**SZABADKAI MŰSZAKI SZAKFŐISKOLA**

**PROJEKTUM**

**Web programozás tantárgyból**

|  |  |
| --- | --- |
| **HALLGATÓK**  **Bicskei Ádám 26223016**  **Balázs Ákos 26223030** | **MENTOR**  **prof. dr Zlatko Čović** |
| **SZABADKA, 2025.** | |

Tartalom

[1. A feladatkör meghatározása 3](#_Toc156903382)

[2. Kidolgozás 4](#_Toc156903383)

[3. Az adatbázis struktúrája (MySQL Workbench) 5](#_Toc156903384)

[4. A projekt működésének leírása 6](#_Toc156903385)

[5. Felhasznált irodalom 7](#_Toc156903386)

# A feladatkör meghatározása

**Projektfeladat meghatározása:**

A projekt célja egy modern, online asztalfoglalási rendszer létrehozása egy étterem számára. Az elkészült webalkalmazás lehetővé teszi, hogy a regisztrált és bejelentkezett felhasználók egyszerűen, néhány kattintással asztalt foglaljanak az étteremben a kívánt időpontra. A rendszer különböző jogosultsági szintekkel rendelkező felhasználókat kezel:

* **Vendégek (felhasználók):** csak regisztráció és bejelentkezés után tudnak foglalást létrehozni és megtekinteni a korábbi foglalásaikat.
* **Munkatársak:** jogosultak a foglalások kezelésére, azok jóváhagyására, módosítására vagy törlésére.
* **Adminisztrátorok:** teljes körű hozzáféréssel rendelkeznek a rendszerhez, beleértve a felhasználók, asztalok, időpontok és egyéb adminisztratív adatok kezelését is.

**A projekt webcíme:**

<https://lm.stud.vts.su.ac.rs/ETTEREM/views/guest_view.php>

# Kidolgozás(főbb kódrészletek)

Adatbázis kapcsolat és SMPT beállítása:

<?php  
  
const *PARAMS* = [  
 "SITE" => 'https://lm.stud.vts.su.ac.rs/ETTEREM/register/',  
 "HOST" => 'localhost',  
 "USER" => 'lm',  
 "PASS" => 'QsMvuBdmB3sS74j',  
 "DBNAME" => 'lm',  
 "CHARSET" => 'utf8mb4'  
];  
  
const MAILTRAP = [  
 "USERNAME" => '0dd68b82818fbc', // your MailTrap username  
 "PASSWORD" => '4033644b873be2', // your MailTrap password  
 "HOST" => 'sandbox.smtp.mailtrap.io',  
 "FROM\_EMAIL" => 'webmaster@example.com',  
 "FROM\_NAME" => 'Webmaster'  
];  
  
//Looking to send emails in production? Check out our Email API/SMTP product!  
//$phpmailer = new PHPMailer();  
//$phpmailer->isSMTP();  
//$phpmailer->Host = 'live.smtp.mailtrap.io';  
//$phpmailer->SMTPAuth = true;  
//$phpmailer->Port = 587;  
//$phpmailer->Username = 'api';  
//$phpmailer->Password = '94cbd2ae215d0365f2248e42f52ede4e';  
  
const *SMTP* = [  
 "HOST" => 'smtp.gmail.com', // e.g., smtp.gmail.com  
 "USERNAME" => 'adambickei12@gmail.com',  
 "PASSWORD" => 'aeyh udlx kknh bgax',  
 "FROM\_EMAIL" => 'adambickei12@gmail.com',  
 "FROM\_NAME" => 'Lapmesterek',  
];  
  
const *SITE* = 'https://lm.stud.vts.su.ac.rs/ETTEREM/register/'; // enter your path on localhost  
  
$dsn = "mysql:host=" . *PARAMS*['HOST'] . ";dbname=" . *PARAMS*['DBNAME'] . ";charset=" . *PARAMS*['CHARSET'];  
  
$pdoOptions = [  
 PDO::*ATTR\_ERRMODE* => PDO::*ERRMODE\_EXCEPTION*,  
 PDO::*ATTR\_DEFAULT\_FETCH\_MODE* => PDO::*FETCH\_ASSOC*,  
 PDO::*ATTR\_EMULATE\_PREPARES* => false  
];  
  
$actions = ['login', 'register', 'forget'];  
  
$messages = [  
 0 => 'No direct access!',  
 1 => 'Unknown user!',  
 2 => 'User with this name already exists, choose another one!',  
 3 => 'Check your email to active your account!',  
 4 => 'Fill all the fields!',  
 5 => 'You are logged out!!',  
 6 => 'Your account is activated, you can login now!',  
 7 => 'Passwords are not equal!',  
 8 => 'Format of e-mail address is not valid!',  
 9 => 'Password is too short! It must be minimum 8 characters long!',  
 10 => 'Password is not enough strong! (min 8 characters, at least 1 lowercase character, 1 uppercase character, 1 number, and 1 special character',  
 11 => 'Something went wrong with mail server. We will try to send email later!',  
 12 => 'Your account is already activated!',  
 13 => 'If you have account on our site, email with instructions for reset password is sent to you.',  
 14 => 'Something went wrong with server.',  
 15 => 'Token or other data are invalid!',  
 16 => 'Your new password is set and you can <a href="index.php" class="text-white">login</a>'  
];  
  
$emailMessages = [  
 'register' => [  
 'subject' => 'Register on web site',  
 'altBody' => 'This is the body in plain text for non-HTML mail clients'  
 ],  
 'forget' => [  
 'subject' => 'Forgotten password - create new password',  
 'altBody' => 'This is the body in plain text for non-HTML mail clients'  
 ]  
];

**Felhasználó aktiválása:**

<?php  
require\_once "config.php";  
require\_once "functions\_def.php";  
  
if (isset($\_GET['token'])) {  
 $token = trim($\_GET['token']);  
}  
  
if (!empty($token) and strlen($token) === 40) {  
  
 $sql = "UPDATE users SET active = 1, registration\_token = '', registration\_expires = NULL  
 WHERE binary registration\_token = :token AND registration\_expires>now()";  
  
 $stmt = $pdo->prepare($sql);  
  
 $stmt->bindParam(':token', $token, PDO::PARAM\_STR);  
 $stmt->execute();  
  
  
 if ($stmt->rowCount() > 0) {  
 redirection('index.php?r=6');  
 } else {  
 redirection('index.php?r=12');  
 }  
} else {  
 redirection('index.php?r=0');  
}

**Főbb JavaScript funkciók:**

const *userIcon* = *document*.querySelector('.user-info');  
const *popup* = *document*.getElementById('user-popup');  
  
*userIcon*.addEventListener('click', (e) => {  
 e.stopPropagation();  
 *popup*.classList.toggle('hidden');  
});  
  
// Bezárás ha máshová kattint a felhasználó  
*document*.addEventListener('click', (e) => {  
 if (!*popup*.contains(e.target) && !*userIcon*.contains(e.target)) {  
 *popup*.classList.add('hidden');  
 }  
});  
  
// Dynamically show/hide login/logout button and user name  
const *welcomeText* = *document*.getElementById('welcome-text');  
const *loginBtn* = *document*.getElementById('login-btn');  
const *logoutBtn* = *document*.getElementById('logout-btn');  
  
// These are set server-side in user\_view.php, so no need to set them here  
// Just ensure the correct button is visible (handled by PHP output)  
  
//Contact Button  
const *contactBtn* = *document*.getElementById('contact-btn'); // ID a footer gombra  
const *contactPopup* = *document*.getElementById('contact-popup');  
  
*contactBtn*.addEventListener('click', () => {  
 *contactPopup*.classList.toggle('hidden');  
});  
  
// Automatikus eltüntetés 5 mp után (opcionális)  
setInterval(() => {  
 if (!*contactPopup*.classList.contains('hidden')) {  
 *contactPopup*.classList.add('hidden');  
 }  
}, 5000);  
  
// Reservation Button (footer) - only if present  
const *resBtn* = *document*.getElementById('res-btn');  
const *resPopup* = *document*.getElementById('res-popup');  
if (*resBtn* && *resPopup*) {  
 *resBtn*.addEventListener('click', () => {  
 *resPopup*.classList.toggle('hidden');  
 });  
  
 setInterval(() => {  
 if (!*resPopup*.classList.contains('hidden')) {  
 *resPopup*.classList.add('hidden');  
 }  
 }, 5000);  
}  
  
const *carousel* = *document*.querySelector('#introCarousel');  
const *indicators* = *document*.querySelectorAll('.custom-indicators .indicator');  
if (*carousel* && *indicators*.length > 0) {  
 *carousel*.addEventListener('slide.bs.carousel', function (event) {  
 *indicators*.forEach(indicator => indicator.classList.remove('active'));  
 *indicators*[event.to].classList.add('active');  
 });  
}  
  
*document*.querySelectorAll('.custom-indicators .indicator').forEach(btn => {  
 btn.addEventListener('click', function(e) {  
 e.preventDefault();  
 });  
});  
  
// --- Header/Footer scroll direction logic with intro carousel detection ---  
  
const *header* = *document*.querySelector('.header');  
const *footer* = *document*.querySelector('.footer');  
const *introCarousel* = *document*.getElementById('introCarousel');  
  
let *lastScrollY* = *window*.scrollY;  
let *ticking* = false;  
  
function isIntroCarouselInView() {  
 if (!*introCarousel*) return false;  
 const rect = *introCarousel*.getBoundingClientRect();  
 // Consider in view if any part is visible in viewport  
 return rect.bottom > 0 && rect.top < *window*.innerHeight;  
}  
  
function updateHeaderFooterVisibility() {  
 if (isIntroCarouselInView()) {  
 *header*.classList.add('visible');  
 *footer*.classList.add('visible');  
 return;  
 }  
 if (*window*.scrollY < 10) {  
 // At the very top, show both  
 *header*.classList.add('visible');  
 *footer*.classList.add('visible');  
 return;  
 }  
 if (*window*.scrollY > *lastScrollY*) {  
 // Scrolling down  
 *header*.classList.remove('visible');  
 *footer*.classList.remove('visible');  
 } else if (*window*.scrollY < *lastScrollY*) {  
 // Scrolling up  
 *header*.classList.add('visible');  
 *footer*.classList.add('visible');  
 }  
 *lastScrollY* = *window*.scrollY;  
}  
  
