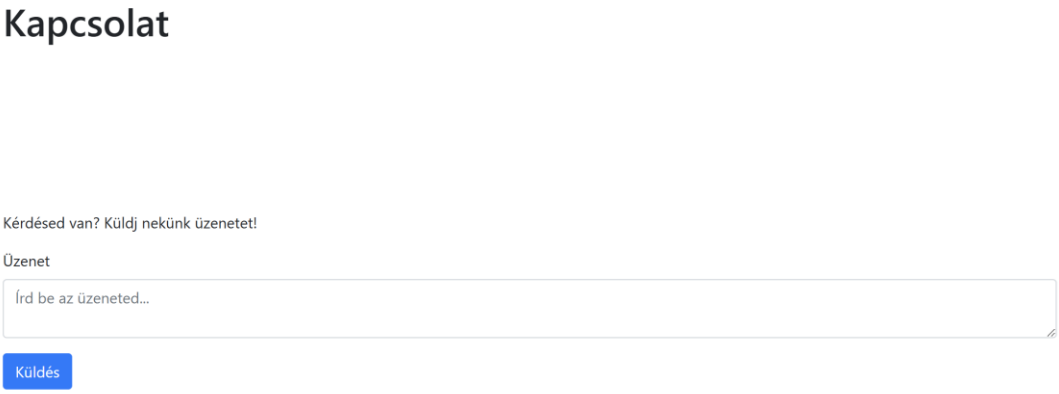


A viselkedésnek elvártan a Regisztráció / Bejelentkezés opció csak azoknak jelenik meg, akik nem bejelentkezett felhasználók.

3.2.5. Kapcsolat menüpont

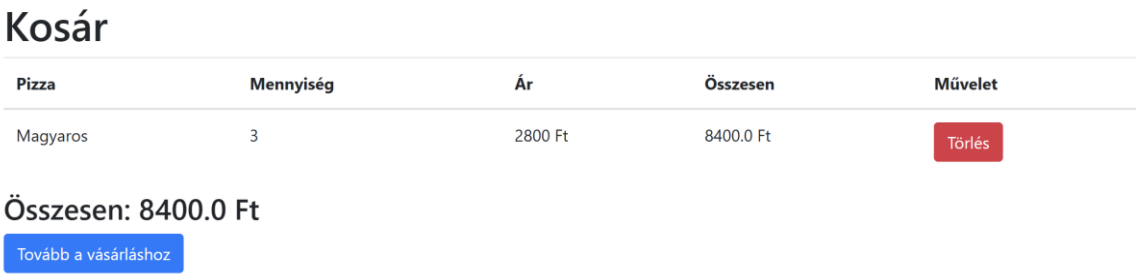
Minden felhasználó számára elérhető. Bejelentkezett felhasználó és vendég is tud üzenetet küldeni. Bejelentkezett felhasználó esetén a nevet menti az adatbázisba, Vendég esetén pedig fixen a „Vendég” stringet.

A mentett adatok a **contact_form** táblába kerülnek.

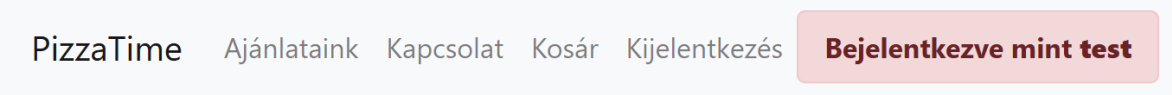


3.2.6. Rendelés leadás

Egyedül a **bejelentkezett felhasználó** tud elemeket adni a kosárba, és rendelést leadni.

