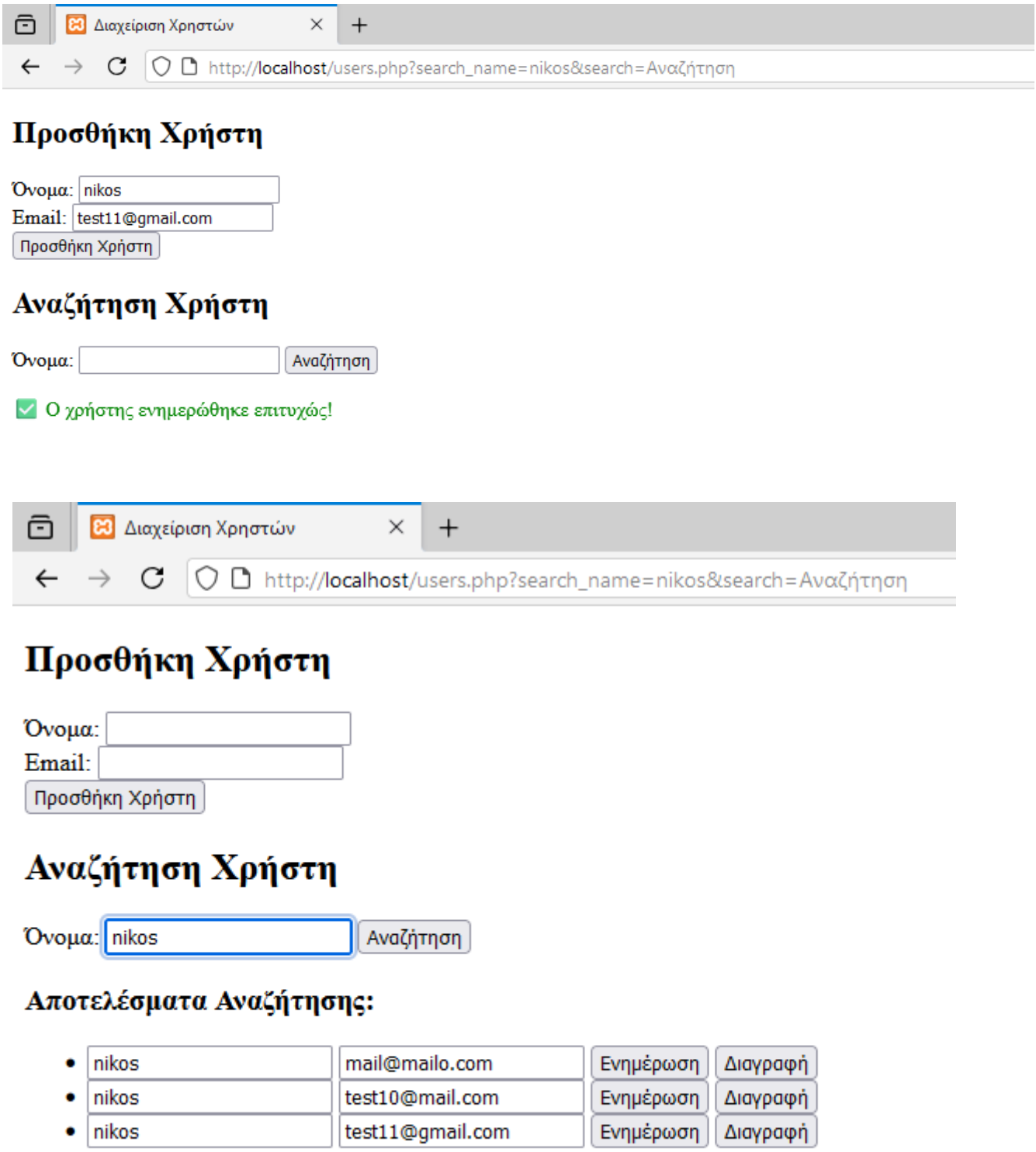
ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ ΣΕ PHP



Γενική περιγραφή

- Το πρόγραμμα επιτρέπει τη διαχείριση χρηστών με πεδία `name` και `email`.
- Υλοποιεί λειτουργίες: **Προσθήκη**, **Αναζήτηση**, **Ενημέρωση**, και **Διαγραφή**.
- Χρησιμοποιεί HTML φόρμες και PHP για να χειριστεί τα δεδομένα.
- Η βάση δεδομένων υποθέτουμε ότι έχει έναν πίνακα `users` με τουλάχιστον πεδία: `id`, `name`, `email`.

HTML Μέρος

1. Προσθήκη Χρήστη

```
<form action="" method="post">
   <label for="name">Όνομα:</label>
   <input type="text" name="name" required>
   <br>
   <label for="email">Email:</label>
   <input type="email" name="email" required>
   <br>
   <input type="submit" name="insert" value="Προσθήκη Χρήστη">
</form>
```

Φόρμα που στέλνει `POST` αίτημα με όνομα και email.

- Το πεδίο `name="insert"` στο κουμπί βοηθάει στο να καταλάβουμε ποια λειτουργία ζητάει ο χρήστης.

2. Αναζήτηση Χρήστη

```
<form action="" method="get">
   <label for="search_name">Όνομα:</label>
   <input type="text" name="search_name" required>
   <input type="submit" name="search" value="Αναζήτηση">
</form>
```

- Φόρμα που στέλνει `GET` αίτημα με το όνομα που θέλουμε να αναζητήσουμε.
- Η αναζήτηση γίνεται με μερικό ταίριασμα ονόματος.

PHP Μέρος

Σύνδεση με βάση δεδομένων

```
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "my_database";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Σφάλμα σύνδεσης: " . $conn->connect_error);
}
```

Δημιουργεί αντικείμενο σύνδεσης με MySQL βάση.