*window*.addEventListener('scroll', () => {  
 if (!*ticking*) {  
 *window*.requestAnimationFrame(() => {  
 updateHeaderFooterVisibility();  
 *ticking* = false;  
 });  
 *ticking* = true;  
 }  
});  
  
// On load, set initial state  
*document*.addEventListener('DOMContentLoaded', () => {  
 updateHeaderFooterVisibility();  
});

**Email küldés funkció**

function sendEmail(PDO $pdo, string $email, array $emailData, string $body, int $id\_user): void  
{  
 $phpmailer = new PHPMailer(true);  
  
 try {  
 $phpmailer->isSMTP();  
  
 // Use Gmail SMTP for real email sending  
 $phpmailer->Host = 'smtp.gmail.com';  
 $phpmailer->SMTPAuth = true;  
 $phpmailer->Port = 587;  
 $phpmailer->Username = 'adambickei12@gmail.com'; // <-- your Gmail address  
 $phpmailer->Password = 'aeyh udlx kknh bgax'; // <-- your Gmail App Password (not your Gmail login password)  
 $phpmailer->setFrom('adambickei12@gmail.com', 'Lapmesterek');  
 $phpmailer->SMTPSecure = PHPMailer::*ENCRYPTION\_STARTTLS*;  
  
 $phpmailer->addAddress($email);  
 $phpmailer->isHTML(true);  
 $phpmailer->Subject = $emailData['subject'];  
 $phpmailer->Body = $body;  
 $phpmailer->AltBody = $emailData['altBody'];  
  
 $phpmailer->send();  
 } catch (Exception $e) {  
 $message = "Message could not be sent ($email). Mailer Error: {$phpmailer->ErrorInfo} ";  
 addEmailFailure($pdo, $id\_user, $message);  
 }  
  
}

**Token generálás**

function createCode(int $length): string  
{  
 $down = 97;  
 $up = 122;  
 $i = 0;  
 $code = "";  
  
 $div = mt\_rand(3, 9); // 3  
  
 while ($i < $length) {  
 if ($i % $div == 0)  
 $character = strtoupper(chr(mt\_rand($down, $up)));  
 else  
 $character = chr(mt\_rand($down, $up)); // mt\_rand(97,122) chr(98)  
 $code .= $character; // $code = $code.$character; //  
 $i++;  
 }  
 return $code;  
}

**Felhasználó kezelés role alapján**

<?php  
if (isset($\_SESSION['id\_user'])) {  
 $firstname = isset($\_SESSION['firstname']) ? $\_SESSION['firstname'] : '';  
 $lastname = isset($\_SESSION['lastname']) ? $\_SESSION['lastname'] : '';  
 $role = isset($\_SESSION['role']) ? $\_SESSION['role'] : '';  
 if (!$firstname || !$lastname || !$role) {  
 require\_once '../register/config.php';  
 // Check for required config variables  
 if (!isset($dsn) || !isset($pdoOptions) || !defined('PARAMS')) {  
 die('Database configuration variables are missing in config.php');  
 }  
 try {  
 $pdo = new PDO($dsn, *PARAMS*['USER'], *PARAMS*['PASS'], $pdoOptions);  
 } catch (PDOException $e) {  
 die('Database connection failed: ' . $e->getMessage());  
 }  
 $stmt = $pdo->prepare("SELECT firstname, lastname, role FROM users WHERE id\_user = ?");  
 $stmt->execute([$\_SESSION['id\_user']]);  
 $user = $stmt->fetch();  
 if ($user) {  
 $firstname = $user['firstname'];  
 $lastname = $user['lastname'];  
 $role = $user['role'];  
 $\_SESSION['firstname'] = $firstname;  
 $\_SESSION['lastname'] = $lastname;  
 $\_SESSION['role'] = $role;  
 }  
 }  
 echo '<p id="welcome-text">Hello, ' . htmlspecialchars($firstname . ' ' . $lastname) . '!</p>';  
 echo '<button id="logout-btn" onclick="location.href=\'../register/logout.php\'">Log out</button>';  
 if ($role === 'staff') {  
 echo '<button id="admin-btn" onclick="location.href=\'../views/staff\_dashboard.php\'" style="margin-top:2px;">Staff</button>';  
 }  
 if ($role === 'admin') {  
 echo '<button id="admin-btn" onclick="location.href=\'../views/admin\_dashbord.php\'" style="margin-top:2px;">Admin</button>';  
 }  
} else {  
 echo '<p id="welcome-text">Hello, Guest!</p>';  
 echo '<button id="login-btn" onclick="location.href=\'../register/index.php\'">Log in</button>';  
}  
?>

**Asztalfoglalás**

<?php  
  
session\_start();  
  
ini\_set('display\_errors', 1);  
ini\_set('display\_startup\_errors', 1);  
error\_reporting(*E\_ALL*);  
  
require\_once '../register/config.php';  
$pdo = new PDO($dsn, *PARAMS*['USER'], *PARAMS*['PASS'], $pdoOptions);  
  
// --- AJAX endpoint for reservation state ---  
if (isset($\_GET['ajax']) && $\_GET['ajax'] === 'table\_states') {  
 $date = $\_GET['date'] ?? '';  
 $startTime = $\_GET['startTime'] ?? '';  
 $endTime = $\_GET['endTime'] ?? '';  
 $result = [];  
  
 if ($date && $startTime && $endTime) {  
 // Get all tables  
 $stmt = $pdo->query("SELECT id, name FROM tables");  
 $tables = [];  
 while ($row = $stmt->fetch(PDO::*FETCH\_ASSOC*)) {  
 $tables[$row['name']] = [  
 'id' => $row['id'],  
 'status' => 'available'  
 ];  
 }  
  
 // Get reservations for the selected time  
 $resStmt = $pdo->prepare("  
 SELECT t.name, r.status  
 FROM reservations r  
 JOIN tables t ON t.id = r.table\_id  
 WHERE r.res\_date = ?  
 AND (  
 (r.start\_time <= ? AND ADDTIME(r.start\_time, SEC\_TO\_TIME(r.duration\_hours\*3600)) > ?)  
 OR  
 (r.start\_time < ? AND ADDTIME(r.start\_time, SEC\_TO\_TIME(r.duration\_hours\*3600)) >= ?)  
 OR  
 (r.start\_time >= ? AND r.start\_time < ?)  
 )  
 AND r.status IN ('active', 'cancelled')  
 ");  
 $resStmt->execute([  
 $date,  
 $startTime, $startTime,  
 $endTime, $endTime,  
 $startTime, $endTime  
 ]);  
 while ($row = $resStmt->fetch(PDO::*FETCH\_ASSOC*)) {  
 // 'active' means reserved, 'cancelled' means cancelled  
 $tables[$row['name']]['status'] = $row['status'] === 'active' ? 'reserved' : 'cancelled';  
 }  
 foreach ($tables as $name => $data) {  
 $result[$name] = $data['status'];  
 }  
 }  
 header('Content-Type: application/json');  
 echo json\_encode($result);  
 exit;  
}  
  
// --- AJAX endpoint for latest reservation end time ---  
if (isset($\_GET['ajax']) && $\_GET['ajax'] === 'latest\_end\_time') {  
 $tableName = $\_GET['table'] ?? '';  
 $date = $\_GET['date'] ?? '';  
 $result = ['latest\_end\_time' => null];  
 if ($tableName && $date) {  
 $stmt = $pdo->prepare("  
 SELECT ADDTIME(start\_time, SEC\_TO\_TIME(duration\_hours\*3600)) AS end\_time  
 FROM reservations r  
 JOIN tables t ON t.id = r.table\_id  
 WHERE t.name = ? AND r.res\_date = ? AND r.status = 'active'  
 ORDER BY end\_time DESC LIMIT 1  
 ");  
 $stmt->execute([$tableName, $date]);  
 $row = $stmt->fetch(PDO::*FETCH\_ASSOC*);  
 if ($row && $row['end\_time']) {  
 $result['latest\_end\_time'] = $row['end\_time'];  
 }  
 }  
 header('Content-Type: application/json');  
 echo json\_encode($result);  
 exit;  
}  
// Reservation logic  
$resMsg = '';  
if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {  
 // Check if user is logged in  
 if (!isset($\_SESSION['id\_user']) || !is\_int($\_SESSION['id\_user'])) {  
 $resMsg = '<div class="error-msg">You must be logged in to make a reservation.</div>';  
 } else {  
 $tableName = $\_POST['selectedTable'] ?? '';  
 $date = $\_POST['date'] ?? '';  
 $startTime = $\_POST['startTime'] ?? '';  
 $endTime = $\_POST['endTime'] ?? '';  
 $guests = (int)($\_POST['guests'] ?? 0);  
  
 // Validate  
 if (!$tableName || !$date || !$startTime || !$endTime || $guests < 1) {  
 $resMsg = '<div class="error-msg">Please fill all fields correctly.</div>';  
 } else {  
 // Get table id  
 $stmt = $pdo->prepare("SELECT id, seats FROM tables WHERE name = ?");  
 $stmt->execute([$tableName]);  
 $table = $stmt->fetch();  
 if (!$table) {  
 $resMsg = '<div class="error-msg">Invalid table selected.</div>';  
 } elseif ($guests > $table['seats']) {  
 $resMsg = '<div class="error-msg">Too many guests for this table.</div>';  
 } else {  
 // Calculate duration in hours  
 $start = strtotime($startTime);  
 $end = strtotime($endTime);  
 $duration = ($end - $start) / 3600;  
  
