**SZABADKAI MŰSZAKI SZAKFŐISKOLA**

**PROJEKTUM**

**Haladó web programozás és Integrált web rendszerek tantárgyakból**

**HALLGATÓ(K) MENTOR**

**Bálint Tamás, Vajer Zsolt dr Zlatko Čović**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SZABADKA, 2024.**

**TARTALOM**

[A FELADATKÖR MEGHATÁROZÁSA 2](#_Toc175611167)

[PIACKUTATÁS 3](#_Toc175611168)

[Ban User NVE( notification via email) 4](#_Toc175611169)

[Banned User logs 5](#_Toc175611170)

[Az összes felhasználó 10](#_Toc175611171)

[KIDOLGOZÁS 14](#_Toc175611172)

[Haladó web programozás 14](#_Toc175611173)

[Felhasználói felület 14](#_Toc175611174)

[Felhasználó bejelentkezése 14](#_Toc175611175)

[Időpontok megtekintése 17](#_Toc175611176)

[Időpontfoglalás 19](#_Toc175611177)

[Szalon tulajdonos 20](#_Toc175611178)

[Integrált web rendszerek 22](#_Toc175611179)

[Hozzáférési eszközök észlelése 23](#_Toc175611180)

[Mobil alkalmazás 26](#_Toc175611181)

[A mobil alkalmazás login felülete: 26](#_Toc175611182)

[A mobil alkalmazás fő oldala: 29](#_Toc175611183)

[Az aktuális időpont megtekintése 35](#_Toc175611184)

[AZ ADATBÁZIS STRUKTURÁJA 42](#_Toc175611185)

[A PROJEKT MŰKÖDÉSÉNEK LEÍRÁSA 43](#_Toc175611186)

[FELHASZNÁLT IRODALOM 47](#_Toc175611187)

# A FELADATKÖR MEGHATÁROZÁSA

A projekt webcíme: https://kv.stud.vts.su.ac.rs/

A project a Haladó Web programozás és Integrált web rendszerek tantárgyakból tevődik össze, mindkét tárgyból különböző elvárások voltak. A projektünk célja az volt, hogy létrehozzunk egy webes alkalmazást, és annak számára egy weboldalt, ez a weboldal pedig biztosítja a fodrászszalonoknak az elektronikus nyilvántartását.

Az alkalmazás hozzáférés tekintetében 4 szintre van felosztva: vendég, regisztrált illetve bejelentkezett felhasználó, szalon tulajdonosa, rendszergazda.

A vendég: megtekintheti az oldalon kínált információkat, megtekintheti a fodrásszalonok kinálatát, és lehetősége van regisztrálni a weboldalon

A szalon tulajdonosa: Létrehozhatja és törölheti a saját szolgáltatásait, napi rendszerességgel hozhat létre időpontfoglalást, megtekintheti az összes időpontfoglalást, illetve felverheti a kapcsolatot a felhasználóval is.

Rendszergazda: Felhasználói fiókokat hozhat létre a szalontulajdonosok számára, megtilthatja/ engedélyezheti a felhasználónak a rendszerbe történő bejelentkezési jogát.

Az integrált web rendszerek projektum elvárásai voltak, hogy a weboldal több platformos (responsive) legyen, DataTables-t kell használni az adatok megjelenítésére, és szerkesztésére. Detektálni kell a bejelentkezett felhasználó IP címét, és további adatait. Létre kell hozni egy mobilalkalmazást, legalább 4 funkcióval, ami kielégíti a CRUD funkciókat. Ezekhez API végpontokat kell létrehozni. Postman-el kell tesztelni az API-kat. Az API dokumentációt pedig swagger-el kell elkészíteni**.**

# PIACKUTATÁS

A következő weboldalakat néztük át:

* <https://www.sredime.rs/>
* <https://kliktermin.com/>
* https://sztakofodraszszalon.hu/

Visztont ezekből nem tudtunk megfelelően inspirálódni, így úgy döntöttünk, hogy saját ötleteket belevonva saját funkciókat határozunk meg, annak érdekében, hogy jobb legyen a fodrászszalon weboldalunk. Az admin felületetre fektettük a hangsulyt a plussz funkciók terén

#### Ban User NVE( notification via email)

Amikor kitiltunk egy felhasználót illetve feloldjuk ez a bizonyos felhasználó kap róla értesítést emailben.

<?php

// email\_notifications.php

require\_once "../../phpmailer/PHPMailerAutoload.php";

function sendNotificationEmail($email, $action, $username) {

$mail = new PHPMailer(true);

try {

//Server settings

$mail->isSMTP();

$mail->Host = 'smtp.gmail.com';

$mail->Port = 587;

$mail->SMTPAuth = true;

$mail->SMTPSecure = 'tls';

$mail->Username = 'vtsfrizer@gmail.com';

$mail->Password = 'uyuy ltjk rzfe mwvl';

//Recipients

$mail->setFrom('vtsfrizer@gmail.com', 'VTSFRIZER');

$mail->addAddress($email);

//Content

$mail->isHTML(true);

$mail->Subject = 'Fiokertesites';

switch ($action) {

case 'ban':

$mail->Body = "Tisztelt $username,<br><br>A fiókját letiltottuk. Ha úgy gondolja, hogy ez tévedés, kérjük, lépjen kapcsolatba az ügyfélszolgálattal.";

break;

case 'unban':

$mail->Body = "Tisztelt $username,<br><br>A fiókodat feloldottuk. Most már újra bejelentkezhetsz.";

break;

}

$mail->send();

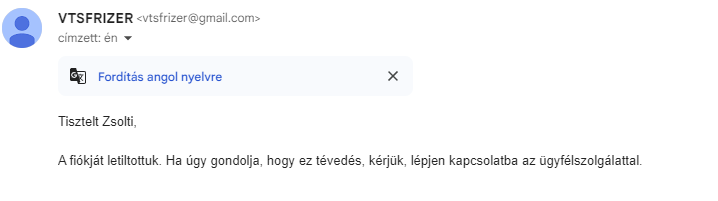
} catch (Exception $e) {

// Handle mail sending error, if needed

echo 'Message could not be sent. Mailer Error: ', $mail->ErrorInfo;

}

}



#### Banned User logs

Az admin felületen létrehoztunk egy történések oldalt, ahol az adminok láthatják, hogy a történéseket, található benne szűrő illetve van lehetőségek ezeket a logokat letölteni

<?php

require '../../db\_config.php';

// Initialize variables

$userIdFilter = '';

$dateFilter = '';

// Handle form submission

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

if (isset($\_POST['search'])) {

$userIdFilter = isset($\_POST['user\_id']) ? trim($\_POST['user\_id']) : '';

$dateFilter = isset($\_POST['date']) ? trim($\_POST['date']) : '';

} elseif (isset($\_POST['truncate'])) {

try {

$truncateQuery = "TRUNCATE TABLE activity\_logs";

$pdo->exec($truncateQuery);

$message = "Aktivitások sikeresen törölve";

} catch (PDOException $e) {

$message = "Error: " . $e->getMessage();

}

} elseif (isset($\_POST['export\_csv'])) {

// Handle CSV export

$userIdFilter = isset($\_POST['user\_id']) ? trim($\_POST['user\_id']) : '';

$dateFilter = isset($\_POST['date']) ? trim($\_POST['date']) : '';

try {

$query = "SELECT user\_id, action, timestamp FROM activity\_logs WHERE 1=1";

if ($userIdFilter) {

$query .= " AND user\_id = :user\_id";

}

if ($dateFilter) {

$query .= " AND DATE(timestamp) = :date";

}

$statement = $pdo->prepare($query);

if ($userIdFilter) {

$statement->bindParam(':user\_id', $userIdFilter, PDO::PARAM\_INT);

}

if ($dateFilter) {

$statement->bindParam(':date', $dateFilter);

}

$statement->execute();

$logs = $statement->fetchAll(PDO::FETCH\_ASSOC);

// Set CSV headers

header('Content-Type: text/csv');

header('Content-Disposition: attachment;filename="activity\_logs.csv"');

$output = fopen('php://output', 'w');

// Add CSV column headers

fputcsv($output, ['User ID', 'Action', 'Timestamp']);

// Add CSV rows

foreach ($logs as $log) {

fputcsv($output, $log);

}

fclose($output);

exit();

} catch (PDOException $e) {

echo "Error: " . $e->getMessage();

}

}

}

try {

// Prepare the query with filters

$query = "SELECT user\_id, action, timestamp FROM activity\_logs WHERE 1=1";

if ($userIdFilter) {

$query .= " AND user\_id = :user\_id";

}

if ($dateFilter) {

$query .= " AND DATE(timestamp) = :date";

}

$statement = $pdo->prepare($query);

// Bind parameters if they are set

if ($userIdFilter) {

$statement->bindParam(':user\_id', $userIdFilter, PDO::PARAM\_INT);

}

if ($dateFilter) {

$statement->bindParam(':date', $dateFilter);

}

$statement->execute();

// Fetch all the results

$logs = $statement->fetchAll(PDO::FETCH\_ASSOC);

} catch (PDOException $e) {

echo "Error: " . $e->getMessage();

}

include('includes/header.php');

?>

<div class="container">

<div class="row">

<div class="col-md-12">

<div class="card">

<div class="card-header">

<h4>Aktivitások</h4>

</div>

<div class="card-body">

<?php if (isset($message)): ?>

<div class="alert alert-info">

<?php echo htmlspecialchars($message); ?>

</div>

<?php endif; ?>

<form method="post" action="">

<div class="form-row">

<div class="form-group col-md-4">

<label for="user\_id">Azonosító</label>

<input type="number" id="user\_id" name="user\_id" class="form-control" value="<?php echo htmlspecialchars($userIdFilter); ?>">

