**📄 create\_bookings\_table.php**

<?php

➡️ Ανοίγει την PHP – υποδεικνύει ότι ακολουθεί κώδικας PHP.

require\_once "db\_con.php";

➡️ Εισάγει το αρχείο db\_con.php **μία φορά**, για να χρησιμοποιηθεί η σύνδεση με τη βάση δεδομένων. Αν δεν το βρει, εμφανίζει σφάλμα και σταματά.

$sql = "CREATE TABLE IF NOT EXISTS bookings (

 id INT AUTO\_INCREMENT PRIMARY KEY,

 name VARCHAR(255) NOT NULL,

 email VARCHAR(255) NOT NULL,

 date DATE NOT NULL,

 time TIME NOT NULL,

 service VARCHAR(255) NOT NULL

)";

➡️ Δημιουργεί SQL εντολή:

* **CREATE TABLE IF NOT EXISTS**: Φτιάχνει τον πίνακα bookings μόνο αν **δεν υπάρχει ήδη**.
* id: Πρωτεύον κλειδί, αυτόματη αύξηση (auto-increment).
* name, email: Κείμενο μέχρι 255 χαρακτήρες, **υποχρεωτικά** (NOT NULL).
* date, time: Πληροφορίες ημερομηνίας/ώρας κράτησης.
* service: Το είδος της υπηρεσίας (π.χ. "θεωρητικό", "πρακτικό").

if (mysqli\_query($conn, $sql)) {

 echo "Table bookings created successfully";

} else {

 echo "Error creating table: " . mysqli\_error($conn);

}

➡️ Εκτελεί την SQL εντολή στη βάση δεδομένων χρησιμοποιώντας το αντικείμενο σύνδεσης $conn:

* Αν πετύχει ➡️ εμφανίζει μήνυμα επιτυχίας.
* Αν αποτύχει ➡️ εμφανίζει το σφάλμα.

mysqli\_close($conn);

➡️ Κλείνει τη σύνδεση με τη βάση – **καλή πρακτική** όταν τελειώνουμε.

?>

➡️ Κλείνει το μπλοκ PHP.

📌 **Σύνοψη**: Το create\_bookings\_table.php δημιουργεί τον πίνακα bookings με πεδία για όνομα, email, ημερομηνία, ώρα και τύπο υπηρεσίας.

**📄 create\_users\_table.php**

<?php

➡️ Έναρξη PHP.

require\_once "db\_con.php";

➡️ Περιλαμβάνει το αρχείο db\_con.php, για να χρησιμοποιηθεί η σύνδεση με τη βάση δεδομένων μέσω της μεταβλητής $conn.

$sql = "CREATE TABLE IF NOT EXISTS users (

 id INT AUTO\_INCREMENT PRIMARY KEY,

 username VARCHAR(50) NOT NULL,

 password VARCHAR(255) NOT NULL

)";

➡️ SQL εντολή για δημιουργία πίνακα users:

* id: Πρωτεύον κλειδί, με αυτόματη αύξηση.
* username: Μέχρι 50 χαρακτήρες, **υποχρεωτικό**.
* password: Μέχρι 255 χαρακτήρες, **υποχρεωτικό** — μεγάλη χωρητικότητα για αποθηκευμένους **κρυπτογραφημένους (hashed)** κωδικούς.

if (mysqli\_query($conn, $sql)) {

 echo "Table users created successfully";

} else {

 echo "Error creating table: " . mysqli\_error($conn);

}

➡️ Εκτελεί την SQL εντολή:

* Αν πετύχει ➡️ μήνυμα επιτυχίας.
* Αν αποτύχει ➡️ εμφανίζεται το σφάλμα από τη βάση.

mysqli\_close($conn);

➡️ Κλείνει η σύνδεση με τη βάση.

?>

➡️ Κλείσιμο PHP.

📌 **Σύνοψη**: Το create\_users\_table.php δημιουργεί έναν πίνακα για τους χρήστες, με πεδία username και password (πιθανώς για login διαχειριστή ή μαθητών).

**📄 db\_con.php**

<?php

➡️ Έναρξη PHP.