 // --- Enforce 15 min gap after latest reservation ---  
 $latestEndStmt = $pdo->prepare("  
 SELECT ADDTIME(start\_time, SEC\_TO\_TIME(duration\_hours\*3600)) AS end\_time  
 FROM reservations  
 WHERE table\_id = ? AND res\_date = ? AND status = 'active'  
 ORDER BY end\_time DESC LIMIT 1  
 ");  
 $latestEndStmt->execute([$table['id'], $date]);  
 $latestEnd = $latestEndStmt->fetchColumn();  
 if ($latestEnd) {  
 $minStart = date('H:i', strtotime($latestEnd) + 15\*60);  
 if ($startTime < $minStart) {  
 $resMsg = '<div class="error-msg">You can only book this table from ' . htmlspecialchars($minStart) . ' or later, due to a previous reservation.</div>';  
 }  
 }  
 if (empty($resMsg)) {  
 if ($duration < 1 || $duration > 6) {  
 $resMsg = '<div class="error-msg">Reservation must be 1-6 hours.</div>';  
 } else {  
 // Check for overlapping reservations  
 $check = $pdo->prepare("  
 SELECT 1 FROM reservations  
 WHERE table\_id = ? AND res\_date = ?  
 AND (  
 (start\_time <= ? AND ADDTIME(start\_time, SEC\_TO\_TIME(duration\_hours\*3600)) > ?)  
 OR  
 (start\_time < ? AND ADDTIME(start\_time, SEC\_TO\_TIME(duration\_hours\*3600)) >= ?)  
 OR  
 (start\_time >= ? AND start\_time < ?)  
 )  
 AND status = 'active'  
 ");  
 $check->execute([  
 $table['id'], $date,  
 $startTime, $startTime,  
 $endTime, $endTime,  
 $startTime, $endTime  
 ]);  
 if ($check->fetch()) {  
 $resMsg = '<div class="error-msg">This table is already reserved for the selected time.</div>';  
 } else {  
 // Insert reservation  
 $reservation\_code = substr(md5(uniqid(mt\_rand(), true)), 0, 12);  
 $user\_id = $\_SESSION['id\_user'];  
  
 $ins = $pdo->prepare("  
 INSERT INTO reservations (user\_id, table\_id, reservation\_code, res\_date, start\_time, duration\_hours, guest\_number, status)  
 VALUES (?, ?, ?, ?, ?, ?, ?, 'pending')  
 ");  
 $ins->execute([  
 $user\_id,  
 $table['id'],  
 $reservation\_code,  
 $date,  
 $startTime,  
 $duration,  
 $guests  
 ]);  
 $resMsg = '<div class="success-msg">Reservation successful! Your code: <b>' . htmlspecialchars($reservation\_code) . '</b></div>';  
 }  
 }  
 }  
 }  
 }  
 }  
}  
  
// Fetch tables and their latest reservation status  
$stmt = $pdo->query("  
 SELECT t.id, t.name, t.location\_description,  
 COALESCE(r.status, 'available') AS status  
FROM tables t  
LEFT JOIN (  
 SELECT table\_id, MIN(status) AS status  
 FROM reservations  
 WHERE res\_date = CURDATE()  
 AND start\_time <= CURTIME()  
 AND ADDTIME(start\_time, SEC\_TO\_TIME(duration\_hours\*3600)) >= CURTIME()  
 GROUP BY table\_id  
) r ON t.id = r.table\_id  
");  
$tables = [];  
while ($row = $stmt->fetch(PDO::*FETCH\_ASSOC*)) {  
 $tables[$row['name']] = [  
 'description' => $row['location\_description'],  
 'status' => $row['status']  
 ];  
}  
?>  
<!DOCTYPE html>  
<html lang="hu">  
<head>  
 <meta charset="UTF-8" />  
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
 <title>Reservation</title>  
 <link rel="stylesheet" href="../styles/table.css">  
</head>  
<body>  
<div class="reservation-layout">  
 <div class="map-container">  
 <img src="../img/asztalok.png" alt="Étterem alaprajz">  
 </div>  
 <div id="tableTooltip" class="table-tooltip"></div>  
 <form class="form-container" method="post">  
 <h2>Reservation</h2>  
 <?php if (!empty($resMsg)) echo $resMsg; ?>  
  
 <div class="legend">  
 <span class="available">Available</span>  
 <span class="reserved">Reserved</span>  
 <span class="cancelled">Cancelled</span>  
 </div>  
  
 <div class="selected-table">  
 <label for="selectedTable">Selected table:</label>  
 <input type="text" id="selectedTable" name="selectedTable" readonly placeholder="Nincs kiválasztva">  
 </div>  
  
 <div class="form-grid">  
 <div class="row-4">  
 <?php foreach (['V1','V2','V3','V4'] as $t):  
 $status = isset($tables[$t]['status']) ? $tables[$t]['status'] : 'available';  
 $desc = isset($tables[$t]['description']) ? htmlspecialchars($tables[$t]['description']) : '';  
 $statusClass = ($status === 'active' || $status === 'reserved') ? 'reserved' : ($status === 'cancelled' ? 'cancelled' : 'available');  
 $disabled = ($statusClass === 'reserved') ? 'disabled' : '';  
 ?>  
 <button type="button" class="table-button <?= $statusClass ?>" data-name="<?= $t ?>" data-desc="<?= $desc ?>" <?= $disabled ?>><?= $t ?></button>  
 <?php endforeach; ?>  
 </div>  
 <div class="row-grid">  
 <?php foreach (['T1','T2','T3','T4','T5','T6','T7','T8','T9','T10','T11','T12'] as $t):  
 $status = isset($tables[$t]['status']) ? $tables[$t]['status'] : 'available';  
 $desc = isset($tables[$t]['description']) ? htmlspecialchars($tables[$t]['description']) : '';  
 $statusClass = ($status === 'active' || $status === 'reserved') ? 'reserved' : ($status === 'cancelled' ? 'cancelled' : 'available');  
 $disabled = ($statusClass === 'reserved') ? 'disabled' : '';  
 ?>  
 <button type="button" class="table-button <?= $statusClass ?>" data-name="<?= $t ?>" data-desc="<?= $desc ?>" <?= $disabled ?>><?= $t ?></button>  
 <?php endforeach; ?>  
 </div>  
 </div>  
  
 <div class="time-section">  
 <label for="date">Date:</label>  
 <input type="date" id="date" name="date">  
  
 <label for="startTime">Start time:</label>  
 <input type="time" id="startTime" name="startTime" step="900">  
  
 <label for="endTime">End time:</label>  
 <input type="time" id="endTime" name="endTime" step="900">  
  
 <label for="guests">Guest number:</label>  
 <input type="number" id="guests" name="guests" min="1" max="8" placeholder="Pl: 4">  
 </div>  
  
 <button type="submit" class="submit-btn">RESERVATION</button>  
 <button type="reset" class="reset-btn">RESET</button>  
 <a href="../views/user\_view.php" class="back-home-btn">  
 <span class="back-arrow">&#8592;</span> Back to Home  
 </a>  
 </form>  
</div>  
  
<script>  
 // Tooltip logic  
 const *tooltip* = *document*.getElementById('tableTooltip');  
 *document*.querySelectorAll('.table-button').forEach(button => {  
 button.addEventListener('mouseenter', function(e) {  
 const desc = this.getAttribute('data-desc');  
 if (desc) {  
 *tooltip*.textContent = desc;  
 *tooltip*.style.display = 'block';  
 const rect = this.getBoundingClientRect();  
 *tooltip*.style.left = (rect.left + *window*.scrollX + rect.width/2 - *tooltip*.offsetWidth/2) + 'px';  
 *tooltip*.style.top = (rect.top + *window*.scrollY - *tooltip*.offsetHeight - 8) + 'px';  
 }  
 });  
 button.addEventListener('mousemove', function(e) {  
 *tooltip*.style.left = (e.pageX + 10) + 'px';  
 *tooltip*.style.top = (e.pageY - 40) + 'px';  
 });  
 button.addEventListener('mouseleave', function() {  
 *tooltip*.style.display = 'none';  
 });  
 button.addEventListener('click', function (e) {  
 e.preventDefault();  
 if (this.disabled) return;  
 const name = this.getAttribute('data-name');  
 *document*.getElementById('selectedTable').value = name;  
 *document*.querySelectorAll('.table-button').forEach(btn => btn.classList.remove('selected'));  
 this.classList.add('selected');  
 fetchLatestEndTime();  
 });  
 });  
  
 // --- Async fetch reservation state ---  
 const *dateInput* = *document*.getElementById('date');  
 const *startInput* = *document*.getElementById('startTime');  
 const *endInput* = *document*.getElementById('endTime');  
 const *submitBtn* = *document*.querySelector('.submit-btn');  
 let *minStartTime* = null;  
  
 function fetchTableStates() {  
 const date = *dateInput*.value;  
 const startTime = *startInput*.value;  
 const endTime = *endInput*.value;  
 if (!date || !startTime || !endTime) return;  
 fetch(`asztalfoglalas.php?ajax=table\_states&date=${encodeURIComponent(date)}&startTime=${encodeURIComponent(startTime)}&endTime=${encodeURIComponent(endTime)}`)  
 .then(res => res.json())  
 .then(states => {  
 *document*.querySelectorAll('.table-button').forEach(btn => {  
 const name = btn.getAttribute('data-name');  
 btn.classList.remove('reserved', 'cancelled', 'available');  
 btn.disabled = false;  
 if (states[name] === 'reserved') {  
 btn.classList.add('reserved');  
 btn.disabled = true;  
 if (btn.classList.contains('selected')) {  
 btn.classList.remove('selected');  
 *document*.getElementById('selectedTable').value = '';  
 }  
 } else if (states[name] === 'cancelled') {  
 btn.classList.add('cancelled');  
 } else {  
 btn.classList.add('available');  
 }  
 });  
 });  
 }  
 *dateInput*.addEventListener('change', () => {  
 fetchTableStates();  
 fetchLatestEndTime();  
 });  
 *startInput*.addEventListener('change', fetchTableStates);  
 *endInput*.addEventListener('change', fetchTableStates);  
  