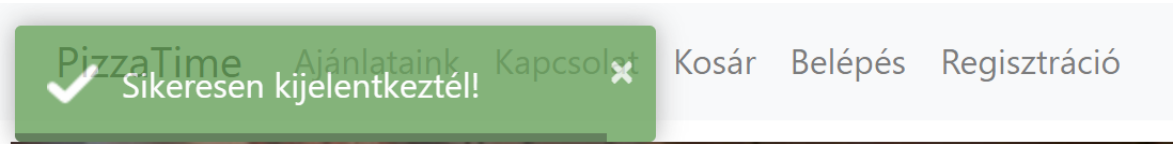


Ilyenkor a navigációs bar is máshogy néz ki:



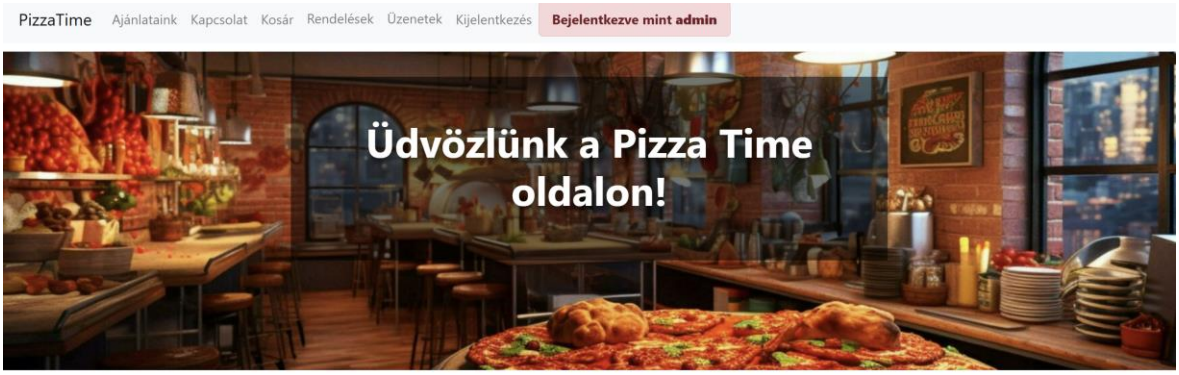
Belépés és Regisztrációs opció helyett „Kijelentkezés” opciót kapunk, és kiírja milyen néven vagyunk bejelentkezve.

Végrehajtott be / ki jelentkezés illetve regisztrációs után felugró üzenetet is kapunk a tevékenység sikerességéről:



3.2.7. Admin menüpont

Mind a három táblából szerepel adat a táblázatban. Alapból az „orders” táblát és annak adatait jelenítjük meg. A pizza kép a „pizza” táblázatból van lekérve, a „felhasználó email” pedig a „user” táblából van megjelenítve.