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "hotwheels";

➡️ Ορισμός παραμέτρων σύνδεσης με MySQL:

* localhost: Ο διακομιστής της βάσης (τρέχει τοπικά).
* root: Το όνομα χρήστη της MySQL (προεπιλεγμένο για τοπικές εγκαταστάσεις).
* "": Κενός κωδικός (όχι ασφαλές, αλλά συνηθισμένο για δοκιμές).
* hotwheels: Το όνομα της βάσης δεδομένων.

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

➡️ Δημιουργεί τη σύνδεση με τη MySQL χρησιμοποιώντας τη συνάρτηση mysqli\_connect().

* Το αποτέλεσμα αποθηκεύεται στη μεταβλητή $conn.

if (!$conn) {

 die("Connection failed: " . mysqli\_connect\_error());

}

➡️ Αν η σύνδεση **αποτύχει**, η mysqli\_connect\_error() εμφανίζει το σφάλμα και η die() τερματίζει το script.

📌 **Σύνοψη**: Το db\_con.php είναι το βασικό αρχείο σύνδεσης με τη MySQL. Εισάγεται σε κάθε script που χρειάζεται επικοινωνία με τη βάση.

**📄 delete.php**

<?php

➡️ Έναρξη PHP.

 // Σύνδεση με βάση δεδομένων

 $conn = mysqli\_connect('localhost', 'zoiokara', 'W8Z32Ppg');

➡️ Σύνδεση με τον τοπικό MySQL server χρησιμοποιώντας:

* **χρήστη**: zoiokara
* **κωδικό**: W8Z32Ppg

 mysqli\_select\_db($conn, 'zoiokara');

➡️ Επιλογή της βάσης δεδομένων με όνομα zoiokara.

 // Λήψη τιμής και διαγραφή

 $id = $\_POST['id2delete'];

➡️ Λαμβάνει την τιμή του id2delete από **φόρμα POST**. Δηλαδή περιμένει να σταλεί ID από form με method="post".

 mysqli\_query($conn, "DELETE FROM testclients WHERE id='$id'");

➡️ Εκτελεί SQL εντολή που διαγράφει εγγραφή από τον πίνακα testclients με το δοσμένο id.

⚠️ **ΠΡΟΒΛΗΜΑ ΑΣΦΑΛΕΙΑΣ**: Το $id μπαίνει απευθείας στο query → επιρρεπές σε SQL Injection. Πρέπει να χρησιμοποιηθεί prepared statement ή τουλάχιστον mysqli\_real\_escape\_string().

 mysqli\_close();

?>

➡️ Κλείνει η σύνδεση με τη βάση (χωρίς να δίνει μεταβλητή — δεν θα λειτουργήσει σωστά!).

**📄 edit.php**

<?php

➡️ Έναρξη PHP.

 // Σύνδεση με βάση δεδομένων

 $conn = mysqli\_connect('localhost', 'zoiokara', 'W8Z32Ppg');

 mysqli\_select\_db($conn, 'zoiokara');

➡️ Δημιουργεί σύνδεση με τη MySQL βάση zoiokara χρησιμοποιώντας τον χρήστη zoiokara και κωδικό W8Z32Ppg.

 // Ασφαλής λήψη των δεδομένων από POST

 $id = mysqli\_real\_escape\_string($conn, $\_POST['client\_selection']);

 $name = mysqli\_real\_escape\_string($conn, $\_POST['my\_name']);

 $address = mysqli\_real\_escape\_string($conn, $\_POST['my\_address']);

 $service = mysqli\_real\_escape\_string($conn, $\_POST['my\_service']);

 $date = mysqli\_real\_escape\_string($conn, $\_POST['my\_date']);

 $time = mysqli\_real\_escape\_string($conn, $\_POST['my\_time']);

➡️ Λαμβάνει τα δεδομένα από POST (π.χ. από φόρμα ή AJAX) και τα **καθαρίζει με mysqli\_real\_escape\_string()** ώστε να αποφευχθούν επιθέσεις SQL Injection.