</div>

<div class="form-group col-md-4">

<label for="date">Dátum</label>

<input type="date" id="date" name="date" class="form-control" value="<?php echo htmlspecialchars($dateFilter); ?>">

</div>

<div class="form-group col-md-4">

<label>&nbsp;</label>

<button type="submit" name="search" class="btn btn-primary form-control">Keresés</button>

<button type="submit" name="truncate" class="btn btn-danger form-control">Aktivitások törlése</button>

<button type="submit" name="export\_csv" class="btn btn-primary form-control">Exportálás CSV</button>

</div>

</div>

</form>

<table class="table table-striped mt-3">

<thead>

<tr>

<th>Felhasználó azonosítója</th>

<th>Akció</th>

<th>Időtartam</th>

</tr>

</thead>

<tbody>

<?php if ($logs): ?>

<?php foreach ($logs as $log): ?>

<tr>

<td><?php echo htmlspecialchars($log['user\_id']); ?></td>

<td><?php echo htmlspecialchars($log['action']); ?></td>

<td><?php echo htmlspecialchars($log['timestamp']); ?></td>

</tr>

<?php endforeach; ?>

<?php else: ?>

<tr>

<td colspan="3">Nincs megjeleníthető adat</td>

</tr>

<?php endif; ?>

</tbody>

</table>

</div>

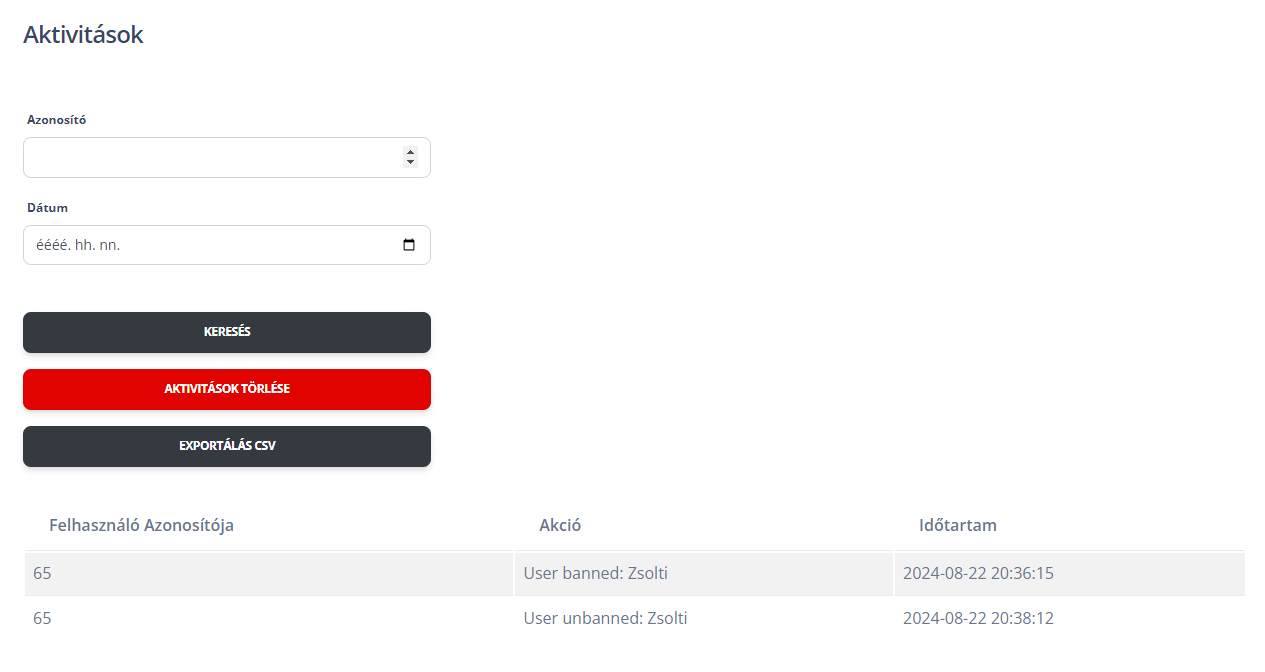
</div>

</div>

</div>

</div>

<?php include('includes/footer.php'); ?>



#### Az összes felhasználó

Az admin felületen létrehoztunk egy Az összes felhasználó tabot, ahol az admin láthatja a felhasználók adatait, illetve törölni tudja őket.

<?php

require '../../db\_config.php';

try {

// Prepare and execute the query to fetch all users

$query = "SELECT user\_id, username, last\_name, first\_name, email, mobile, active, is\_blocked FROM users";

$statement = $pdo->prepare($query);

$statement->execute();

// Fetch all the results

$users = $statement->fetchAll(PDO::FETCH\_ASSOC);

} catch (PDOException $e) {

echo "Error: " . $e->getMessage();

}

include('includes/header.php');

?>

<div class="container mt-4">

<div class="row">

<div class="col-md-12">

<div class="card">

<div class="card-header">

<h4>

Felhasználók listája

</h4>

</div>

<div class="card-body">

<table class="table table-striped table-bordered">

<thead class="table-dark">

<tr>

<th>Felhasználó ID</th>

<th>Felhasználónév</th>

<th>Vezetéknév</th>

<th>Keresztnév</th>

<th>Email</th>

<th>Mobil</th>

<th>Aktív</th>

<th>Blokkolva</th>

<th>Akciók</th>

</tr>

</thead>

<tbody id="user-table-body">

<?php if ($users): ?>

<?php foreach ($users as $user): ?>

<tr id="user-<?php echo htmlspecialchars($user['user\_id']); ?>">

<td><?php echo htmlspecialchars($user['user\_id']); ?></td>

<td><?php echo htmlspecialchars($user['username']); ?></td>

<td><?php echo htmlspecialchars($user['last\_name']); ?></td>

<td><?php echo htmlspecialchars($user['first\_name']); ?></td>

<td><?php echo htmlspecialchars($user['email']); ?></td>

<td><?php echo htmlspecialchars($user['mobile']); ?></td>

<td><?php echo $user['active'] ? '<span class="badge bg-success">Igen</span>' : '<span class="badge bg-secondary">Nem</span>'; ?></td>

<td><?php echo $user['is\_blocked'] ? '<span class="badge bg-danger">Igen</span>' : '<span class="badge bg-secondary">Nem</span>'; ?></td>

<td>

<button class="btn btn-danger btn-sm delete-btn" data-user-id="<?php echo htmlspecialchars($user['user\_id']); ?>">Törlés</button>

</td>

</tr>

<?php endforeach; ?>

<?php else: ?>

<tr>

<td colspan="9" class="text-center">Nincs megjeleníthető adat.</td>

</tr>

<?php endif; ?>

</tbody>

</table>

</div>

</div>

</div>

</div>

</div>

<?php include('includes/footer.php'); ?>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<script>

$(document).ready(function() {

$('.delete-btn').click(function() {

var userId = $(this).data('user-id');

var row = $(this).closest('tr');

if (confirm('Biztosan törölni szeretnéd ezt a felhasználót?')) {

$.ajax({

url: 'delete\_user.php',

type: 'POST',

data: { user\_id: userId },

dataType: 'json', // Expect JSON response

success: function(response) {

if (response.status === 'success') {

row.remove();

} else {

alert('Hiba történt a törlés során: ' + (response.message || 'Ismeretlen hiba'));

}

},

error: function(xhr, status, error) {

console.log(xhr.responseText); // Részletesebb hibaüzenet a konzolon

alert('AJAX hiba: ' + error);

}

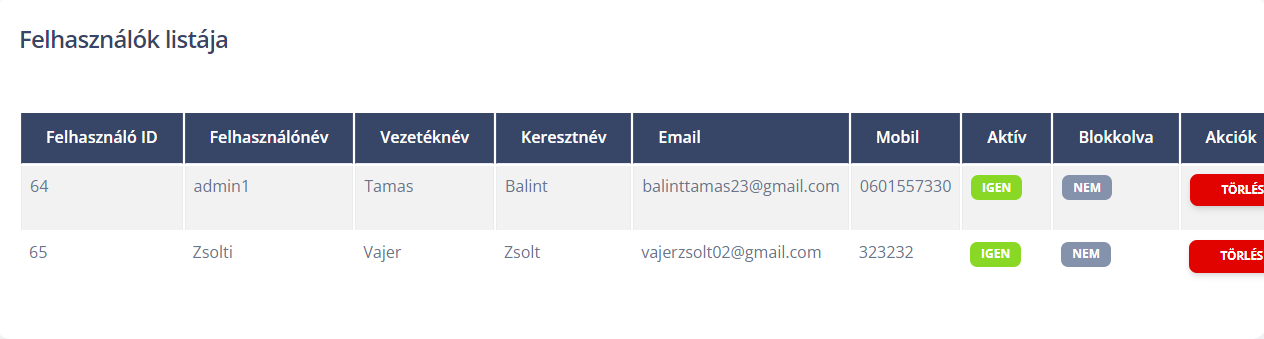
});

}

});

});

</script>



# KIDOLGOZÁS

Ebben a részben kell feltüntetni a kód részeit (tetszőlegesen kiválaszthatja a kód részeit). A program kódját **Courier** betűtípussal, **10-es betűmérettel** kell írni. A leírás alkalmával kötelezően el kell határolni a projektum két részét. **Kötelező az API-végpontok leírásá amelyek a CRUD funkciókat végzík.** Ha több ilyen API-végpont van, válasszon egyet minden CRUD funkcióhoz tetszőlegesen.