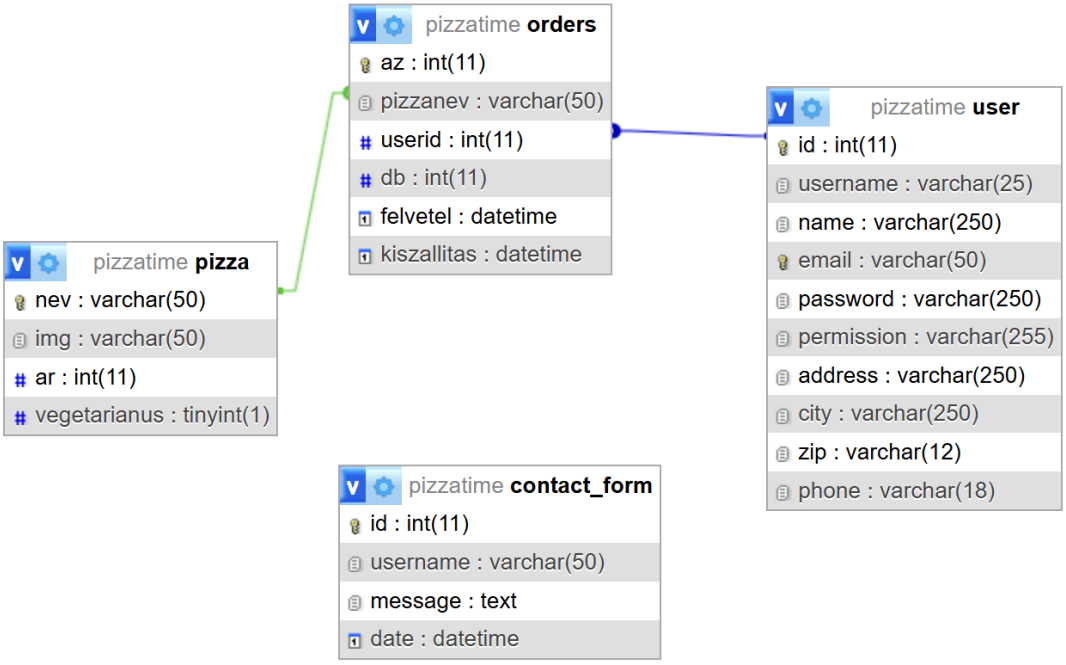
Üzenetek

ID	Küldő	Üzenet	Küldés ideje
12	test	Most be van jelentkezve :)	2024-12-09T17:53:58
11	Vendég	Szia attila	2024-12-09T17:53:31
10	test	Ez meg egy test üzenet legyen!!!!	2024-12-09T17:25:29

Itt láthatóak az üzenetek az admin menüben, a legújabb legelöl sorrendben.

3.3. Adatbázis

Adatbázis tervező nézet



Adatbázis struktúra

```
CREATE DATABASE IF NOT EXISTS Pizzatime;
USE Pizzatime;

-- Felhasználók tábla
CREATE TABLE User (
    id INT PRIMARY KEY AUTO_INCREMENT,
```

```
    username VARCHAR(25) NOT NULL,
    name VARCHAR(250) NOT NULL,
    email VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(250) NOT NULL,
    permission VARCHAR(255) DEFAULT 'user',
    address VARCHAR(250) NOT NULL,
    city VARCHAR(250) NOT NULL,
    zip VARCHAR(12) NOT NULL,
    phone VARCHAR(18) NOT NULL
);

-- Pizza tábla
CREATE TABLE Pizza (
    nev VARCHAR(50) PRIMARY KEY,
    img VARCHAR(50) NOT NULL,
    ar INT NOT NULL,
    vegetarianus BOOLEAN DEFAULT FALSE
);

-- Rendelések tábla
CREATE TABLE Orders (
    az INT PRIMARY KEY AUTO_INCREMENT,
    pizzanev VARCHAR(50) NOT NULL,
    userid INT NOT NULL,
    db INT NOT NULL,
    felvetel DATETIME NOT NULL,
    kiszallitas DATETIME NULL,
    FOREIGN KEY (pizzanev) REFERENCES Pizza(nev) ON DELETE CASCADE,
    FOREIGN KEY (userid) REFERENCES User(id) ON DELETE CASCADE
);

CREATE TABLE contact_form (
    id INT PRIMARY KEY AUTO_INCREMENT,   -- Azonosító
    username VARCHAR(50) DEFAULT 'Vendég',   -- Felhasználónév, ha nincs, akkor 'anonymous'
    message TEXT NOT NULL,               -- Üzenet
    date DATETIME NOT NULL               -- Üzenet küldésének dátuma
);
```

Teszteléshez használt adatok

```
INSERT INTO Pizza (nev, img, ar, vegetarianus) VALUES
('Sajtos', 'sajtos.jpg', 2000, TRUE),
('Sonkás', 'sonkas.jpg', 2300, FALSE),
('Sonka-kukoricás', 'sonka_kukoricas.jpg', 2500, FALSE),
('Magyaros', 'magyaros.jpg', 2800, FALSE),
('Zöldséges', 'zoldseges.jpg', 2400, TRUE);

INSERT INTO User (username, name, email, password, permission, address, city, zip, phone)
VALUES ('admin','Admin','admin@admin.com',
'$2a$10$QKfV4wdkHjYe/Yf2jRXHxenaq.c.hjQLIMjqP5PcJz618snVijtwK','admin','-', '-', '-','-');
INSERT INTO User (username, name, email, password, permission, address, city, zip, phone)
VALUES ('test', 'Test', 'test@test.com',
'$2a$10$ikvaQjS7PKifgTDOjd9vy.3pgtswyRGkt6TyWfiYFjYDZdcN/6wei','user','-', '-', '-','-');
```