Μεταβλητές:

* client\_selection: το ID του πελάτη που θα ενημερωθεί.
* my\_name, my\_address, my\_service, my\_date, my\_time: τα νέα δεδομένα του πελάτη.

 // Ενημέρωση όλων των πεδίων με ένα query

 $query = "UPDATE testclients

 SET name='$name', address='$address', service='$service', date='$date', time='$time'

 WHERE id='$id'";

 mysqli\_query($conn, $query);

➡️ Εκτελεί SQL UPDATE που αλλάζει τα πεδία της εγγραφής στον πίνακα testclients για το συγκεκριμένο id.

 // Ανάκτηση των νέων δεδομένων για επιστροφή στο frontend

 $result = mysqli\_query($conn, "SELECT id, name, address, service, date, time FROM testclients WHERE id = '$id'");

 $row = mysqli\_fetch\_array($result);

➡️ Εκτελεί ένα SELECT ώστε να πάρει τα ενημερωμένα δεδομένα από τη βάση, και τα αποθηκεύει στον πίνακα $row.

 // Επιστροφή δεδομένων σε μορφή JSON

 echo json\_encode($row);

➡️ Μετατρέπει τα δεδομένα σε **JSON μορφή** και τα στέλνει πίσω στο frontend. Χρήσιμο αν χρησιμοποιείται AJAX για ενημέρωση της σελίδας χωρίς refresh.

 mysqli\_close($conn);

?>

➡️ Κλείνει τη σύνδεση με τη βάση και τελειώνει η PHP.

**✅ Συνοψίζοντας:**

* Το edit.php λαμβάνει δεδομένα μέσω POST.
* Ενημερώνει μια εγγραφή στον πίνακα testclients.
* Επιστρέφει τα νέα δεδομένα σε JSON, για χρήση από το frontend (π.χ. με AJAX).
* Είναι **σχετικά ασφαλές** (χρήση mysqli\_real\_escape\_string) αλλά όχι ιδανικά — **θα ήταν καλύτερο με prepared statements**.

**📄 funcs.php**

<?php

function login\_page($error\_msg, $target\_page) {

➡️ Έναρξη της συνάρτησης login\_page, η οποία:

* Δέχεται ένα μήνυμα σφάλματος ($error\_msg).
* Και μια σελίδα στόχο ($target\_page) στην οποία θα σταλεί το POST.

 echo '<form method="post" action="'.$target\_page.'">';

➡️ Εκτυπώνει την αρχή της φόρμας:

* method="post": τα δεδομένα θα σταλούν με POST.
* action="$target\_page": ορίζει σε ποια σελίδα θα σταλούν (π.χ. login\_a.php).

?>

 <table width="100%" height="100%">

 <tr>

 <td>

 <table align="center">

 <tr>

 <td width="110">Όνομα Χρήστη:</td>

 <td align="right"><input type="text" name="uname"></td>

 </tr>

 <tr>

 <td>Κωδικός:</td>

 <td align="right"><input type="password" name="pass"></td>

 </tr>

 <tr>

 <td colspan="2" align="center"><input type="submit" value="Είσοδος"></td>

 </tr>

 </table>

<?php

➡️ HTML κώδικας της φόρμας:

* Ζητά **όνομα χρήστη** και **κωδικό**.
* Χρησιμοποιεί input πεδία με name="uname" και name="pass".
* Το κουμπί υποβολής είναι Είσοδος.

 echo '<p align="center">'.$error\_msg.'</p>';

➡️ Αν υπάρχει κάποιο μήνυμα σφάλματος (π.χ. "Λάθος στοιχεία"), το εμφανίζει στο κέντρο της σελίδας.

?>

 </td>

 </tr>

 </table>

</form>

<?php

}

?>

➡️ Κλείνει τα HTML tags και την PHP συνάρτηση.

**✅ Σύνοψη:**

Η login\_page():

* Εμφανίζει μια φόρμα σύνδεσης με όνομα χρήστη και κωδικό.
* Στέλνει τα δεδομένα στη σελίδα $target\_page.
* Αν υπάρχει σφάλμα, το εμφανίζει κάτω από τη φόρμα.