 // --- Enforce min start time logic ---  
 function pad(num) { return num.toString().padStart(2, '0'); }  
 function addMinutesToTime(timeStr, mins) {  
 const [h, m, s] = timeStr.split(':');  
 let date = new *Date*();  
 date.setHours(parseInt(h), parseInt(m), s ? parseInt(s) : 0, 0);  
 date.setMinutes(date.getMinutes() + mins);  
 return pad(date.getHours()) + ':' + pad(date.getMinutes());  
 }  
 function fetchLatestEndTime() {  
 const table = *document*.getElementById('selectedTable').value;  
 const date = *dateInput*.value;  
 if (!table || !date) {  
 *startInput*.min = '';  
 *minStartTime* = null;  
 validateTimeInputs();  
 return;  
 }  
 fetch(`asztalfoglalas.php?ajax=latest\_end\_time&table=${encodeURIComponent(table)}&date=${encodeURIComponent(date)}`)  
 .then(res => res.json())  
 .then(data => {  
 if (data.latest\_end\_time) {  
 // Add 15 minutes  
 *minStartTime* = addMinutesToTime(data.latest\_end\_time, 15);  
 *startInput*.min = *minStartTime*;  
 } else {  
 *minStartTime* = null;  
 *startInput*.min = '';  
 }  
 validateTimeInputs();  
 });  
 }  
 *startInput*.addEventListener('change', validateTimeInputs);  
 *endInput*.addEventListener('change', validateTimeInputs);  
 *dateInput*.addEventListener('change', validateTimeInputs);  
 *document*.getElementById('selectedTable').addEventListener('change', fetchLatestEndTime);  
  
 function validateTimeInputs() {  
 const startVal = *startInput*.value;  
 const endVal = *endInput*.value;  
 let valid = true;  
 if (!startVal || !endVal) valid = false;  
 if (*minStartTime* && startVal && startVal < *minStartTime*) valid = false;  
 if (startVal && endVal && startVal >= endVal) valid = false;  
 *submitBtn*.disabled = !valid;  
 }  
 // Disable submit by default  
 *submitBtn*.disabled = true;  
</script>  
  
</body>  
</html>

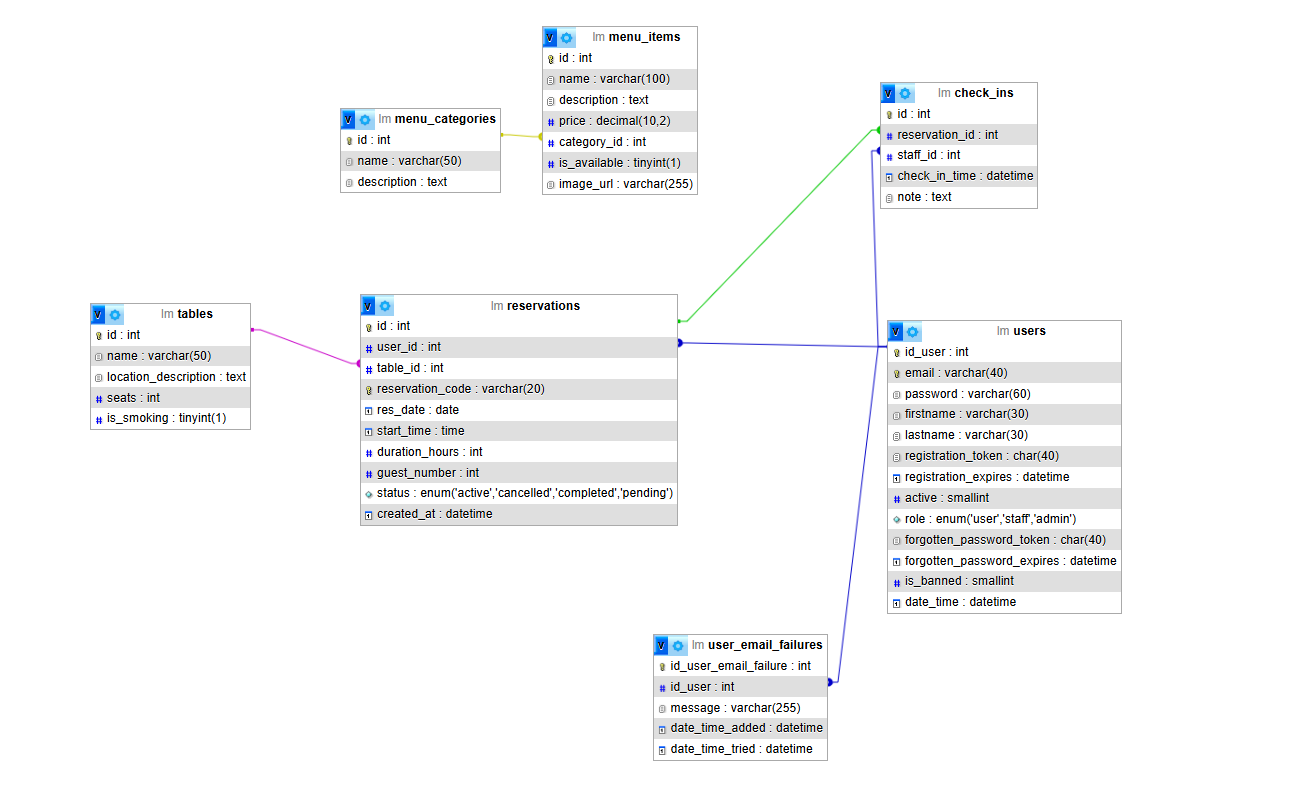
**Felkapott ételek random megjelenítése**

function getPopularDishes($limit = 8) {  
 global $pdo;  
  
 // In a real application, you might determine popularity based on orders or ratings  
 // For now, we'll just get random dishes from the menu\_items table  
 $query = "SELECT mi.id, mi.name, mi.description, mi.price, mi.image\_url, mc.name as category\_name   
 FROM menu\_items mi  
 JOIN menu\_categories mc ON mi.category\_id = mc.id  
 WHERE mi.is\_available = 1  
 ORDER BY RAND()  
 LIMIT :limit";  
  
 $stmt = $pdo->prepare($query);  
 $stmt->bindParam(':limit', $limit, PDO::*PARAM\_INT*);  
 $stmt->execute();  
  
 return $stmt->fetchAll();  
}

**Étel rendelés**

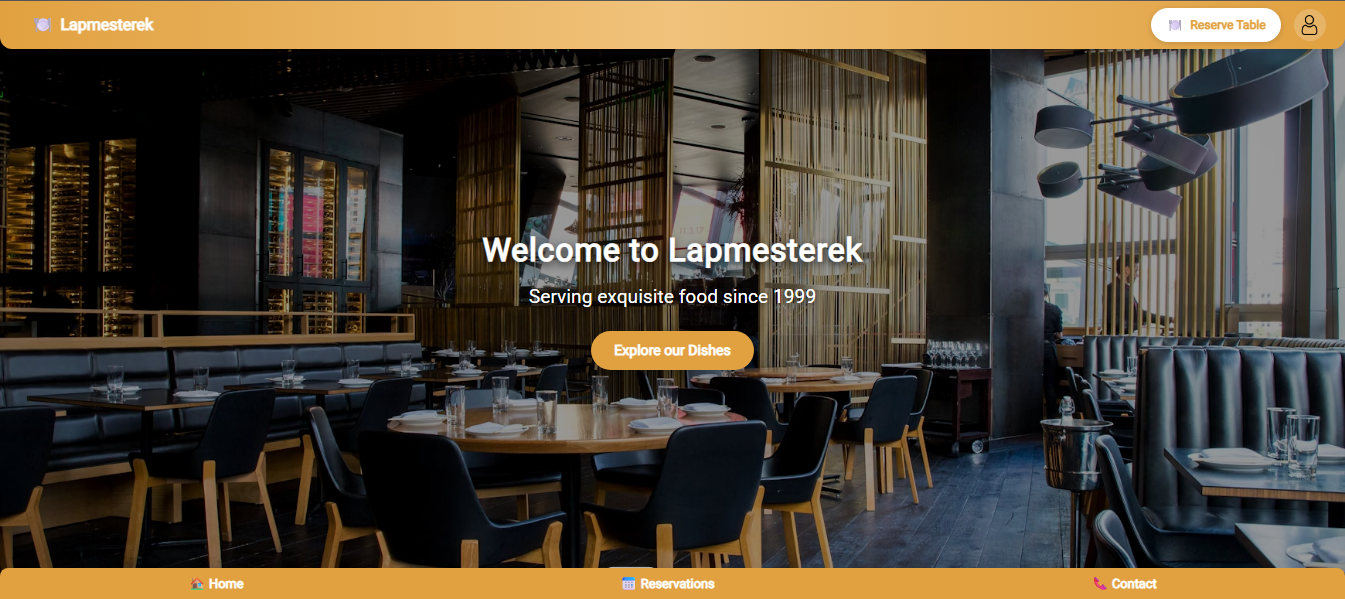
$pdo = new PDO($dsn, *PARAMS*['USER'], *PARAMS*['PASS'], $pdoOptions);  
  