4. Controllerek

Például a rendelés a következő képpen épül fel:

controllers/OrdersController.java

```
package javagyakorlat.pizzatime.controller;

import jakarta.servlet.http.HttpSession;
```

```java
import javagyakorlat.pizzatime.model.Orders;
import javagyakorlat.pizzatime.model.User;
import javagyakorlat.pizzatime.services.OrderService;
import javagyakorlat.pizzatime.services.PizzaService;
import javagyakorlat.pizzatime.services.UserService;
import org.hibernate.query.Order;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import java.util.List;

@Controller public class OrdersController {

    @Autowired     private OrderService orderService;
    @Autowired     private UserService userService;
    @Autowired     private PizzaService pizzaService;

    @GetMapping("/orders")
    public String ordersPage(HttpSession session, Model model) {
        User user = (User) session.getAttribute("user");


        List<Orders> orders = orderService.getAllOrders();
        for(Orders order : orders)
        {
            order.pizzakep = pizzaService.getPizzaByNev(order.getPizzanev()).getImg();
            order.useremail = userService.findUserEmailById(order.getUserid());
        }
        model.addAttribute("orders", orders);
        model.addAttribute("user", user);
        return "orders";
    }
}
```

model/Orders.java

```java
package javagyakorlat.pizzatime.model;

import jakarta.persistence.*;
import java.time.LocalDateTime;

@Entity public class Orders {

    @Id     @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int az;

    @Column(nullable = false)
    private String pizzanev;

    @Column(nullable = false)
    private int userid;

    @Column(nullable = false)
    private int db;

    @Column(nullable = false)
    private LocalDateTime felvetel;

    private LocalDateTime kiszallitas;

    @Transient     public String pizzakep;
    @Transient     public String useremail;
```

```
    // Getters és Setters
    public int getAz() {
        return az;
    }

    public void setAz(int az) {
        this.az = az;
    }

    public String getPizzanev() {
        return pizzanev;
    }

    public void setPizzanev(String pizzanev) {
        this.pizzanev = pizzanev;
    }

    public int getUserid() {
        return userid;
    }

    public void setUserid(int userid) {
        this.userid = userid;
    }

    public int getDb() {
        return db;
    }

    public void setDb(int db) {
        this.db = db;
    }

    public LocalDateTime getFelvetel() {
        return felvetel;
    }

    public void setFelvetel(LocalDateTime felvetel) {
        this.felvetel = felvetel;
    }

    public LocalDateTime getKiszallitas() {
        return kiszallitas;
    }

    public void setKiszallitas(LocalDateTime kiszallitas) {
        this.kiszallitas = kiszallitas;
    }
}
```

Interfész a kapcsolathoz:

OrdersRepository.java

```
package javagyakorlat.pizzatime.repository;

import javagyakorlat.pizzatime.model.Orders;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository public interface OrdersRepository extends JpaRepository<Orders, Integer> {
}
```

orders.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Rendelések</title>
  <link th:href="@{/css/bootstrap.min.css}" rel="stylesheet">
</head>
<body>
<div th:replace="layout/navbar :: navbar"></div>
<div class="container mt-5">
  <h1>Rendelések</h1>
  <table class="table table-bordered table-striped">
    <thead>
    <tr>
      <th>Pizza név</th>
      <th>Pizza kép</th> <!-- user táblából adat --> <!-- pizza táblából adat -->
      <th>Felhasználó ID</th>
      <th>Felhasználó Email</th> <!-- user táblából adat --> <!-- pizza táblából adat -->
      <th>Darab</th>
      <th>Felvétel dátuma</th>
      <th>Kiszállítás dátuma</th>
    </tr>
    </thead>
    <tbody>
    <tr th:each="order : ${orders}">
      <td th:text="${order.pizzanev}"></td>
      <td><img th:src="@{'/img/' + ${order.pizzakep}}" alt="Pizza kép" style="width: 100px;
height: auto;"></td>
      <td th:text="${order.userid}"></td>
      <td th:text="${order.useremail}"></td>
      <td th:text="${order.db}"></td>
      <td th:text="${order.felvetel}"></td>
      <td th:text="${order.kiszallitas}"></td>
    </tr>
    </tbody>
  </table>
</div>
<script th:src="@{/js/bootstrap.bundle.js}"></script>
<!-- Footer -->
<div th:replace="layout/footer :: footer"></div>
</body>
</html>
```

A navbar és a footer külön fájlba ki vannak szervezve, hogy ne legyen ismétlődés, ezzel követve a DRY elvet.

4.1. Security konfigurációk:

security/CustomAuthenticationSuccessHandler:

Amikor egy felhasználó sikeresen bejelentkezik, ez a handler végrehajtja a következőket: megszerzi a bejelentkezett felhasználó adatait (UserDetails), és eltárolja azokat a session-ben a **"currentUser"** nevű attribútumként. Ez lehetővé teszi, hogy a rendszer más részei könnyen hozzáférjenek a bejelentkezett felhasználó adataihoz. Végül átirányítja a felhasználót a **"/home"** oldalra. Lényegében az a célja, hogy kezelje a bejelentkezés utáni állapotot, és biztosítsa, hogy a felhasználói adatok elérhetők legyenek.

