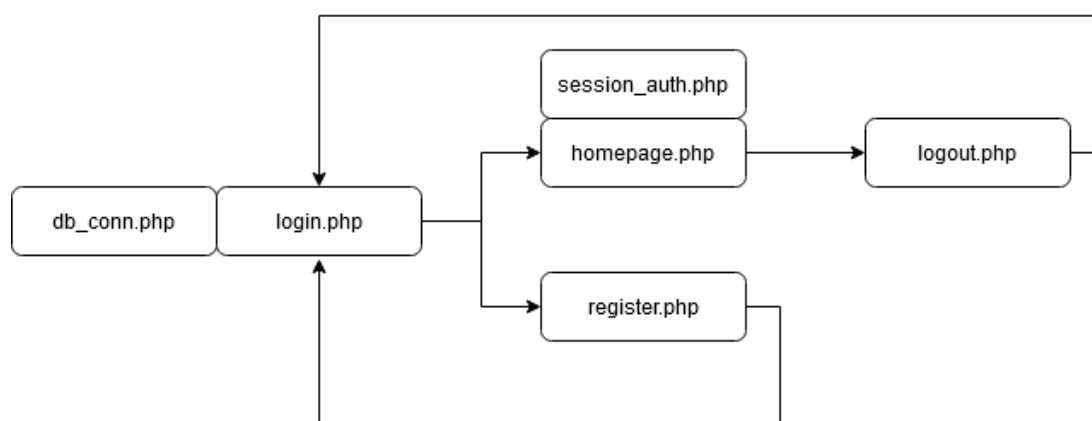


# Δομή και λειτουργία σελίδας

Επιδιώκοντας να δημιουργήσουμε μια σελίδα πραγματικών συνθηκών, δημιουργήσαμε και όλες τις άλλες απαραίτητες ιστοσελίδες που συνοδεύουν ένα Login Form. Το μοντέλο μας δηλαδή περιλαμβάνει μια σελίδα Register που γίνεται εγγραφή νέων χρηστών καθώς και ένα εικονικό Homepage που εμφανίζεται έπειτα από επιτυχημένη σύνδεση. Αναλυτικά το προτεινόμενο μοντέλο περιγράφεται στο διάγραμμα:



Συγκεκριμένα:

- Για την σωστή λειτουργία του μοντέλου, τα αρχεία (login.php και register.php) απαιτούν το **db\_conn.php**. Πρόκειται για ένα απλό αρχείο που εδραιώνει την σωστή σύνδεση με την βάση δεδομένων MySQL. Σε περίπτωση μη επιτυχημένης σύνδεσης εμφανίζεται μήνυμα στην οθόνη.
- Η κεντρική σελίδα είναι η **login.php** από την οποία ο χρήστης μπορεί είτε να επιδιώξει την είσοδο στο 'σύστημα' είτε να δημιουργήσει έναν νέο χρήστη στην βάση δεδομένων μέσω της **register.php**.
- Ύστερα από επιτυχημένο Login, ο χρήστης παραπέμπεται στην σελίδα **homepage.php**, από την οποία η μόνη επιλογή που έχει είναι το Logout. Αποσυνδέεται δηλαδή, μια εργασία που αναλαμβάνει η **logout.php**, διαλύοντας το Session που έχει δημιουργηθεί και ανακατευθύνοντας τον χρήστη στην αρχική Login σελίδα. Σε περίπτωση που επιχειρήσει να μεταβεί χειροκίνητα στην προηγούμενη σελίδα μέσω του browser, την **login.php**, τότε ανακατευθύνεται πάλι στην **homepage.php** αφού δεν έχει αποσυνδεθεί.
- Η σελίδα **homepage.php** απαιτεί το αρχείο **session\_auth.php**, ένα απλό αρχείο που ελέγχει εάν το session και η σύνδεση έχουν γίνει σωστά. Σε περίπτωση που αυτό δεν ισχύει, ανακατευθύνει τον χρήστη πίσω στην αρχική σελίδα Login.