// --- Ensure email is set in session for logged-in users ---  
if (isset($\_SESSION['id\_user']) && empty($\_SESSION['email'])) {  
 $stmt = $pdo->prepare("SELECT email FROM users WHERE id\_user = ?");  
 $stmt->execute([$\_SESSION['id\_user']]);  
 $user = $stmt->fetch();  
 if ($user && !empty($user['email'])) {  
 $\_SESSION['email'] = $user['email'];  
 }  
}  
  
// Fetch all menu categories  
$stmtCategories = $pdo->query("SELECT id, name FROM menu\_categories ORDER BY name");  
$categories = $stmtCategories->fetchAll();  
  
// Fetch all menu items  
$stmtItems = $pdo->query("SELECT m.id, m.name, m.description, m.price, m.image\_url, m.category\_id, c.name AS category\_name   
 FROM menu\_items m   
 LEFT JOIN menu\_categories c ON m.category\_id = c.id   
 WHERE m.is\_available = 1  
 ORDER BY c.name, m.name");  
$menuItems = $stmtItems->fetchAll();  
  
// Accept dish\_id from GET for pre-selection  
$preselected\_dish\_id = isset($\_GET['id']) ? (int)$\_GET['id'] : 0;  
  
$orderSuccess = false;  
$orderError = '';  
  
// Pre-fill user data if logged in (but phone is always empty)  
$userData = [  
 'name' => '',  
 'email' => '',  
 'phone' => ''  
];  
if (isset($\_SESSION['id\_user'])) {  
 $userData['name'] = trim(($\_SESSION['firstname'] ?? '') . ' ' . ($\_SESSION['lastname'] ?? ''));  
 $userData['email'] = $\_SESSION['email'] ?? '';  
}  
  
if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {  
 $name = trim($\_POST['name']);  
 $email = trim($\_POST['email']);  
 $phone = trim($\_POST['phone']);  
 $notes = trim($\_POST['notes']);  
 $order\_items = json\_decode($\_POST['order\_items'], true) ?? [];  
  
 // Address fields  
 $street = trim($\_POST['street'] ?? '');  
 $city = trim($\_POST['city'] ?? '');  
 $zip = trim($\_POST['zip'] ?? '');  
 $country = 'Serbia'; // Always Serbia  
  
 // Validate required fields and order items  
 if (filter\_var($email, *FILTER\_VALIDATE\_EMAIL*) && $name && $phone && $street && $city && $zip && !empty($order\_items)) {  
 // Fetch ordered items details  
 $item\_ids = array\_keys($order\_items);  
 $placeholders = str\_repeat('?,', count($item\_ids) - 1) . '?';  
 $stmt = $pdo->prepare("SELECT m.id, m.name, m.description, m.price, m.image\_url, c.name AS category\_name  
 FROM menu\_items m  
 LEFT JOIN menu\_categories c ON m.category\_id = c.id  
 WHERE m.id IN ($placeholders)");  
 $stmt->execute($item\_ids);  
 $ordered\_dishes = $stmt->fetchAll(PDO::*FETCH\_ASSOC*);  
  
 // Calculate total and create order summary  
 $total\_price = 0;  
 $order\_summary = '';  
 $item\_images\_html = '';  
  
 foreach ($ordered\_dishes as $dish) {  
 $quantity = $order\_items[$dish['id']];  
 $item\_total = ($dish['price'] \* $quantity) / 100;  
 $total\_price += $item\_total;  
  
 $order\_summary .= "<tr>  
 <td>" . htmlspecialchars($dish['name']) . "</td>  
 <td>" . $quantity . "</td>  
 <td>€" . number\_format($dish['price'] / 100, 2) . "</td>  
 <td>€" . number\_format($item\_total, 2) . "</td>  
 </tr>";  
  
 // Add item image for email  
 if (!empty($dish['image\_url'])) {  
 $item\_images\_html .= "<div style='display:inline-block; margin:10px; text-align:center;'>  
 <img src='" . htmlspecialchars($dish['image\_url']) . "' alt='" . htmlspecialchars($dish['name']) . "'   
 style='width:120px; height:80px; object-fit:cover; border-radius:8px; border:1px solid #ddd;'>  
 <br><span style='font-size:12px; color:#666;'>" . htmlspecialchars($dish['name']) . " (x" . $quantity . ")</span>  
 </div>";  
 }  
 }  
  
 // Compose email with images  
 $subject = "Order Confirmation - Your Food Order";  
 $body = "<h2>Thank you for your order!</h2>";  
  
 if ($item\_images\_html) {  
 $body .= "<h3>Your Ordered Items:</h3>  
 <div style='margin:20px 0; padding:15px; background:#f9f9f9; border-radius:8px;'>  
 " . $item\_images\_html . "  
 </div>";  
 }  
  
 $body .= "<h3>Order Summary:</h3>  
 <table style='width:100%; border-collapse:collapse; margin:20px 0;'>  
 <thead>  
 <tr style='background:#E1A140; color:white;'>  
 <th style='padding:10px; border:1px solid #ddd;'>Item</th>  
 <th style='padding:10px; border:1px solid #ddd;'>Quantity</th>  
 <th style='padding:10px; border:1px solid #ddd;'>Unit Price</th>  
 <th style='padding:10px; border:1px solid #ddd;'>Total</th>  
 </tr>  
 </thead>  
 <tbody>" . $order\_summary . "</tbody>  
 <tfoot>  
 <tr style='background:#f5f5f5; font-weight:bold;'>  
 <td colspan='3' style='padding:10px; border:1px solid #ddd; text-align:right;'>Total:</td>  
 <td style='padding:10px; border:1px solid #ddd;'>€" . number\_format($total\_price, 2) . "</td>  
 </tr>  
 </tfoot>  
 </table>  
 <p><b>Customer Details:</b></p>  
 <p><b>Name:</b> " . htmlspecialchars($name) . "</p>  
 <p><b>Email:</b> " . htmlspecialchars($email) . "</p>  
 <p><b>Phone:</b> " . htmlspecialchars($phone) . "</p>  
 <p><b>Delivery Address:</b><br>"  
 . htmlspecialchars($street) . "<br>"  
 . htmlspecialchars($zip) . " " . htmlspecialchars($city) . "<br>"  
 . "Serbia</p>";  
  
 if ($notes) {  
 $body .= "<p><b>Notes:</b> " . nl2br(htmlspecialchars($notes)) . "</p>";  
 }  
  
 $body .= "<p><b>Estimated delivery time:</b> 25–30 minutes</p>";  
  
 $emailData = [  
 'subject' => $subject,  
 'altBody' => strip\_tags($body)  
 ];  
  
 // Send email  
 sendEmail($pdo, $email, $emailData, $body, isset($\_SESSION['id\_user']) ? (int)$\_SESSION['id\_user'] : 0);  
  
 $orderSuccess = true;  
 // Save address for map display  
 $orderAddress = $street . ', ' . $zip . ' ' . $city . ', Serbia';  
 } else {  
 $orderError = "Please fill all required fields and add at least one item to your order.";  
 }  
}  
?>

# Az adatbázis struktúrája (MySQL Workbench)

****

# A projekt működésének leírása

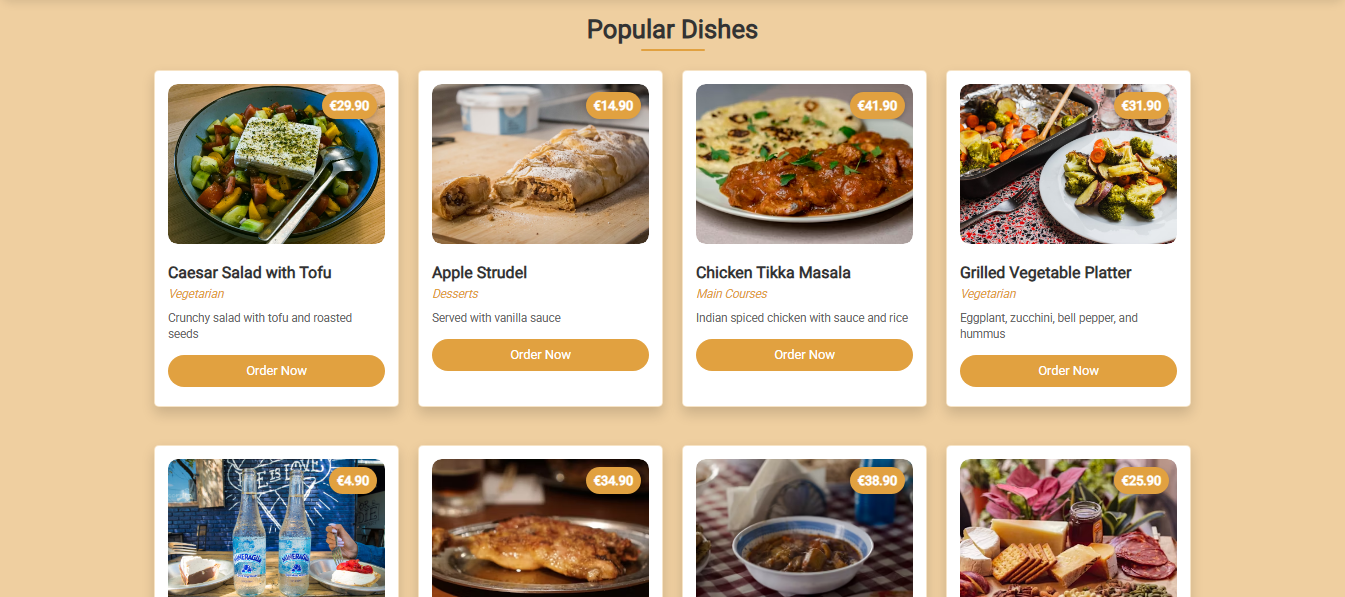
Ha belépünk az oldalra ez fogad minket vendégként. Van a fejlécben 2 gombunk jobb oldalon az első gomb az asztalfoglalásra vezet de mivel nem vagyunk bejelentkezve így a regisztrációs oldalra dob minket