✍️ **Καλή πρακτική** για επαναχρησιμοποιήσιμη φόρμα σύνδεσης.

**📄 hello.php**

<html>

<head>

<META charset=”UTF8">

<title>Hello world</title>

</head>

➡️ HTML **επικεφαλίδα**:

* Ορίζει τον τίτλο της σελίδας ως "Hello world".
* Περιλαμβάνει meta πληροφορία για χαρακτήρες.

<body>

<H1>Η πρώτη μου σελίδα σε PHP</H1>

<p>

<?php

echo "Hello world!";

?>

</p>

</body>

</html>

➡️ Το κυρίως σώμα:

* Εμφανίζει τίτλο σε H1: **"Η πρώτη μου σελίδα σε PHP"**
* Μέσα στο <p> χρησιμοποιεί PHP για να τυπώσει: **"Hello world!"**

<?php

echo "Hello world!";

?>

➡️ Βασική PHP εντολή: εμφανίζει τη φράση Hello world! στην οθόνη.

**✅ Σύνοψη:**

* Το hello.php είναι ένα **δοκιμαστικό αρχείο**.
* Περιέχει μικρή HTML σελίδα που ενσωματώνει PHP.
* Είναι το **"Hello World"** του PHP για εκπαίδευση και έλεγχο ότι ο server τρέχει σωστά.

**📄 insert.php**

<?php

➡️ Έναρξη PHP.

 // Σύνδεση με βάση δεδομένων

 $conn = mysqli\_connect('localhost', 'zoiokara', 'W8Z32Ppg');

 mysqli\_select\_db($conn, 'zoiokara');

➡️ Δημιουργεί σύνδεση με τη βάση δεδομένων zoiokara με χρήση των στοιχείων zoiokara / W8Z32Ppg.

 // Ασφαλής λήψη των δεδομένων από POST

 $name = mysqli\_real\_escape\_string($conn, $\_POST['my\_name']);

 $address = mysqli\_real\_escape\_string($conn, $\_POST['my\_address']);

 $service = mysqli\_real\_escape\_string($conn, $\_POST['my\_service']);

 $date = mysqli\_real\_escape\_string($conn, $\_POST['my\_date']);

 $time = mysqli\_real\_escape\_string($conn, $\_POST['my\_time']);

➡️ Λαμβάνει τα δεδομένα από POST και τα **καθαρίζει** για αποφυγή SQL injection.

 // Εκτέλεση ερωτήματος εισαγωγής

 $query = "INSERT INTO testclients (name, address, service, date, time)

 VALUES ('$name', '$address', '$service', '$date', '$time')";

 mysqli\_query($conn, $query);

➡️ Εκτελεί SQL INSERT για να δημιουργήσει νέα εγγραφή στον πίνακα testclients.

 // Λήψη του ID του τελευταίου πελάτη

 $inserted\_id = mysqli\_insert\_id($conn);

➡️ Παίρνει το id της εγγραφής που μόλις εισήχθη, για να το χρησιμοποιήσει αμέσως μετά.

 // Ανάκτηση όλων των δεδομένων του πελάτη για επιστροφή στο frontend

 $result = mysqli\_query($conn, "SELECT id, name, address, service, date, time FROM testclients WHERE id = '$inserted\_id'");

 $row = mysqli\_fetch\_array($result);

➡️ Εκτελεί SELECT για να φέρει όλα τα πεδία της νέας εγγραφής, τα οποία αποθηκεύονται στον πίνακα $row.

 // Επιστροφή δεδομένων σε μορφή JSON

 echo json\_encode($row);

➡️ Τα δεδομένα στέλνονται πίσω στο frontend σε μορφή **JSON** — χρήσιμο για AJAX.

 mysqli\_close($conn);

?>

➡️ Κλείνει τη σύνδεση με τη βάση και τη PHP.

**✅ Συνοψίζοντας:**

Το insert.php:

* Δέχεται νέα δεδομένα πελάτη μέσω POST.
* Εισάγει την εγγραφή στον πίνακα testclients.
* Στέλνει πίσω τα ίδια δεδομένα σε JSON για χρήση στο frontend (π.χ. για ενημέρωση πίνακα χωρίς refresh).