## Haladó web programozás

### Felhasználói felület

#### Felhasználó bejelentkezése

Ez a kód egy bejelentkezési rendszer része, amely ellenőrzi a felhasználó hitelesítési adatait, és ha helyesek, elmenti az eszköz adatait (IP-cím, böngésző, operációs rendszer) egy adatbázisba. Az eszközadatokat egy külön osztály (`DeviceRepository`) segítségével gyűjti össze. Emellett a kód kezeli a munkameneteket, és ha a bejelentkezés sikeres, elmenti a felhasználó adatait a munkamenetbe. Ha a hitelesítés sikertelen, hibaüzeneteket ad vissza.

require\_once 'device.repository.php';

$user\_agent = $\_SERVER['HTTP\_USER\_AGENT'];

$device\_repo = new DeviceRepository($user\_agent);

if (session\_status() == PHP\_SESSION\_NONE) {

session\_start();

}

$isLoggedIn = isset($\_SESSION['username']);

$username = $isLoggedIn ? $\_SESSION['username'] : '';

require\_once 'db\_config.php';

function collectdata($dsn,$pdoOptions,$device\_repo)

{

try {

$conn = new PDO($dsn, PARAMS['USER'], PARAMS['PASSWORD'], $pdoOptions);

$conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$query = "INSERT INTO frizer.user\_info (ip, device, os, browser, date) VALUES (:ip, :device, :os, :browser, :date)";

$stmt =$conn->prepare($query);

$ip = $device\_repo->get\_ip();

$device = $device\_repo->get\_device();

$os = $device\_repo->get\_os();

$browser = $device\_repo->get\_browser();

$date = date('Y-m-d H:i:s');

$stmt->bindParam("ip", $ip);

$stmt->bindParam("device", $device);

$stmt->bindParam("os", $os);

$stmt->bindParam("browser", $browser);

$stmt->bindParam("date", $date);

$stmt->execute();

} catch (PDOException $exc) {

echo $exc->getMessage();

}

}

function sanitize($data)

{

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

if (isset($\_POST['login'])) {

$errors = array();

$email = filter\_var(trim($\_POST['email']), FILTER\_SANITIZE\_EMAIL);

$username = filter\_var(trim($\_POST['username']), FILTER\_SANITIZE\_STRING);

$password = trim($\_POST['password']);

if (empty($email)) {

$errors[] = 'Email is required';

} elseif (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

$errors[] = 'Invalid email';

}

if (empty($username)) {

$errors[] = 'Username is required';

}

if (empty($password)) {

$errors[] = 'Password is required';

}

if (empty($errors)) {

try {

$conn = new PDO($dsn, PARAMS['USER'], PARAMS['PASSWORD'], $pdoOptions);

$conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

// Ez a querry 2 dolgot néz meg , az egyik az hogy a felhasználó ki van e tiltva, illetve hogy verifikálta-e.

$sql = "SELECT \* FROM users WHERE email = :email AND active = 1 AND is\_blocked = 0";

$stmt = $conn->prepare($sql);

$stmt->bindParam(':email', $email);

$stmt->execute();

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($stmt->rowCount() > 0) {

if (password\_verify($password, $result['password'])) {

$\_SESSION['user\_id'] = $result['user\_id'];

$\_SESSION['username'] = $result['username'];

$\_SESSION['email'] = $result['email'];

$\_SESSION['active'] = $result['active'];

header('Location: index.php');

collectdata($dsn,$pdoOptions,$device\_repo);

exit();

} else {

$errors[] = 'Hibás jelszó';

}

} else {

$errors[] = 'A felhasználói fiókod le van tiltva, vagy nem verifikáltad!';

}

} catch (PDOException $e) {

$errors[] = 'Error logging in: ' . $e->getMessage();

} finally {

$conn = null;

}

}

if (!empty($errors)) {

$\_SESSION['errors'] = $errors;

header("Location: login.php");

exit();

}

}

?>

#### Időpontok megtekintése

A weboldal e részén a felhasználó megtekintheti az időpontfoglalását, és ugyanakkor lemondhatja azt

session\_start();

require\_once 'db\_config.php';

if (!isset($\_SESSION['user\_id'])) {

header("Location: login.php");

exit;

}

$isLoggedIn = isset($\_SESSION['username']);

$username = $isLoggedIn ? $\_SESSION['username'] : '';

$user\_id = $\_SESSION['user\_id'];

try {

$pdo = new PDO($dsn, PARAMS['USER'], PARAMS['PASSWORD'], $pdoOptions);

} catch (PDOException $e) {

echo "Error connecting to the database: " . $e->getMessage();

exit;

}

if (isset($\_POST['delete']) && isset($\_POST['reservation\_id'])) {

$reservation\_id = $\_POST['reservation\_id'];

$updateQuery = "UPDATE reservations SET cancelled = 1 WHERE reservation\_id = ? AND user\_id = ? AND TIMESTAMP(date, start\_time) > DATE\_ADD(NOW(), INTERVAL 1 HOUR)";

$updateStmt = $pdo->prepare($updateQuery);

$updateStmt->execute([$reservation\_id, $user\_id]);

}

$queryPast = "SELECT r.reservation\_id, r.date, r.start\_time, r.end\_time, r.user\_id, f.frizer\_name, t.treatment\_name, r.comment

FROM reservations r

JOIN frizers f ON r.frizer\_id = f.frizer\_id

JOIN treatment t ON r.treatment\_id = t.treatment\_id

WHERE r.user\_id = ? AND (r.date < CURDATE() OR (r.date = CURDATE() AND r.end\_time <= CURTIME())) AND r.cancelled = 0";

$stmtPast = $pdo->prepare($queryPast);

$stmtPast->execute([$user\_id]);

$pastAppointments = $stmtPast->fetchAll(PDO::FETCH\_ASSOC);

$queryActive = "SELECT r.reservation\_id, r.date, r.start\_time, r.end\_time, r.user\_id, f.frizer\_name, t.treatment\_name, r.comment

FROM reservations r

JOIN frizers f ON r.frizer\_id = f.frizer\_id

JOIN treatment t ON r.treatment\_id = t.treatment\_id

WHERE r.user\_id = ? AND (r.date > CURDATE() OR (r.date = CURDATE() AND r.end\_time > CURTIME())) AND r.cancelled = 0";

$stmtActive = $pdo->prepare($queryActive);

$stmtActive->execute([$user\_id]);

$activeAppointments = $stmtActive->fetchAll(PDO::FETCH\_ASSOC);

$queryCancelled = "SELECT r.reservation\_id, r.date, r.start\_time, r.end\_time, r.user\_id, f.frizer\_name, t.treatment\_name, r.comment

FROM reservations r

JOIN frizers f ON r.frizer\_id = f.frizer\_id

JOIN treatment t ON r.treatment\_id = t.treatment\_id

WHERE r.user\_id = ? AND r.cancelled = 1";

$stmtCancelled = $pdo->prepare($queryCancelled);

$stmtCancelled->execute([$user\_id]);

$cancelledAppointments = $stmtCancelled->fetchAll(PDO::FETCH\_ASSOC);

Ez a kód egy felhasználó foglalásait kezeli egy adatbázisból. Először ellenőrzi, hogy a felhasználó be van-e jelentkezve, és ha nem, átirányítja a bejelentkezési oldalra. Ezután csatlakozik az adatbázishoz, és lehetőséget biztosít a felhasználónak arra, hogy törölje egy jövőbeli foglalását. A kód három lekérdezést hajt végre:

Korábbi foglalások: Azokat a foglalásokat gyűjti össze, amelyek már lezárultak.

Aktív foglalások: A jövőbeli, még érvényes foglalásokat listázza.

Lemondott foglalások: A korábban lemondott foglalásokat gyűjti össze.

Mindegyik lekérdezés összekapcsolja a foglalásokat a megfelelő fodrász és kezelés adataival az adatbázisból.

#### Időpontfoglalás

A weboldalon lehetőségünk van időpontot foglalni különböző fodrásszalonokhoz.

require\_once 'db\_config.php';

require\_once 'functions.php';

if (session\_status() == PHP\_SESSION\_NONE) {

session\_start();

}

$isLoggedIn = isset($\_SESSION['username']);

$username = $isLoggedIn ? $\_SESSION['username'] : '';

$user\_id = $\_SESSION['user\_id'];

$duration = 30;

$cleanup = 0;

$start = "09:00";

$end = "15:01";

try {

$pdo = new PDO($dsn, PARAMS['USER'], PARAMS['PASSWORD'], $pdoOptions);

} catch (PDOException $e) {

echo "Error connecting to the database: " . $e->getMessage();

exit;

}

$allReservations = getAllReservations();

$e = $\_GET['e'] ?? 0;

if ($e == 1) {

echo '<script>alert("Töltsd ki az összes mezőt!")</script>';

}

if ($e == 3) {

echo '<script>alert("Sikeres foglalás!")</script>';

}

A getAllReservations() függvény segítségével lekéri az összes nem lemondott foglalást.