****

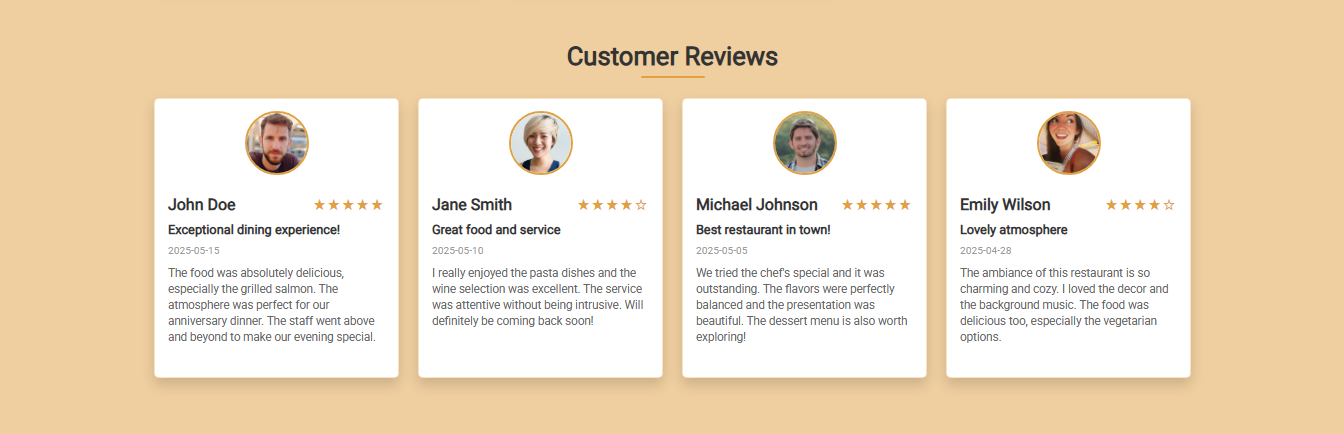
Ha a második gombot nyomjuk meg akkor be tudunk jelentkezni/regisztrálni tudunk



Ha lejjebb görgetünk akkor az ételekkel találjuk szembe magunkat amiket szintén akkor tudunk megrendelni miután bejelentkeztünk



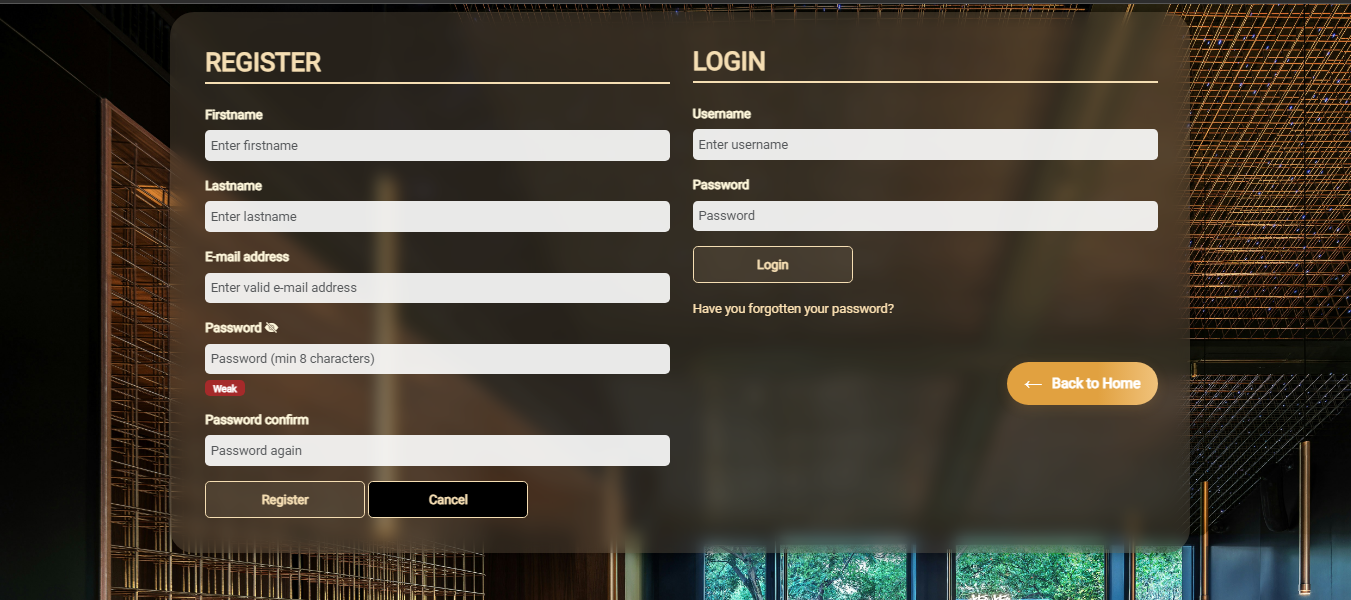
Legalul egy visszajelzés szekciót találunk ahol véleményeket írtak az emberek



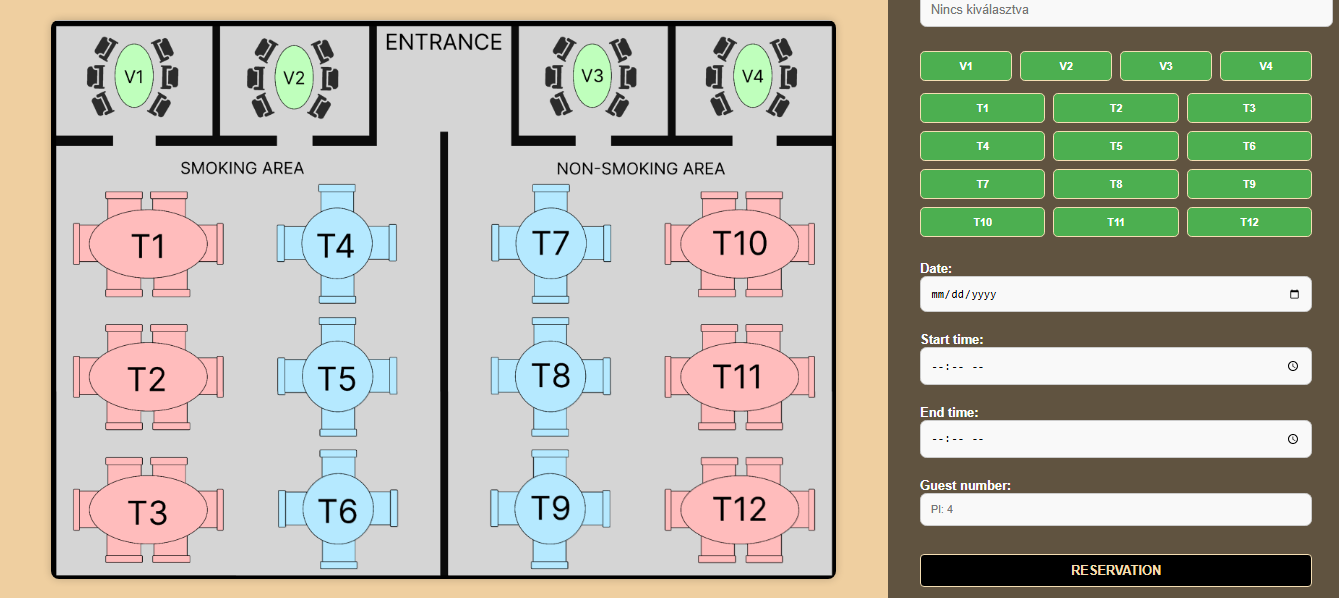
A lábléc részben 3 gombunk van az első felvezet minket az oldal tetejére a középső gomb segítségével meg tudjuk tekinteni a foglalásainkat az utolsó gomb megnyomásával pedig az elérhetőségünket jelenítjük meg



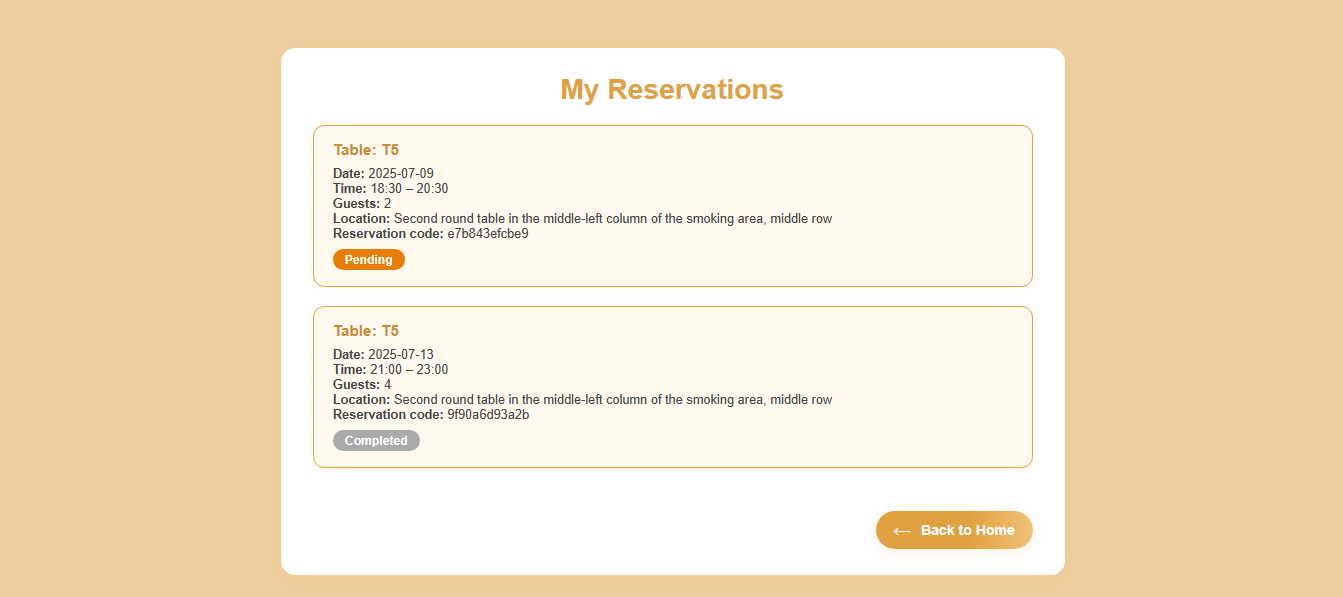
A bejelentkezés/regisztréciós oldalon ez fogad minket ahol tudunk új felhasználót létrehozni, bejelentkezni egy meglévőbe, illetve ha elfelejtettük a jelszavunkat akkor meg tudjuk azt változtatni