**login.php**

**🔸 PHP ΤΜΗΜΑ (επάνω μέρος αρχείου)**

<?php

session\_start();

include "db\_con.php";

$error = "";

➡️ Ξεκινάει το session για να μπορέσει να αποθηκεύσει login στοιχεία.
➡️ Εισάγει τη σύνδεση με τη βάση από db\_con.php.
➡️ Ορίζει μεταβλητή $error (θα χρησιμοποιηθεί αν τα στοιχεία είναι λάθος).

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

 $uname = $\_POST["uname"];

 $pass = $\_POST["pass"];

➡️ Αν η φόρμα υποβλήθηκε (μέθοδος POST), αποθηκεύει τα στοιχεία που πληκτρολόγησε ο χρήστης.

 $query = "SELECT \* FROM users WHERE uname='$uname' AND pass='$pass'";

 $result = mysqli\_query($con, $query);

 $user = mysqli\_fetch\_array($result);

➡️ Ετοιμάζει SQL εντολή για έλεγχο στοιχείων από τον πίνακα users.

⚠️ **ΠΡΟΣΟΧΗ**:

* Ελέγχει τον κωδικό **χωρίς κρυπτογράφηση**.
* Εισάγει τα δεδομένα απευθείας στο SQL ➤ υπάρχει **κίνδυνος SQL Injection**.

 if ($user) {

 $\_SESSION["uname"] = $user["uname"];

 header("Location: management.php");

 exit;

 } else {

 $error = "❌🚗 Λάθος όνομα ή κωδικός.";

 }

}

?>

➡️ Αν βρεθεί χρήστης, αποθηκεύει το όνομα στο SESSION και κάνει redirect στον πίνακα διαχείρισης (management.php).

Αλλιώς, εμφανίζει μήνυμα λάθους.

**🔸 HTML ΤΜΗΜΑ (φόρμα σύνδεσης)**

* Χρησιμοποιεί **Bootstrap 5** για responsive σχεδίαση.
* Έχει form που στέλνει τα στοιχεία με POST πίσω στο login.php.
* Περιλαμβάνει input πεδία για:
	+ uname → Όνομα χρήστη
	+ pass → Κωδικός
* Στυλ με login-card, btn, form-control, hover effects κ.λπ.
* Κουμπί "Επιστροφή στην Αρχική"
* Όλα τα πεδία είναι required

**✅ Συνοψίζοντας:**

Το login.php:

* Παρουσιάζει τη φόρμα σύνδεσης διαχειριστή.
* Ελέγχει τα στοιχεία με SQL από τον πίνακα users.
* Αν είναι σωστά, δημιουργεί session και μεταφέρει τον χρήστη στο management.php.

📌 **Βελτιώσεις που προτείνω**:

1. Κρυπτογράφηση κωδικού με password\_hash() και password\_verify().
2. Χρήση **prepared statements** για αποφυγή SQL injection.
3. Εμφάνιση του $error στο HTML όταν υπάρχει.

**📄 login\_a.php – ΤΜΗΜΑ HTML/CSS (στο <head>)**

html

 <meta charset="UTF-8">

 <title>Καλωσόρισμα</title>

 <style>

 body { background-color: #f75d59; ... }

 .container { ... }

 .card { ... }

 input[type="submit"] { ... }

 </style>

</head>

➡️ Ορίζει τη σελίδα με ελληνική κωδικοποίηση και στυλ για μια κάρτα-μήνυμα καλωσορίσματος:

* Κεντραρισμένο περιεχόμενο
* Σκίαση
* Κουμπί "Αποσύνδεση"

**📄 PHP ΤΜΗΜΑ (στο <body>)**

<?php

include "db\_con.php";

include "funcs.php";

➡️ Εισάγει:

* db\_con.php: για σύνδεση με βάση
* funcs.php: για χρήση της login\_page() σε περίπτωση σφάλματος

if (isset($\_POST["uname"])) {

 $uname = $\_POST["uname"];

 $pass = $\_POST["pass"];

➡️ Αν έχουν σταλεί στοιχεία σύνδεσης (μέσω POST), τα αποθηκεύει σε μεταβλητές.