#### Szalon tulajdonos

A weboldalanak ezen részén a szalon tulajdonosa megtekintheti az aktiv, lemondott és archivált foglalásokat, illetve a szolgáltatásokat törölhet, módosithat, és létrehozhat, datatable segítségével:

require\_once 'db\_config.php';

session\_start();

if (isset($\_SESSION['salon\_owner\_id'])) {

$salon\_id\_owner = $\_SESSION['salon\_owner\_id'];

// echo "Salon ID Owner: " . $salon\_id\_owner; // For testing purposes

} else {

echo "No salon\_owner\_id found in session.";

exit(); // Terminate script if session variable not set

}

$dsn = "mysql:host=" . PARAMS['HOST'] . ";dbname=" . PARAMS['DB'] . ";charset=" . PARAMS['CHARSET'];

$options = [

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION,

PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_ASSOC,

PDO::ATTR\_EMULATE\_PREPARES => false

];

try {

$pdo = new PDO($dsn, PARAMS['USER'], PARAMS['PASSWORD'], $options);

} catch (PDOException $e) {

die('Connection failed: ' . $e->getMessage());

}

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$filter = isset($\_POST['filter']) ? $\_POST['filter'] : 'all';

$salon\_id\_owner = $\_SESSION['salon\_owner\_id'];

$query = "SELECT r.date, u.username AS user\_id, f.frizer\_name AS frizer\_id, t.treatment\_name AS treatment\_id, r.comment, r.cancelled

FROM reservations r

JOIN users u ON r.user\_id = u.user\_id

JOIN frizers f ON r.frizer\_id = f.frizer\_id

JOIN treatment t ON r.treatment\_id = t.treatment\_id";

if ($filter === 'active') {

$query .= " WHERE r.cancelled = 0 AND r.date > NOW()";

} elseif ($filter === 'archived') {

$query .= " WHERE r.cancelled = 0 AND r.date <= NOW()";

} elseif ($filter === 'cancelled') {

$query .= " WHERE r.cancelled = 1";

}

$stmt = $pdo->prepare($query);

$stmt->execute();

$reservations = $stmt->fetchAll();

$data = [];

foreach ($reservations as $reservation) {

$data[] = [

'date' => $reservation['date'],

'user\_id' => $reservation['user\_id'],

'frizer\_id' => $reservation['frizer\_id'],

'treatment\_id' => $reservation['treatment\_id'],

'comment' => $reservation['comment'],

'cancelled' => $reservation['cancelled']

];

}

echo json\_encode(['data' => $data]);

exit();

}

try {

$query = "

SELECT t.treatment\_id, t.salon\_id, t.treatment\_name, t.treatment\_price

FROM treatment t

JOIN salon\_owner so ON t.salon\_id = so.salon\_owner\_id

WHERE so.salon\_owner\_id = :salon\_owner\_id

";

// Prepare the statement

$stmt = $pdo->prepare($query);

$stmt->bindParam(':salon\_owner\_id', $salon\_id\_owner, PDO::PARAM\_INT);

$stmt->execute();

$rows = $stmt->fetchAll(PDO::FETCH\_ASSOC);

} catch (PDOException $e) {

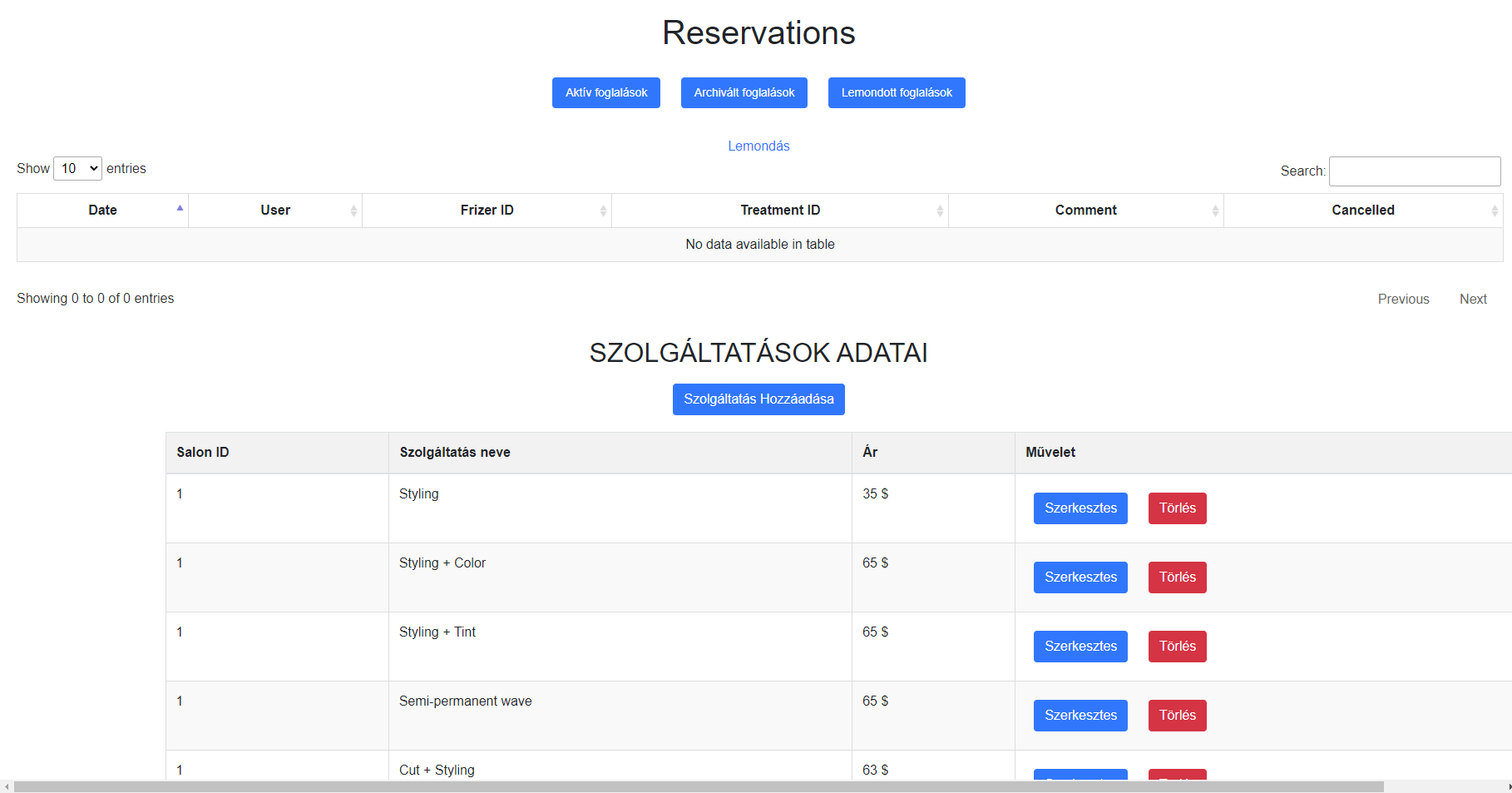
die("Error: " . $e->getMessage());

}

A foglalásokat különböző szűrők alapján lekérdezi (aktív, archivált, lemondott), és JSON formátumban visszaküldi az adatokat. Lekéri a szalon tulajdonosához tartozó kezeléseket, és az adatokat felhasználásra előkészíti.

## Integrált web rendszerek

A szalon owner felületet DataTablessel oldottuk meg ami a felhasználók kezelésénel a következő képpen néz ki:



### Hozzáférési eszközök észlelése

Erre a célra a Mobile Detect PHP könyvtárat használjuk.

<?php

require\_once \_\_DIR\_\_ . "/../vendor/autoload.php";

use Detection\MobileDetect;

class DeviceRepository

{

private $detect;

private $user\_agent;

public function \_\_construct($user\_agent)

{

$this->user\_agent = $user\_agent;

$this->detect = new MobileDetect();

}

public function get\_device()

{

if ($this->detect->isTablet())

return "Tablet";

else if ($this->detect->isMobile())

return "Mobile";

else

return "Desktop";

}

public function get\_ip()

{

$ipServiceUrl = 'https://api.ipify.org?format=json';

$response = file\_get\_contents($ipServiceUrl);

$data = json\_decode($response, true);

return $data['ip'];

}

function get\_os()

{

$osArray = [

'Windows NT 10.0' => 'Windows 10',

'Windows NT 6.3' => 'Windows 8.1',

'Windows NT 6.2' => 'Windows 8',

'Windows NT 6.1' => 'Windows 7',

'Windows NT 6.0' => 'Windows Vista',

'Windows NT 5.1' => 'Windows XP',

'Mac OS X' => 'Mac OS X',

'Android' => 'Android',

'iPhone' => 'iOS',

'iPad' => 'iOS',

'Linux' => 'Linux',

'Ubuntu' => 'Ubuntu'

];

foreach ($osArray as $regex => $value) {

if (strpos($this->user\_agent, $regex) !== false) {

return $value;

}

}

return 'Unknown';

}

function get\_browser()

{

$browsers = [

'Opera' => 'Opera',

'OPR/' => 'Opera',

'Edge' => 'Edge',

'Chrome' => 'Chrome',

'Safari' => 'Safari',

'Firefox' => 'Firefox',

'MSIE' => 'Internet Explorer',

'Trident/7' => 'Internet Explorer'

];

foreach ($browsers as $browser => $name) {

if (strpos($this->user\_agent, $browser) !== false) {

return $name;

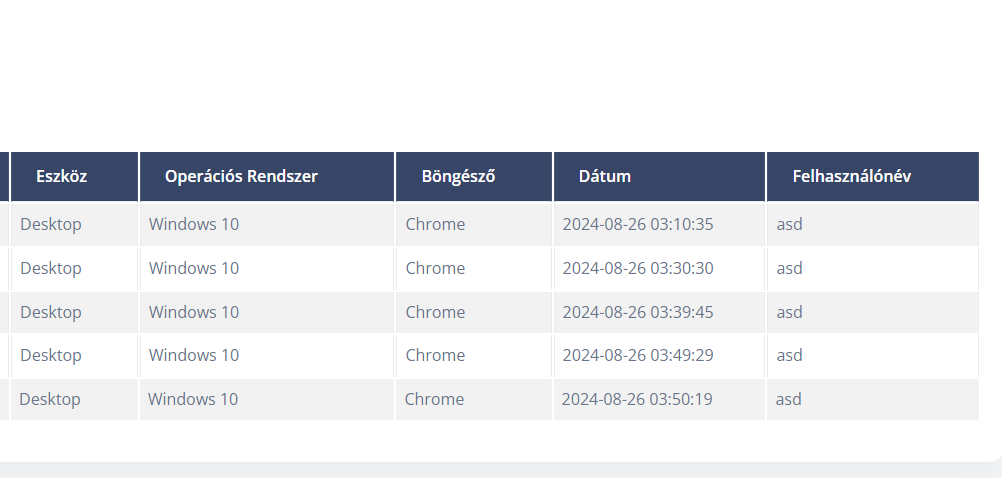
}

}

return 'Unknown';