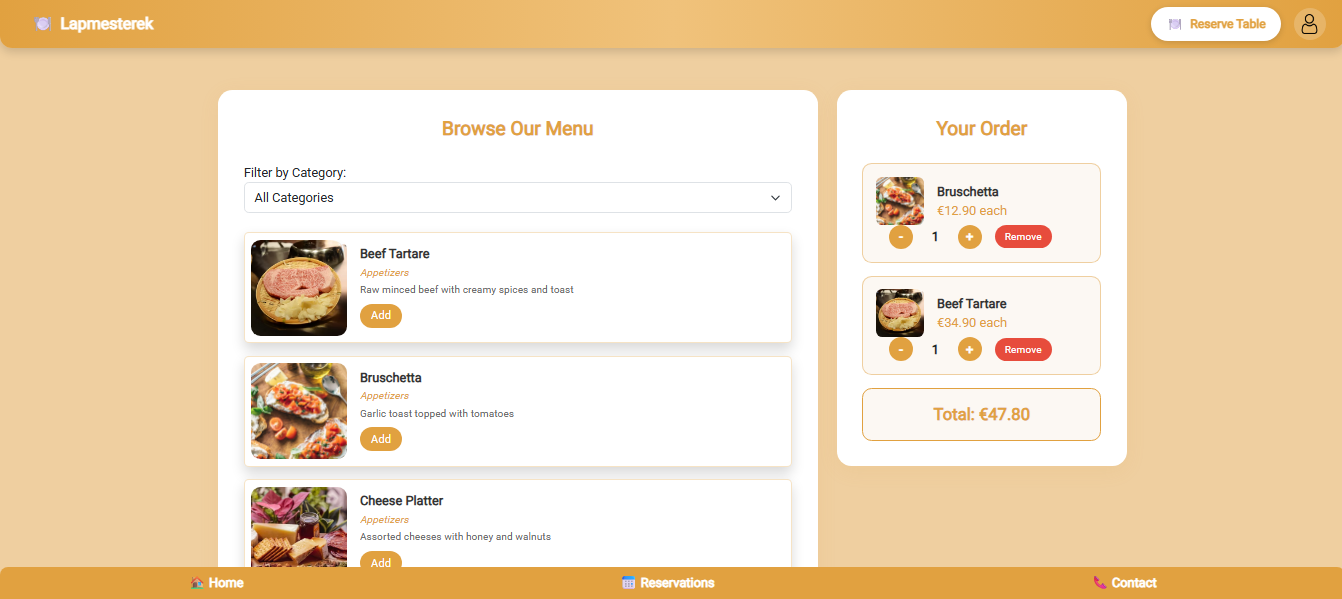
Miután bejelentkeztünk a Reserve Table gomb segítségével meg tudjuk nyitni az asztalfoglalást ahol le tudjuk foglalni a kiválasztott asztalunkat a kívánt időpontra. Meg kell adnunk a dátumot az időtartamot hogy mettől meddig szeretnénk ott tartózkodni illetve a létszámot



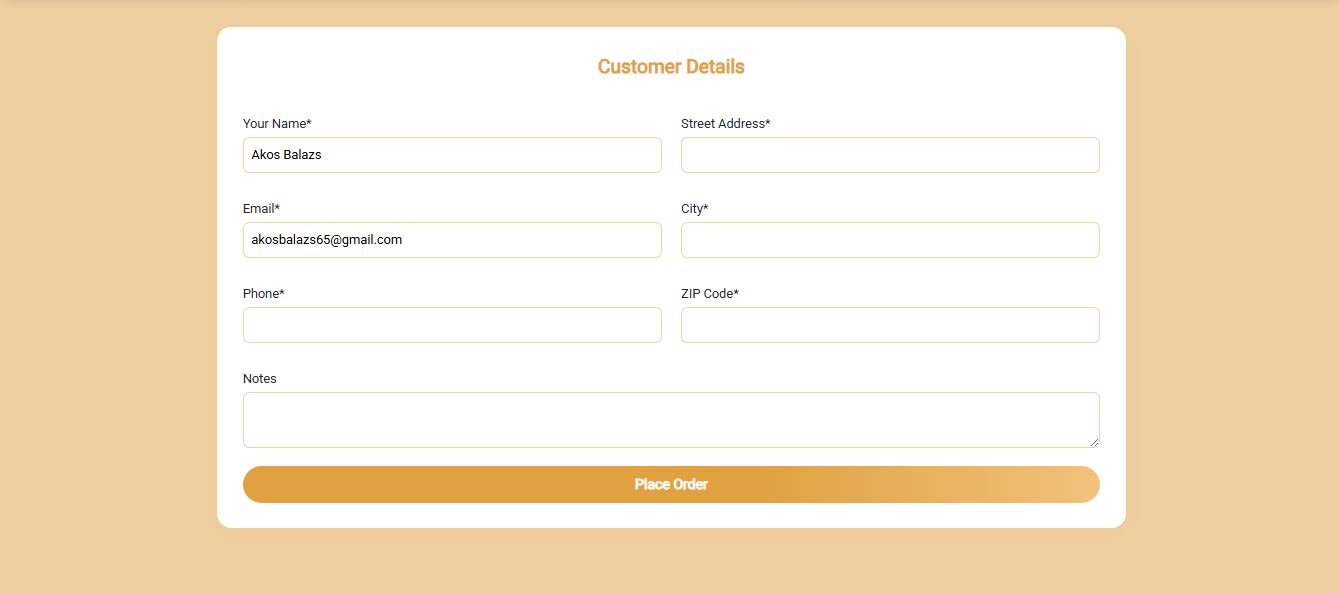
Miután lefoglaltuk az asztalunkat a footerben található Reservations gomb segítségével meg tudjuk nézni a foglalásainkat.



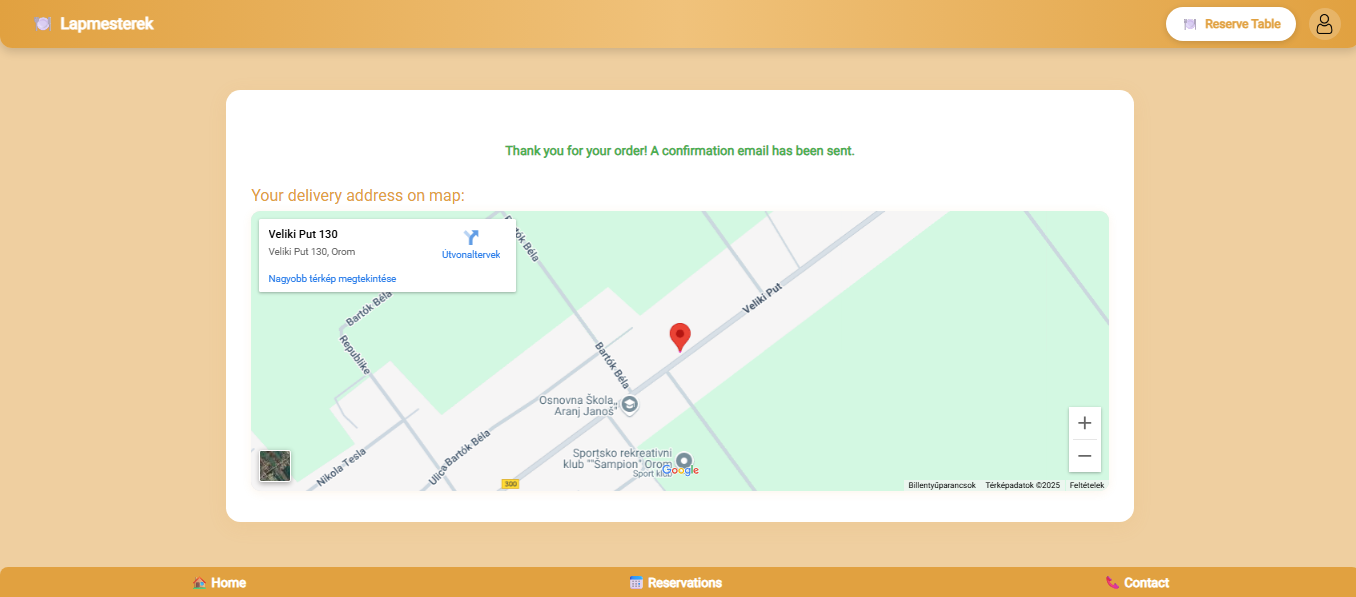
Az ételrendelés oldalon tudjuk szelektálni az ételeket kategória alapján illetve az add gomb megnyomásával hozzá tudunk adni ételeket/italokat a rendeléshez, látjuk hogy mennyit fogunk fizetni.