 $query = 'SELECT \* FROM users WHERE uname="' . $uname . '" AND pass="' . $pass . '"';

 $result = mysqli\_query($con, $query);

 $x = mysqli\_fetch\_array($result);

➡️ Δημιουργεί SQL εντολή και αναζητά τον χρήστη στη βάση.

* **⚠️ Προσοχή:** Δεν υπάρχει καμία προστασία! Δεν χρησιμοποιείται ούτε mysqli\_real\_escape\_string, ούτε prepared statements.

 if ($x != NULL) {

 echo '<div class="container">';

 echo '<div class="card">';

 echo '<h2> Καλώς ήρθες 🚗 ' . htmlspecialchars($x['uname']) . '!</h2>';

➡️ Αν ο χρήστης βρέθηκε, εμφανίζεται μήνυμα καλωσορίσματος με το όνομά του.

 echo '<ul>';

 echo '<li><strong>uname:</strong> ' . htmlspecialchars($x['uname']) . '</li>';

 echo '<li><strong>pass:</strong> ' . htmlspecialchars($x['pass']) . '</li>';

 echo '</ul>';

➡️ Εμφανίζει τα στοιχεία του χρήστη: όνομα και κωδικό.
⚠️ **ΜΕΓΑΛΗ ΠΡΟΕΙΔΟΠΟΙΗΣΗ**: Δεν πρέπει ποτέ να εμφανίζεται ο κωδικός του χρήστη (ούτε "hashed").

 echo '<form action="index.php" method="post">';

 echo '<input type="submit" value="Αποσύνδεση">';

 echo '</form>';

➡️ Φόρμα αποσύνδεσης με κουμπί που επιστρέφει στην αρχική.

 } else {

 login\_page("😔 Λάθος <i>Κωδικός</i>", "login\_a.php");

 }

} else {

 login\_page("", "login\_a.php");

}

?>

➡️ Αν τα στοιχεία είναι λάθος ή δεν έχουν σταλεί καθόλου:

* Καλεί τη login\_page() από funcs.php για εμφάνιση της φόρμας.

**✅ Συνοψίζοντας:**

Το login\_a.php:

* Ελέγχει αν υπάρχουν σωστά στοιχεία σύνδεσης.
* Αν είναι σωστά ➜ εμφανίζει "κάρτα καλωσορίσματος".
* Αν είναι λάθος ➜ ξαναεμφανίζει τη φόρμα.
* Εμφανίζει **username και password**, κάτι που **πρέπει να αποφεύγεται για λόγους ασφαλείας**.

**📄 logout.php**

<?php

session\_start();

➡️ Ξεκινάει το session ώστε να μπορεί να το καταστρέψει. Αν δεν καλέσεις session\_start(), δεν μπορείς να χειριστείς session μεταβλητές.

session\_destroy();

➡️ Καταστρέφει εντελώς το session του χρήστη, δηλαδή:

* Ο χρήστης αποσυνδέεται.
* Διαγράφονται όλα τα $\_SESSION δεδομένα (π.χ. όνομα χρήστη, status, κ.ά.).

header("Location: index.php");

exit;

➡️ Μεταφέρει τον χρήστη πίσω στην αρχική σελίδα (index.php) και σταματά την εκτέλεση του script.

**✅ Συνοψίζοντας:**

Το logout.php:

* Τερματίζει το session.
* Κάνει redirect στην αρχική.
* Είναι η **τυπική μορφή αρχείου αποσύνδεσης** σε PHP.

Το αρχείο **management.php** είναι η βασική διαχειριστική σελίδα της εφαρμογής. Συνδυάζει:

* **PHP (backend)** για προστασία και ανάγνωση δεδομένων.
* **JavaScript + AJAX** για διαχείριση πελατών σε πραγματικό χρόνο.
* **HTML** για τη δημιουργία πίνακα πελατών και φορμών.