}

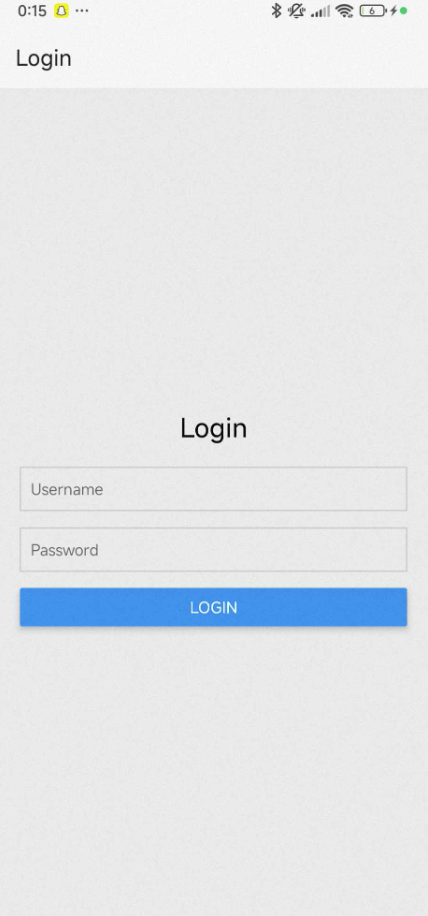
}



1. ábra: Az admin felület

### Mobil alkalmazás

#### A mobil alkalmazás login felülete:



import React, { useState } from 'react';

import { View, Text, TextInput, Button, StyleSheet, Alert } from 'react-native';

import axios from 'axios';

const LoginScreen = ({ navigation }) => {

const [username, setUsername] = useState('');

const [password, setPassword] = useState('');

const handleLogin = () => {

axios.post('http://192.168.0.16:3000/login', {

username,

password,

})

.then(response => {

if (response.data.success) {

const userId = response.data.userId;

navigation.navigate('Home', { userId });

} else {

Alert.alert('Login Failed', response.data.message || 'Invalid credentials');

}

})

.catch(error => {

console.error('There was an error!', error);

Alert.alert('Error', 'There was an error connecting to the server.');

});

};

return (

<View style={styles.container}>

<Text style={styles.title}>Login</Text>

<TextInput

style={styles.input}

placeholder="Username"

value={username}

onChangeText={setUsername}

/>

<TextInput

style={styles.input}

placeholder="Password"

secureTextEntry

value={password}

onChangeText={setPassword}

/>

<Button title="Login" onPress={handleLogin} />

</View>

);

};

const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: 'center',

paddingHorizontal: 20,

},

title: {

fontSize: 24,

fontWeight: 'bold',

marginBottom: 20,

textAlign: 'center',

},

input: {

height: 40,

borderColor: '#ccc',

borderWidth: 1,

marginBottom: 15,

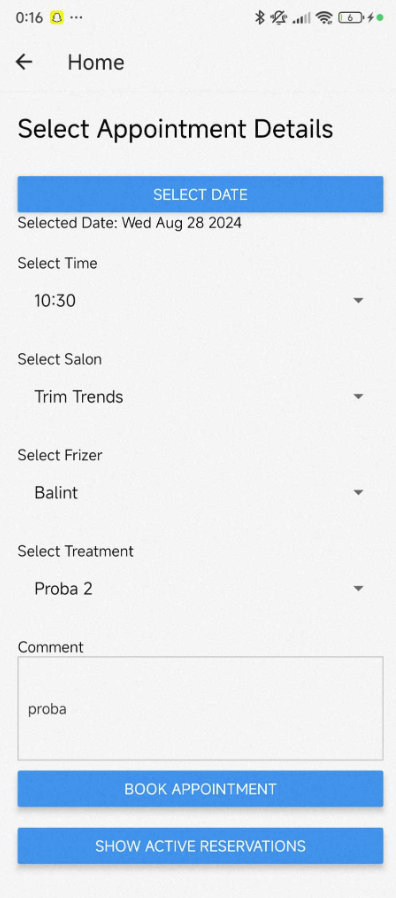
paddingHorizontal: 10,

},

});

export default LoginScreen;

#### A mobil alkalmazás fő oldala:



import React, { useState, useEffect } from 'react';

import { View, Text, StyleSheet, Button, TextInput } from 'react-native';

import DateTimePicker from '@react-native-community/datetimepicker';

import { Picker } from '@react-native-picker/picker';

import axios from 'axios';

const HomeScreen = ({ route, navigation }) => {

const [date, setDate] = useState(new Date());

const [showDatePicker, setShowDatePicker] = useState(false);

const [selectedTime, setSelectedTime] = useState('');

const [selectedSalon, setSelectedSalon] = useState('');

const [selectedFrizer, setSelectedFrizer] = useState('');

const [selectedTreatment, setSelectedTreatment] = useState('');

const [frizers, setFrizers] = useState([]);

const [treatments, setTreatments] = useState([]);

const [availableTimes, setAvailableTimes] = useState([

'09:00', '09:30', '10:00', '10:30', '11:00', '11:30',

'12:00', '12:30', '13:00', '13:30', '14:00', '14:30'

]);

const [comment, setComment] = useState(''); // Added state for comment

const { userId } = route.params;

useEffect(() => {

if (selectedSalon) {

axios.get(`http://192.168.0.16:3000/frizers?salon\_id=${selectedSalon}`)

.then(response => setFrizers(response.data))

.catch(error => console.error('Error fetching frizers:', error));

axios.get(`http://192.168.0.16:3000/treatments?salon\_id=${selectedSalon}`)

.then(response => setTreatments(response.data))

.catch(error => console.error('Error fetching treatments:', error));

}

}, [selectedSalon]);

const handleReservation = () => {

const startTime = `${date.toISOString().split('T')[0]} ${selectedTime}:00`;

const startDate = new Date(startTime);

const endDate = new Date(startDate.getTime() + 30 \* 60000);

const formattedEndTime = `${endDate.toISOString().split('T')[0]} ${endDate.toTimeString().split(' ')[0].slice(0, 5)}`;

const reservationData = {

store\_id: selectedSalon,

date: date.toISOString().split('T')[0], // Added date field

start\_time: startTime,

end\_time: formattedEndTime,

user\_id: userId,

frizer\_id: selectedFrizer,

treatment\_id: selectedTreatment,

comment: comment // Added comment field

};

axios.post('http://192.168.0.16:3000/makeReservation', reservationData)

.then(response => {

alert('Reservation successful!');

})

.catch(error => {

console.error('Error creating reservation:', error);

alert('Failed to create reservation.');

});

};

const handleShowActiveReservations = () => {

navigation.navigate('ActiveReservations', { userId: userId });

};

return (

<View style={styles.container}>

<Text style={styles.title}>Select Appointment Details</Text>

<View style={styles.pickerContainer}>

<Button onPress={() => setShowDatePicker(true)} title="Select Date" />

{showDatePicker && (

<DateTimePicker

value={date}

mode="date"

display="default"

onChange={(event, selectedDate) => {

const currentDate = selectedDate || date;

setShowDatePicker(false);

setDate(currentDate);

}}

/>

)}

<Text>Selected Date: {date.toDateString()}</Text>

</View>

{/\* Time Picker \*/}

<View style={styles.pickerContainer}>

<Text>Select Time</Text>

<Picker

selectedValue={selectedTime}

onValueChange={(itemValue) => setSelectedTime(itemValue)}

>

{availableTimes.map(time => (

<Picker.Item key={time} label={time} value={time} />

))}

</Picker>

</View>

{/\* Salon Picker \*/}

<View style={styles.pickerContainer}>

<Text>Select Salon</Text>

<Picker

selectedValue={selectedSalon}

onValueChange={(itemValue) => setSelectedSalon(itemValue)}

>

<Picker.Item label="Select a Salon" value="" />

<Picker.Item label="Hairway to Heaven" value="1" />

<Picker.Item label="Trim Trends" value="2" />

<Picker.Item label="Hair Hub" value="3" />

<Picker.Item label="Timeless Trims" value="4" />

<Picker.Item label="Vajers Place" value="5" />

<Picker.Item label="Bark and Bath Boutique" value="6" />

</Picker>

</View>

{/\* Frizer Picker \*/}

<View style={styles.pickerContainer}>

<Text>Select Frizer</Text>

<Picker

selectedValue={selectedFrizer}

onValueChange={(itemValue) => setSelectedFrizer(itemValue)}

enabled={frizers.length > 0}

>

<Picker.Item label="Select a Frizer" value="" />

{frizers.map(frizer => (

<Picker.Item key={frizer.frizer\_id} label={frizer.frizer\_name} value={frizer.frizer\_id} />

))}

</Picker>

</View>

{/\* Treatment Picker \*/}

<View style={styles.pickerContainer}>

<Text>Select Treatment</Text>

<Picker

selectedValue={selectedTreatment}

onValueChange={(itemValue) => setSelectedTreatment(itemValue)}

enabled={treatments.length > 0}

>

<Picker.Item label="Select a Treatment" value="" />

{treatments.map(treatment => (

<Picker.Item key={treatment.treatment\_id} label={treatment.treatment\_name} value={treatment.treatment\_id} />

))}

</Picker>

</View>

<View style={styles.textAreaContainer}>

<Text>Comment</Text>

<TextInput

style={styles.textArea}

multiline

numberOfLines={4}

value={comment}

onChangeText={setComment}

placeholder="Write your comment here..."