```
package javagyakorlat.pizzatime.security;
```

```
import jakarta.servlet.FilterChain;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.AuthenticationSuccessHandler;
import org.springframework.stereotype.Component;


import javax.servlet.http.HttpSession;
import java.io.IOException;

@Component
public class CustomAuthenticationSuccessHandler implements AuthenticationSuccessHandler {

    @Override
    public void onAuthenticationSuccess(jakarta.servlet.http.HttpServletRequest request,
jakarta.servlet.http.HttpServletResponse response, FilterChain chain, Authentication
authentication) throws IOException {
        UserDetails userDetails = (UserDetails) authentication.getPrincipal();
        HttpSession session = (HttpSession) request.getSession();
        session.setAttribute("currentUser", userDetails);
        response.sendRedirect("/home");
    }

    @Override
    public void onAuthenticationSuccess(jakarta.servlet.http.HttpServletRequest request,
jakarta.servlet.http.HttpServletResponse response, Authentication authentication) throws
IOException {
        UserDetails userDetails = (UserDetails) authentication.getPrincipal();
        HttpSession session = (HttpSession) request.getSession();
        session.setAttribute("currentUser", userDetails);
        response.sendRedirect("/home");
    }
}
```

config/SecurityConfig

Ez a fájl egyedi sikeres hitelesítési kezelőt (AuthenticationSuccessHandler) valósít meg a Spring Security-ben. Azt csinálja, hogy amikor egy felhasználó sikeresen bejelentkezik, akkor megszerzi a felhasználó adatait (UserDetails), és ezeket eltárolja a session-ben a "currentUser" nevű attribútumban. Ezáltal más részeken könnyen elérhetővé válik a bejelentkezett felhasználó. Ezután átirányítja a felhasználót a "/home" oldalra. Lényegében arról gondoskodik, hogy a bejelentkezés után a felhasználó adatai legyenek elérhetők és a megfelelő helyre kerüljön.

```
package javagyakorlat.pizzatime.security;

import jakarta.servlet.FilterChain;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.AuthenticationSuccessHandler;
import org.springframework.stereotype.Component;


import javax.servlet.http.HttpSession;
import java.io.IOException;

@Component
public class CustomAuthenticationSuccessHandler implements AuthenticationSuccessHandler {

    @Override
```

```
    public void onAuthenticationSuccess(jakarta.servlet.http.HttpServletRequest request,
jakarta.servlet.http.HttpServletResponse response, FilterChain chain, Authentication
authentication) throws IOException {
        UserDetails userDetails = (UserDetails) authentication.getPrincipal();
        HttpSession session = (HttpSession) request.getSession();
        session.setAttribute("currentUser", userDetails);
        response.sendRedirect("/home");
    }

    @Override
    public void onAuthenticationSuccess(jakarta.servlet.http.HttpServletRequest request,
jakarta.servlet.http.HttpServletResponse response, Authentication authentication) throws
IOException {
        UserDetails userDetails = (UserDetails) authentication.getPrincipal();
        HttpSession session = (HttpSession) request.getSession();
        session.setAttribute("currentUser", userDetails);
        response.sendRedirect("/home");
    }
}
```