Ας το δούμε **αναλυτικά και οργανωμένα**:

**🔹 1. Έλεγχος πρόσβασης με PHP**

session\_start();

include "db\_con.php";

if (!isset($\_SESSION["uname"])) {

 header("Location: login.php");

 exit;

}

➡️ Αν δεν υπάρχει ενεργό session (uname), γίνεται ανακατεύθυνση στο login.php → ✅ **προστασία πρόσβασης**.

**🔹 2. Ανάκτηση κρατήσεων**

$result = mysqli\_query($con, "SELECT \* FROM bookings");

➡️ Φαίνεται ότι γίνεται ανάκτηση από πίνακα bookings, αλλά **πιο κάτω ο πίνακας είναι testclients** – πιθανόν έχει μείνει από παλιότερη έκδοση.

**🔹 3. JavaScript + AJAX**

Υπάρχουν **τρεις συναρτήσεις AJAX**:

**✅ ajaxFunctionEdit()**

* Παίρνει τα δεδομένα της φόρμας.
* Βρίσκει το επιλεγμένο πελάτη με radio button.
* Στέλνει POST στο edit.php.
* Όταν έρθει απάντηση, **ενημερώνει τον πίνακα** χωρίς refresh.

**✅ ajaxFunctionInsert()**

* Παίρνει δεδομένα από τη φόρμα.
* Στέλνει POST στο insert.php.
* Όταν έρθει απάντηση, **προσθέτει νέα γραμμή** στον πίνακα.

**✅ ajaxFunctionDelete(id)**

* Στέλνει POST στο delete.php με το id του πελάτη.
* Αν η διαγραφή πετύχει, **αφαιρεί τη γραμμή από τον πίνακα DOM**.

**🔹 4. Πίνακας πελατών (HTML + PHP)**

$conn = mysqli\_connect(...);

$y = mysqli\_query($conn, "SELECT \* FROM testclients");

while ($x = mysqli\_fetch\_array($y)) {

➡️ Φορτώνει πελάτες από τον πίνακα testclients.

Κάθε γραμμή περιλαμβάνει:

* Επιλογή (radio)
* Όνομα, διεύθυνση, υπηρεσία, ημερομηνία, ώρα
* Εικονίδιο διαγραφής (del.png) με onclick="ajaxFunctionDelete(id)"

**🔹 5. Φόρμα εισαγωγής/ενημέρωσης**

Περιλαμβάνει πεδία:

* Όνομα
* Διεύθυνση
* Υπηρεσία
* Ημερομηνία
* Ώρα

Κάτω από τα πεδία υπάρχουν:

* Εικόνα "Επεξεργασία" που καλεί ajaxFunctionEdit()
* Εικόνα "Προσθήκη" που καλεί ajaxFunctionInsert()

**🔹 6. Λειτουργία χωρίς refresh**

Χάρη στα AJAX scripts:

* Μπορείς να προσθέσεις/επεξεργαστείς/διαγράψεις εγγραφές **χωρίς επαναφόρτωση** της σελίδας.
* Το αποτέλεσμα εμφανίζεται άμεσα στον πίνακα.

**✅ Τι να προσέξεις:**

1. ✅ **Πολύ καλή διαχείριση client-side με AJAX.**
2. ❗ **SQL injection**: το edit.php, insert.php, delete.php πρέπει να χρησιμοποιούν **prepared statements**.
3. ❗ Χρήση id μέσα σε DOM/JavaScript χρειάζεται προσοχή (π.χ. σε περίπτωση ειδικών χαρακτήρων).
4. ✅ Αν χρησιμοποιούσες fetch() αντί για XMLHttpRequest, ο κώδικας θα ήταν πιο απλός.

Αναλυτική εξήγηση του αρχείου **setup\_testclients.php**, το οποίο χρησιμοποιείται για **δημιουργία και αρχικοποίηση του πίνακα testclients** με δείγμα εγγραφών.

**🔹 Τι κάνει το αρχείο:**

1. Δημιουργεί σύνδεση με τη βάση.
2. Διαγράφει τον πίνακα testclients αν υπάρχει.
3. Τον ξαναδημιουργεί με νέο schema.
4. Εισάγει 4 δείγματα εγγραφών.
5. Εμφανίζει:
	* Όλους τους πίνακες της βάσης.
	* Τη δομή του testclients.
	* Όλες τις εγγραφές του.