/>

</View>

<Button title="Book Appointment" onPress={handleReservation} />

<View style={{ marginTop: 20 }}>

<Button title="Show Active Reservations" onPress={handleShowActiveReservations} />

</View>

</View>

);

};

const styles = StyleSheet.create({

container: {

flex: 1,

padding: 20,

backgroundColor: '#fff',

},

title: {

fontSize: 24,

fontWeight: 'bold',

marginBottom: 20,

},

pickerContainer: {

marginVertical: 10,

},

textAreaContainer: {

marginVertical: 10,

},

textArea: {

height: 100,

borderColor: '#ccc',

borderWidth: 1,

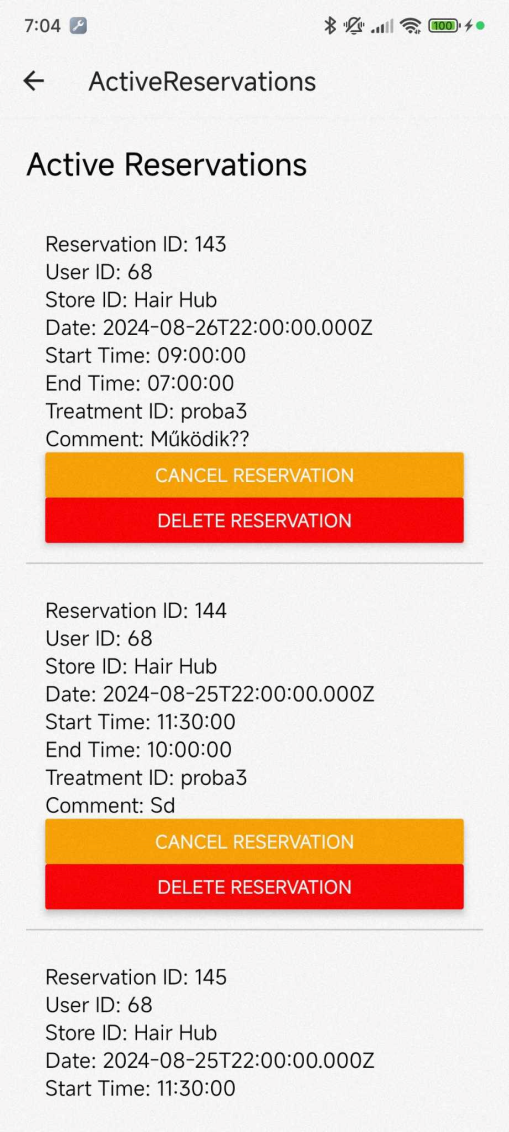
padding: 10,

},

});

export default HomeScreen;

#### Az aktuális időpont megtekintése és lemondása - törlése



import React, { useEffect, useState, useCallback } from 'react';

import { View, Text, StyleSheet, FlatList, Button } from 'react-native';

import axios from 'axios';

const salonNames = {

  1: 'Hairway to Heaven',

  2: 'Trim Trends',

  3: 'Hair Hub',

  4: 'Timeless Trims',

  5: 'Vajers Place',a

  6: 'Bark and Bath Boutique'

};

const ActiveReservations = ({ route }) => {

  const { userId } = route.params;

  const [reservations, setReservations] = useState([]);

  const [loading, setLoading] = useState(true);

  const fetchActiveReservations = useCallback(async () => {

    try {

      setLoading(true);

      const response = await axios.get(`http://192.168.0.16:3000/reservations?user\_id=${userId}`);

      setReservations(response.data);

      setLoading(false);

    } catch (error) {

      console.error('Error fetching reservations:', error);

      alert('Failed to fetch reservations.');

      setLoading(false);

    }

  }, [userId]);

  useEffect(() => {

    fetchActiveReservations();

  }, [fetchActiveReservations]);

  const cancelReservation = async (reservationId) => {

    try {

      await axios.patch(`http://192.168.0.16:3000/reservations/${reservationId}/cancel`);

      alert('Reservation cancelled successfully.');

      fetchActiveReservations();

    } catch (error) {

      console.error('Error cancelling reservation:', error);

      alert('Failed to cancel reservation.');

    }

  };

  const deleteReservation = async (reservationId) => {

    try {

      await axios.delete(`http://192.168.0.16:3000/reservations/${reservationId}`);

      alert('Reservation deleted successfully.');

      fetchActiveReservations();

    } catch (error) {

      console.error('Error deleting reservation:', error);

      alert('Failed to delete reservation.');

    }

  };

  const renderReservation = ({ item }) => (

    <View style={styles.reservationItem}>

      <Text style={styles.reservationText}>Reservation ID: {item.reservation\_id}</Text>

      <Text style={styles.reservationText}>User ID: {item.user\_id}</Text>

      <Text style={styles.reservationText}>Store ID: {salonNames[item.store\_id]}</Text>

      <Text style={styles.reservationText}>Date: {item.date}</Text>

      <Text style={styles.reservationText}>Start Time: {item.start\_time}</Text>

      <Text style={styles.reservationText}>End Time: {item.end\_time}</Text>

      <Text style={styles.reservationText}>Treatment Name: {item.treatment\_name}</Text>

      <Text style={styles.reservationText}>Comment: {item.comment}</Text>

      <Button

        title="Cancel Reservation"

        onPress={() => cancelReservation(item.reservation\_id)}

        color="orange"

      />

      <Button

        title="Delete Reservation"

        onPress={() => deleteReservation(item.reservation\_id)}

        color="red"

      />

    </View>

  );

  return (

    <View style={styles.container}>

      <Text style={styles.title}>Active Reservations</Text>

      {loading ? (

        <Text>Loading...</Text>

      ) : reservations.length > 0 ? (

        <FlatList

          data={reservations}

          keyExtractor={(item) => item.user\_id.toString()}

          renderItem={renderReservation}

        />

      ) : (

        <Text>No active reservations found.</Text>

      )}

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    padding: 20,

    backgroundColor: '#fff',

  },

  title: {

    fontSize: 24,

    fontWeight: 'bold',

    marginBottom: 20,

  },

  reservationItem: {

    padding: 15,

    borderBottomColor: '#ccc',

    borderBottomWidth: 1,

    marginBottom: 10,

  },

  reservationText: {

    fontSize: 16,

  },

});

export default ActiveReservations;

**A mobilalkalmazás SQL manipulációja része**

const express = require('express');

const mysql = require('mysql');

const cors = require('cors');

const app = express();

app.use(cors());

const db = mysql.createConnection({

  host: 'localhost',

  user: 'root',

  password: '',

  database: 'frizer'

});

db.connect((err) => {

  if (err) {

    console.error('Error connecting to the database:', err);

    return;

  }

  console.log('Connected to the MySQL database.');

});

app.post('/login', (req, res) => {

  let body = '';

  req.on('data', chunk => {

    body += chunk.toString();

  });

  req.on('end', () => {

    try {

      const { username, password } = JSON.parse(body);

      const query = 'SELECT \* FROM users WHERE username = ?';

      db.query(query, [username], (err, results) => {

        if (err) {

          res.status(500).json({ error: 'Database error' });

        } else if (results.length > 0) {

          const user = results[0];

          if (password === user.password) {

            res.json({ success: true, userId: user.user\_id });

          } else {

            res.json({ success: false, message: 'Incorrect password' });

          }

        } else {

          res.json({ success: false, message: 'User not found' });

        }

      });

    } catch (e) {

      res.status(400).json({ error: 'Invalid JSON' });

    }

  });

});

app.get('/frizers', (req, res) => {

  const salon\_id = req.query.salon\_id;

  const query = 'SELECT \* FROM frizers WHERE salon\_id = ?';

  db.query(query, [salon\_id], (err, results) => {

    if (err) {

      res.status(500).json({ error: err });

    } else {

      res.json(results);

    }

  });

});

app.get('/treatments', (req, res) => {

  const { salon\_id } = req.query;

  const query = 'SELECT \* FROM treatment WHERE salon\_id = ?';

  db.query(query, [salon\_id], (err, results) => {

    if (err) {

      res.status(500).json({ error: err });

    } else {

      res.json(results);

    }

  });

});

app.post('/makeReservation', (req, res) => {

  let body = '';

  req.on('data', chunk => {

    body += chunk.toString();

  });

  req.on('end', () => {

    try {

      const { store\_id, date, start\_time, end\_time, user\_id, frizer\_id, treatment\_id, comment } = JSON.parse(body); //

      if (!store\_id || !date || !start\_time || !end\_time || !user\_id || !frizer\_id || !treatment\_id) {

        return res.status(400).json({ error: 'Missing required fields' });

      }

      const query = `

        INSERT INTO reservations (

          store\_id, date, start\_time, end\_time, user\_id, frizer\_id, treatment\_id, comment, cancelled

        ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)`;

      db.query(query, [

        store\_id, date, start\_time, end\_time, user\_id, frizer\_id, treatment\_id,

        comment || null,

        0

      ], (err, results) => {

        if (err) {

          res.status(500).json({ error: 'Database error' });

        } else {

          res.json({ success: true, message: 'Reservation created successfully' });

        }

      });

    } catch (e) {

      res.status(400).json({ error: 'Invalid JSON' });

    }

  });

});

app.get('/reservations', (req, res) => {

  const userId = req.query.user\_id;

  if (!userId) {

    return res.status(400).json({ error: 'Missing user\_id parameter' });

  }

  const query = `

    SELECT r.reservation\_id, r.user\_id, r.store\_id, r.date, r.start\_time, r.end\_time, t.treatment\_name, r.comment