Αν η σύνδεση αποτύχει, σταματά το πρόγραμμα με μήνυμα σφάλματος.

1. Προσθήκη Χρήστη

```
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['insert'])) {
    $name = $_POST['name'];
    $email = $_POST['email'];

    $stmt = $conn->prepare("INSERT INTO users (name, email) VALUES (?, ?)");
    $stmt->bind_param("ss", $name, $email);

    if ($stmt->execute()) {
        echo "<p style='color: green;'>☑ Ο χρήστης προστέθηκε επιτυχώς!</p>";
    } else {
        echo "<p style='color: red;'>❌ Σφάλμα: " . $stmt->error . "</p>";
    }
    $stmt->close();
}
```

- Ελέγχει αν η φόρμα προσθήκης υποβλήθηκε.
- Παίρνει τα δεδομένα `name` και `email` από τη φόρμα.
- Χρησιμοποιεί προετοιμασμένη δήλωση (`prepare`) για ασφάλεια από SQL injection.

- Εισάγει νέο χρήστη στον πίνακα `users`.
- Εμφανίζει μήνυμα επιτυχίας ή σφάλματος.

---

2. Διαγραφή Χρήστη

```
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['delete'])) {
    $user_id = $_POST['user_id'];

    $stmt = $conn->prepare("DELETE FROM users WHERE id = ?");
    $stmt->bind_param("i", $user_id);

    if ($stmt->execute()) {
        echo "<p style='color: green;'>☑ Ο χρήστης διαγράφηκε επιτυχώς!</p>";
    } else {
        echo "<p style='color: red;'>❌ Σφάλμα: " . $stmt->error . "</p>";
    }
    $stmt->close();
}
```

Αν υποβληθεί φόρμα διαγραφής, παίρνει το `user_id`.

- Διαγράφει τον χρήστη από τη βάση.
- Εμφανίζει μήνυμα επιτυχίας ή σφάλματος.

3. Ενημέρωση Χρήστη

```
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['edit'])) {
    $user_id = $_POST['user_id'];
    $new_name = $_POST['new_name'];
    $new_email = $_POST['new_email'];

    $stmt = $conn->prepare("UPDATE users SET name = ?, email = ? WHERE id = ?");
    $stmt->bind_param("ssi", $new_name, $new_email, $user_id);




    if ($stmt->execute()) {
        echo "<p style='color: green;'>☑ Ο χρήστης ενημερώθηκε επιτυχώς!</p>";
    } else {
        echo "<p style='color: red;'>❌ Σφάλμα: " . $stmt->error . "</p>";
    }
    $stmt->close();
```

}

Αν υποβληθεί φόρμα ενημέρωσης, παίρνει το `user_id` και τα νέα στοιχεία.

- Ενημερώνει τα στοιχεία του χρήστη στη βάση.
- Εμφανίζει μήνυμα επιτυχίας ή σφάλματος.

---

4. Αναζήτηση Χρήστη

```
if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET['search'])) {
   $search_name = $_GET['search_name'];

   $stmt = $conn->prepare("SELECT id, name, email FROM users WHERE name LIKE ?");
   $search = "%" . $search_name . "%";
   $stmt->bind_param("s", $search);
   $stmt->execute();
   $result = $stmt->get_result();

   echo "<h3>Αποτελέσματα Αναζήτησης:</h3>";
   if ($result->num_rows > 0) {
      echo "<ul>";
      while ($row = $result->fetch_assoc()) {
         echo "<li>
               <form action='' method='post' style='display:inline;'>
                  <input type='hidden' name='user_id' value='" . $row['id'] . "'>
                  <input type='text' name='new_name' value='" . htmlspecialchars($row['name']) . "'
required>
                  <input type='email' name='new_email' value='" . htmlspecialchars($row['email']) . "'
required>
                  <input type='submit' name='edit' value='Ενημέρωση'>
                  <input type='submit' name='delete' value='Διαγραφή'>
               </form>
            </li>";
      }
      echo "</ul>";
   } else {
      echo "<p style='color: red;'>❌ Δεν βρέθηκαν χρήστες.</p>";
   }
   $stmt→close();
}
```

Λαμβάνει το όνομα που αναζητήθηκε.

- Χρησιμοποιεί `LIKE` με `%` για μερικό ταίριασμα.
- Εμφανίζει όλα τα αποτελέσματα σε λίστα.
- Κάθε αποτέλεσμα έχει τη δική του φόρμα για ενημέρωση ή διαγραφή

Τελικές ενέργειες

$conn->close();

Κλείνει τη σύνδεση με τη βάση δεδομένων.

Επιπλέον Σχόλια

πιπλέον Σχόλια

- **Ασφάλεια:** Χρησιμοποιεί `prepare` και `bind_param`, αποφεύγοντας το SQL injection.
- **Δομή:** Όλες οι ενέργειες γίνονται στην ίδια σελίδα, με βάση το είδος του αιτήματος (`GET` ή `POST`) και ποιο `submit` κουμπί πατήθηκε.
- **Χρήση HTML ειδικών χαρακτήρων:** Στην εμφάνιση ονομάτων/email στην αναζήτηση, χρησιμοποιείται `htmlspecialchars` για αποφυγή XSS.
- **Προβλήματα προς βελτίωση:**
    - Δεν υπάρχει validation πέραν του `required`.
    - Δεν γίνεται έλεγχος για μοναδικό email.
    - Μπορεί να βελτιωθεί το UI και να διαχωριστούν οι φόρμες σε διαφορετικές σελίδες για καλύτερη οργάνωση.
    - Θα ήταν καλό να υπάρχει έλεγχος για έγκυρη μορφή email από την πλευρά του server.