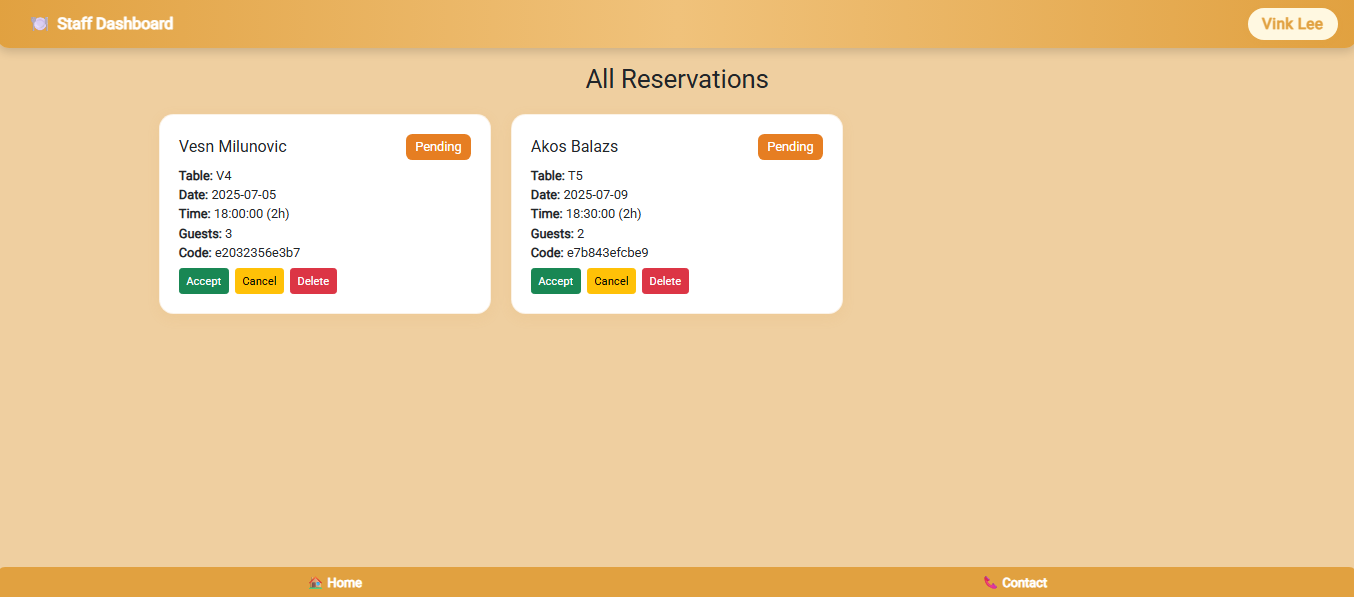
Ha lejebb megyünk akkor az adatok megadásával és a Place Order gomb megnyomásával le tudjuk adni a rendelésünket

****

Miután megadtuk az adatokat és rányomtunk a place order gombra akkor megjelenik egy térkép bejelölve az uticélt ahova rendeltük az ételünket

****

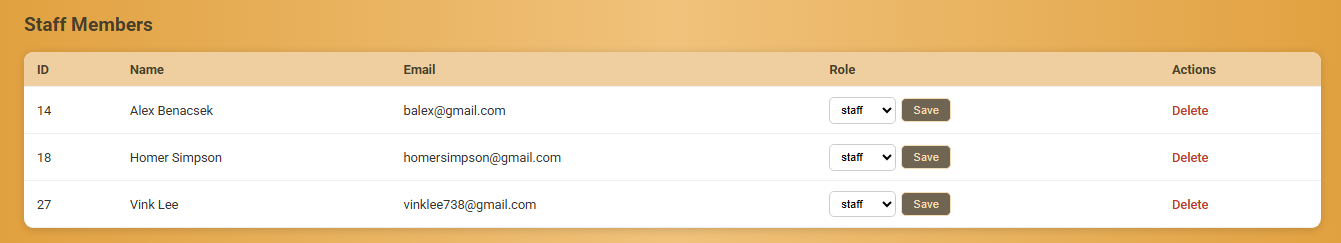
Ha munnkásként vagyunk bejelentkezve akko kezelni tudjuk a foglalásokat el tudjuk őket fogadni, elutasítani vagy akár törölni is tudjuk őket

****

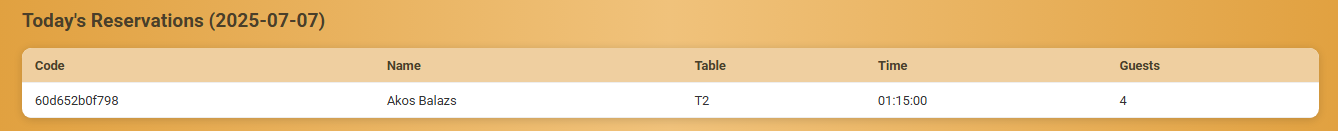
Ha adminként vagyunk bejelentkezve akkor az első szekcióban kezelni tudjuk a felhasználókat tudjuk a beosztásukat módosítani illetve tudjuk őket törölni

****

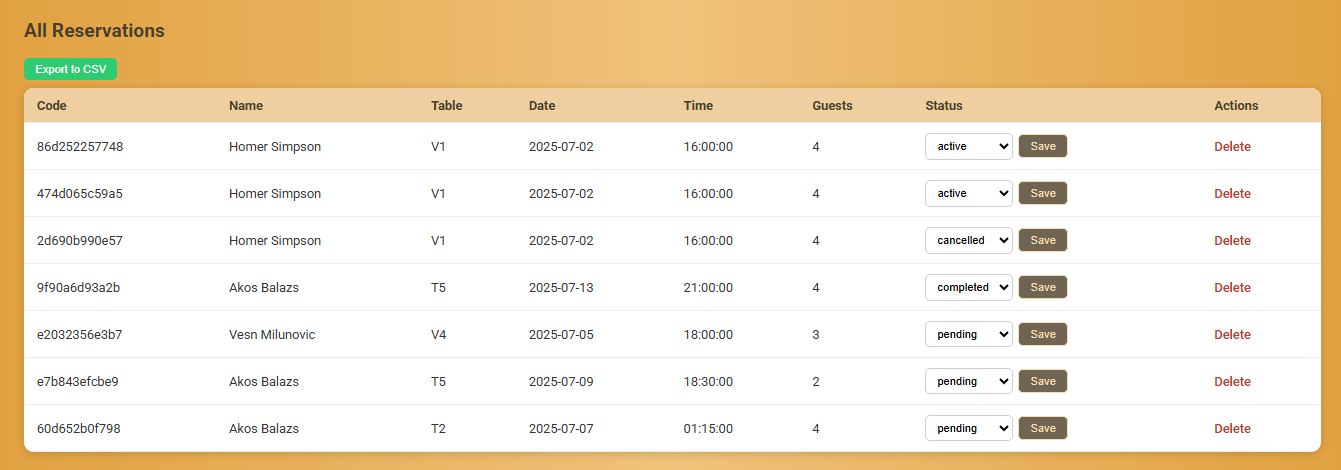
Külön tudjuk a munkásokat is kezelni lehet törölni őket illetve módosítani a beosztásukat

****

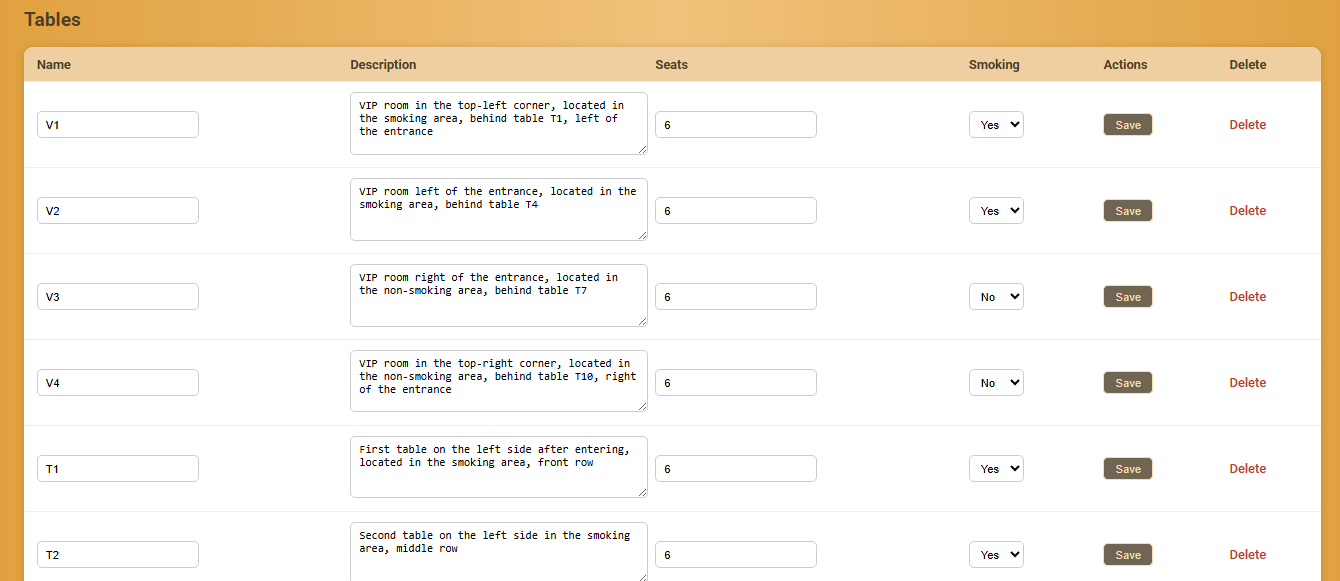
Látjuk az aktuális napon hogy milyen rendelések vannak

****

Tudjuk kezelni az összes rendelést tudjuk őket módosjtani illetve törölni



A kövekező részen az asztalokat és az információkat találjuk. Az asztalok adatait tudjuk múdosítani illetve tudunk új asztalt is hozzáadni





# Felhasznált irodalom

W1. <http://www.w3schools.com/>

W2. <https://www.php.net/>

W3. <https://getbootstrap.com/>

W4. <https://javascript.info/>

W5. <https://chat.openai.com/>