    FROM reservations r

    JOIN treatment t ON r.treatment\_id = t.treatment\_id

    WHERE r.user\_id = ? AND r.cancelled = 0

  `;

  db.query(query, [userId], (err, results) => {

    if (err) {

      console.error('Error fetching reservations:', err);

      res.status(500).json({ error: 'Database error' });

    } else {

      res.json(results);

    }

  });

});

app.patch('/reservations/:reservationId/cancel', (req, res) => {

  const { reservationId } = req.params;

  const query = 'UPDATE reservations SET cancelled = 1 WHERE reservation\_id = ?';

  db.query(query, [reservationId], (err, results) => {

    if (err) {

      console.error('Error cancelling reservation:', err);

      res.status(500).json({ error: 'Database error' });

    } else {

      res.json({ success: true, message: 'Reservation cancelled successfully' });

    }

  });

});

app.delete('/reservations/:reservationId', (req, res) => {

  const { reservationId } = req.params;

  const query = 'DELETE FROM reservations WHERE reservation\_id = ?';

  db.query(query, [reservationId], (err, results) => {

    if (err) {

      console.error('Error deleting reservation:', err);

      res.status(500).json({ error: 'Database error' });

    } else {

      res.json({ success: true, message: 'Reservation deleted successfully' });

    }

  });

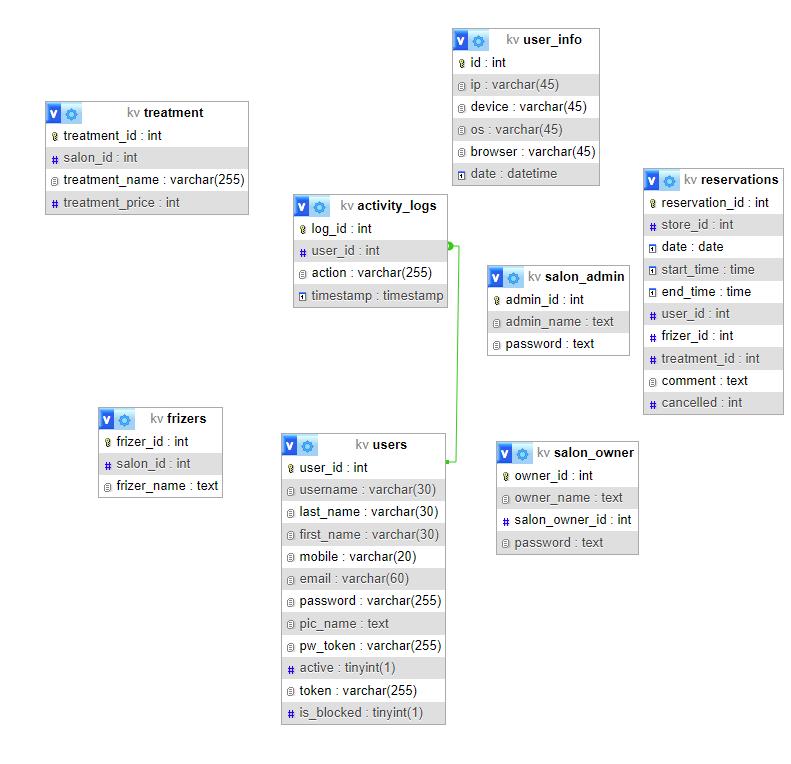
});

app.listen(3000, () => {

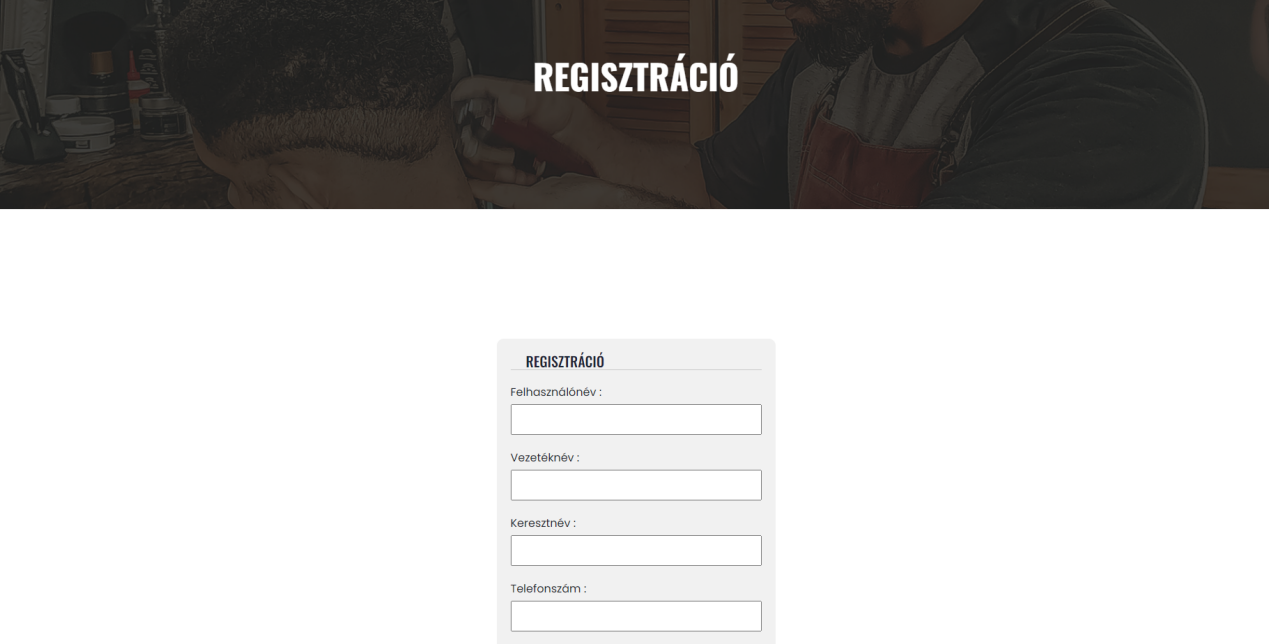
  console.log('Server is running on port 3000');

});