**🔍 Γραμμή-γραμμή:**

**🧩 Σύνδεση με βάση:**

$conn = mysqli\_connect('localhost', 'zoiokara', 'W8Z32Ppg');

mysqli\_select\_db($conn, 'zoiokara');

➡️ Σύνδεση με τη βάση zoiokara.

**🧨 Διαγραφή πίνακα (αν υπάρχει):**

mysqli\_query($conn, "DROP TABLE IF EXISTS testclients");

➡️ Αν υπάρχει ήδη ο πίνακας, διαγράφεται για να δημιουργηθεί καθαρός από την αρχή.

**🛠️ Δημιουργία πίνακα:**

mysqli\_query($conn, "

CREATE TABLE testclients (

 id INT NOT NULL AUTO\_INCREMENT,

 name VARCHAR(40) NOT NULL,

 address VARCHAR(100),

 service VARCHAR(100),

 date DATE,

 time TIME,

 PRIMARY KEY(id)

) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci

");

➡️ Δημιουργεί τον πίνακα με στήλες:

* id: αυτόματη αύξηση (πρωτεύον κλειδί)
* name, address, service: πληροφορίες πελάτη
* date, time: πότε έχει το μάθημα

**📝 Εισαγωγή εγγραφών:**

INSERT INTO testclients (name, address, service, date, time)

VALUES

('Αλεξάνδρα', 'Δαρδανελίων 24', 'Θεωριτικό Μάθημα 4', '2025-05-29', '10:00:00'),

('Αντώνης', 'Διγενή Ακρίτα 113', 'Τελευταίο Πρακτικό Μάθημα', '2025-05-30', '17:00:00'),

('Δήμητρα', 'Δαρδανελίων 32', 'Πρώτο Πρακτικό Μάθημα', '2025-06-13', '12:00:00'),

('Κωνσταντίνος', 'Δέγλερη 21', 'Θεωριτικό Μάθημα 2', '2025-06-23', '14:30:00')

➡️ Εισάγει 4 εγγραφές δοκιμής με όνομα, διεύθυνση, τύπο μαθήματος, ημερομηνία και ώρα.

**📋 Εμφάνιση πινάκων:**

$y = mysqli\_query($conn, "SHOW TABLES");

while ($x = mysqli\_fetch\_row($y)) {

 echo $x[0] . '<br>';

}

➡️ Εμφανίζει όλους τους πίνακες που υπάρχουν στη βάση (μετά τη δημιουργία).

**🧱 Εμφάνιση δομής πίνακα:**

$y = mysqli\_query($conn, "DESCRIBE testclients");

while ($x = mysqli\_fetch\_row($y)) {

 echo $x[0] . ' ' . $x[1] . ' ' . $x[2] . ' ' . $x[3] . '<br>';

}

➡️ Εμφανίζει τις στήλες του πίνακα testclients: όνομα, τύπος δεδομένων, αν δέχεται NULL κ.λπ.

**📦 Εμφάνιση περιεχομένων πίνακα:**

$y = mysqli\_query($conn, "SELECT \* FROM testclients");

while ($x = mysqli\_fetch\_array($y)) {

 echo $x['id'] . ' ' . $x['name'] . ' ' . $x['address'] . ' ' . $x['service'] . ' ' . $x['date'] . ' ' . $x['time'] . '<br>';

}

➡️ Τυπώνει τις εγγραφές που εισήχθησαν στον πίνακα testclients.

**✅ Τελευταίο βήμα:**

mysqli\_close();

➡️ Κλείνει τη σύνδεση με τη βάση δεδομένων.

**✨ Τελικό αποτέλεσμα:**

Εκτελώντας το αρχείο αυτό:

* Δημιουργείται πλήρως λειτουργικός πίνακας testclients.
* Εισάγονται 4 παραδείγματα μαθητών.
* Στην οθόνη εμφανίζονται:
	+ Όλοι οι πίνακες της βάσης
	+ Η δομή του testclients
	+ Οι εγγραφές του