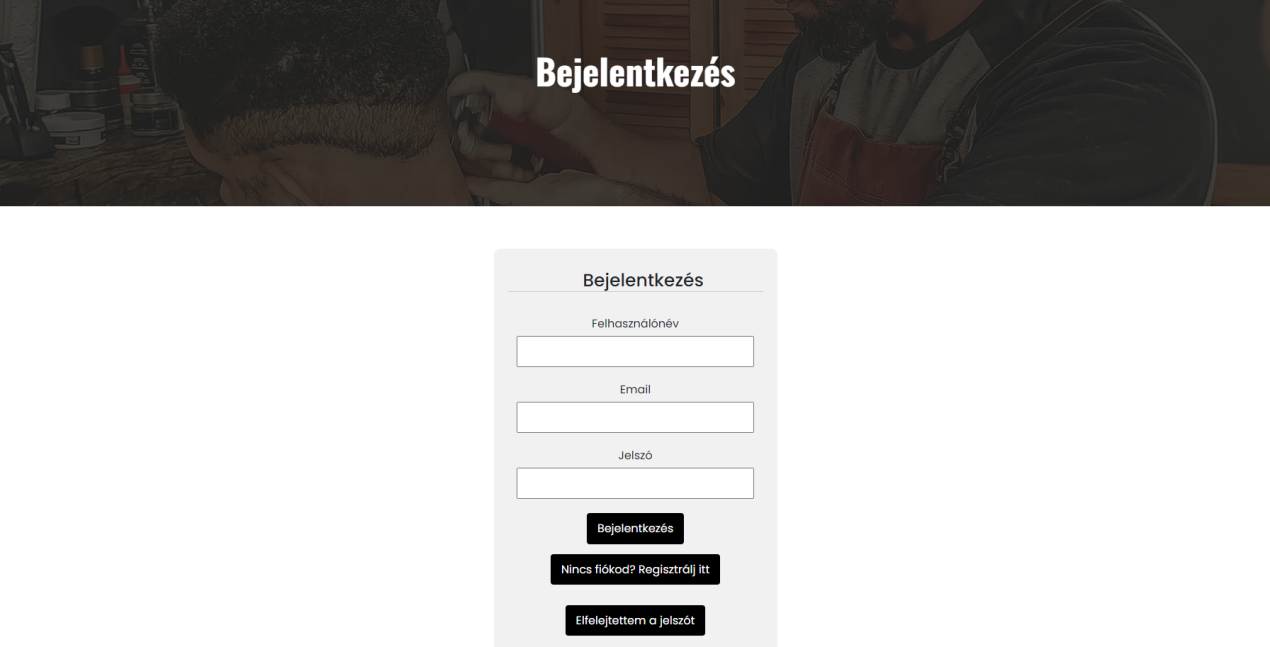
# AZ ADATBÁZIS STRUKTURÁJA



# A PROJEKT MŰKÖDÉSÉNEK LEÍRÁSA



2. ábra: Regisztráció



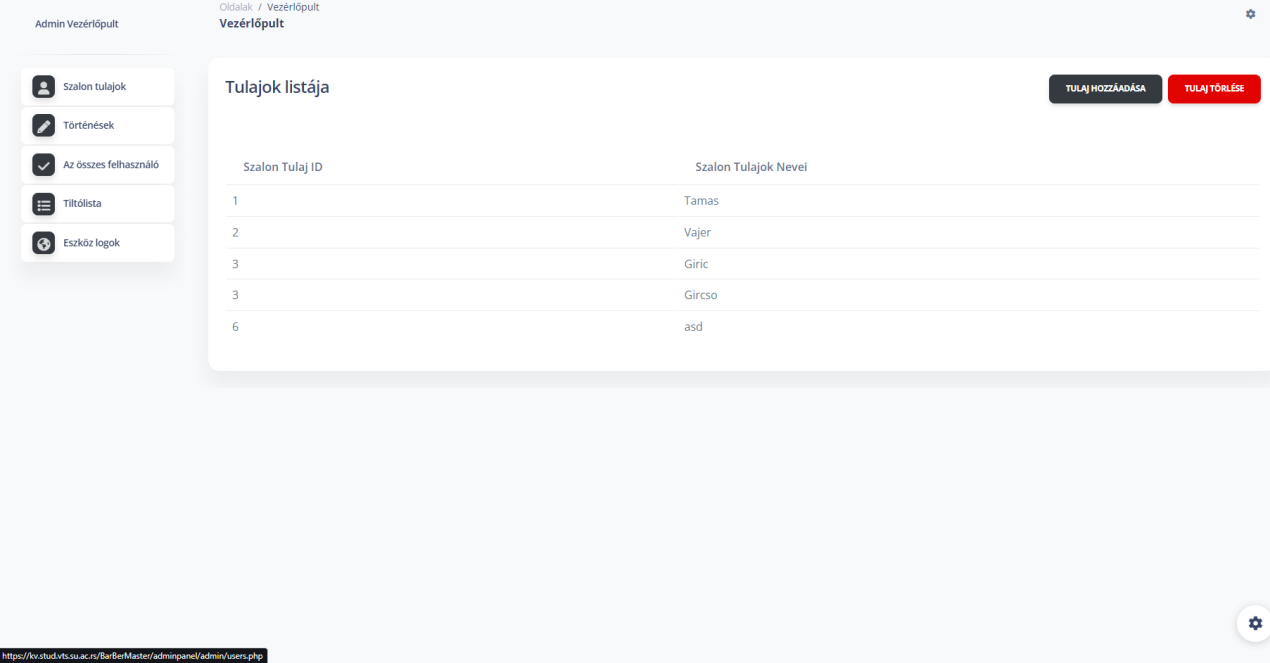
3. ábra Bejelentkezés



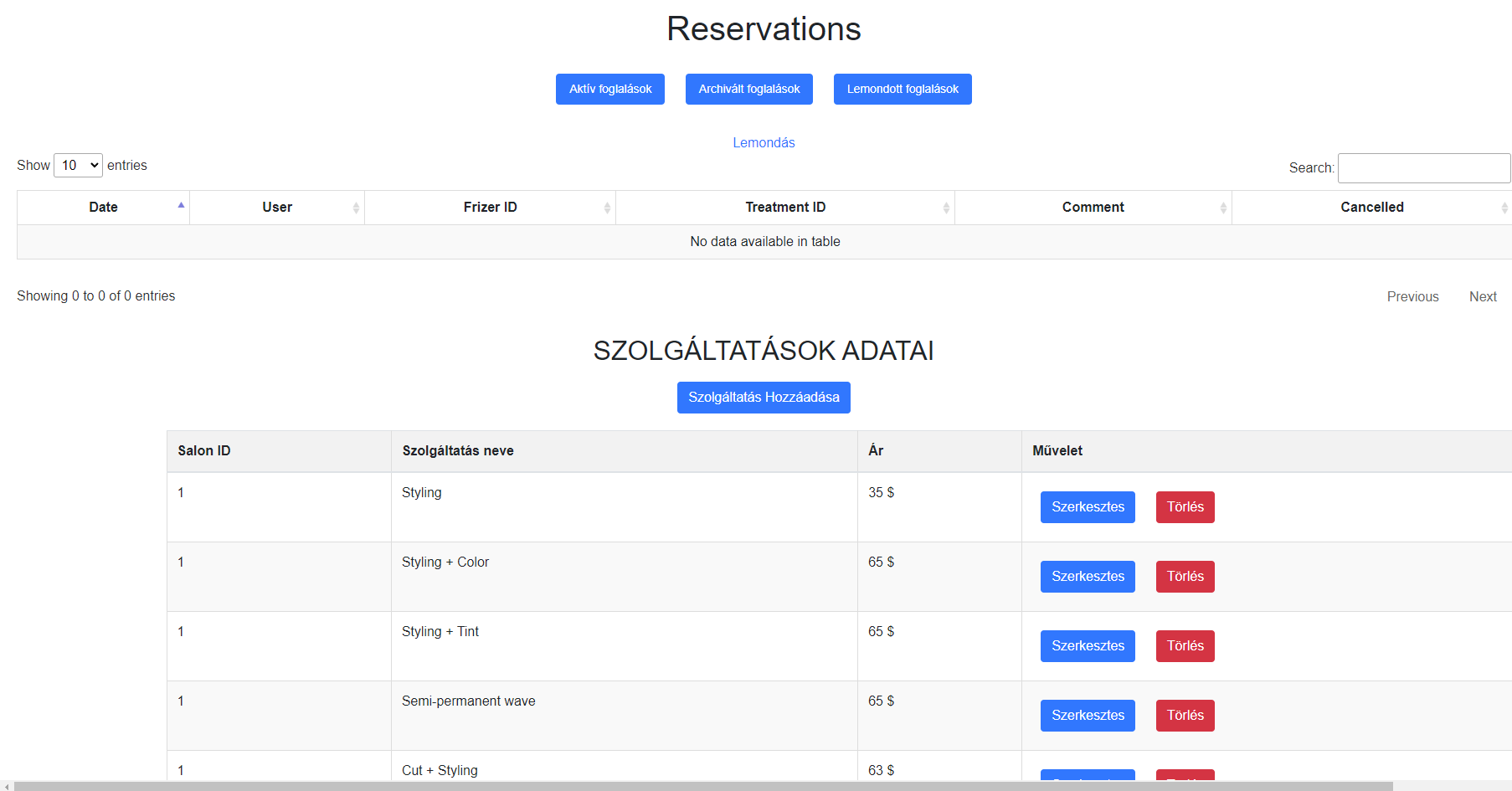
4. ábra: Foglalás



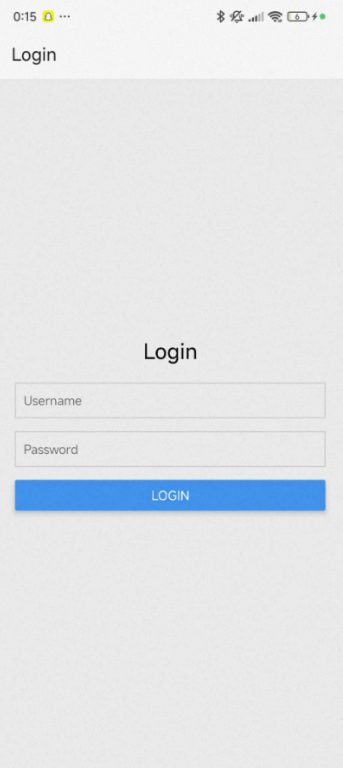
5. ábra: Itt láthatjuk az időpontunkat



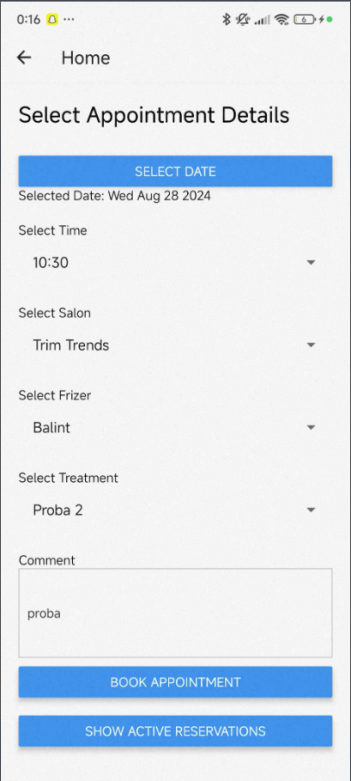
6. ábra: Admin felület



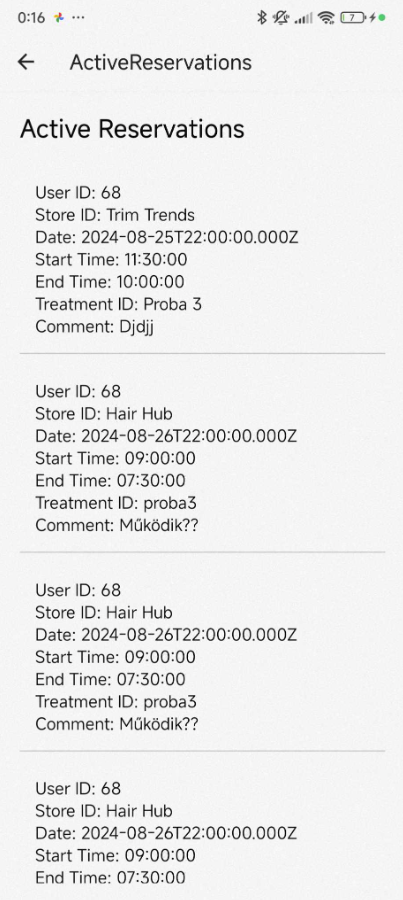
7. ábra: Szalon tulajdonos oldala



8. ábra: Ezen a felületen tudunk bejelentkezni



9. ábra: Ezen a felületen tudunk foglalni



10. ábra: Ezen a felületen megtekinthetjük az aktiv foglalásunkat

# FELHASZNÁLT IRODALOM

A felhasznált irodalomban a szerzők nevét ábécé sorrendben kell feltüntetni. Példa:

1. Matt Zandstra, “PHP 8 objekti, obrasci i praksa objektno orijentisan pristup”, Kompjuter biblioteka, Beograd, 2021
2. Robin Nixon, “Naučite PHP, MySQL i JavaScript: razvoj dinamičkih veb strana korak po korak prevod 6. izdanja”, Kompjuter biblioteka, Beograd, 2021

W1. <https://reactnative.dev/docs/tutorial>

W2. <https://phptherightway.com/>