Η επικοινωνία μεταξύ των σελίδων γίνεται με την χρήση SESSION μεταβλητών

Οι σελίδες συνοδεύονται από τα αντίστοιχα αρχεία .css που μορφοποιούν τις στατικές σελίδες που είναι γραμμένες σε html.

Συνολικά στην υλοποίηση, εφαρμόζονται επιγραμματικά τα εξής μέτρα προστασίας:

- Input Sanitization

Όλες οι εισόδους που δίνει ο χρήστης, ελέγχονται με συναρτήσεις όπως οι `mysqli_real_escape_string()`, `stripslashes()`, `strip_tags()`, `mysqli_set_charset()` και διώχνουν ανεπιθύμητες εισόδους (χαρακτήρες).

Προστασία από: **SQL Injection, XSS**

- Prepared Statements

Με συναρτήσεις όπως οι `mysqli_prepare()`, `mysqli_stmt_bind_param()`, `mysqli_stmt_store_result()` και `mysqli_stmt_close()`, δημιουργούμε “έτοιμα” SQL queries για ανταλλαγή δεδομένων με την βάση MySQL.

Προστασία από: **SQL Injection**

- Input Validation(RegEx)

Έλεγχος εισόδων του χρήστη για έγκυρα δεδομένα με χρήση Regular expressions(συνάρτηση `preg_match()`).

Προστασία από: **SQL Injection, XSS**

- Password strength

Τουλάχιστον 8 χαρακτήρες με υποχρεωτικά ένα κεφαλαίο, πεζό, νούμερο και ειδικό χαρακτήρα.

Προστασία από: **Brute Force Attack**

- Encrypted password at storage

Αποθήκευση κωδικού στην βάση δεδομένων με χρήση του κρυπτογραφικού αλγορίθμου SHA2-224.

Προστασία από: **Brute Force Attack, Insecure cryptographic storage**

- Human user verification

Χρήση ερωτήσεων ταυτοποίησης ανθρώπινου χρήστη που σε ρεαλιστικά σενάρια μία μηχανή δεν είναι σε θέση να απαντήσει ορθά (προς το παρόν), αποτρέποντας έτσι την απόπειρα login απο αυτοματοποιημένες εφαρμογές (bot). Επιλέξαμε αυτόν το αντίμετρο αντί του κλειδώματος λογαριασμού, ως επιπλέον επίπεδο ασφάλειας, γιατί διαφορετικά μία επίθεση θα μπορούσε να προκαλέσει βλάβη στην συνολική λειτουργικότητα. Για παράδειγμα να κλειδώσει προσωρινά όλους τους λογαριασμούς.

Προστασία από: **Brute Force Attack**

- Encrypted communication

Χρήση HTTPS πρωτοκόλλου για την επικοινωνία όλων των request από και προς τον Apache Server.

Προστασία από: **Overall Protection**

- XSS protection

Ενεργοποίηση του Header X-XSS-Protection στον Server για επιθέσεις Cross Site Scripting.

Προστασία από: **XSS**

- Limited user rights

Ο χρήστης που χρησιμοποιείται έχει πρόσβαση μόνο στην βάση δεδομένων της εφαρμογής και δικαιώματα μόνο για τις ενέργειες INSERT και SELECT.

Προστασία απο: **SQL Injection**

- Έλεγχος της IP που θέλει να έχει πρόσβαση στην σελίδα

Η διαδικασία που περιγράφεται στην συνέχεια.

Προστασία απο: **Session hijacking**

# Βάση δεδομένων

Δημιουργήσαμε μια βάση δεδομένων με όνομα Login\_Form η οποία περιέχει τα tables users και questions.

Το table users χρησιμοποιείται για την αποθήκευση των δεδομένων του χρήστη και περιέχει τα εξής πεδία:

- Το πεδίο id, το οποίο ορίζεται ως κλειδί του table και είναι ένας μοναδικός ακέραιος αριθμός για κάθε χρήστη που αυξάνεται αυτόματα σε κάθε νέα καταχώρηση.
- Το πεδίο username που περιέχει ένα αλφαριθμητικό συνθηματικό του χρήστη.
- Το πεδίο email που περιέχει την ηλεκτρονική διεύθυνση του χρήστη επίσης σε αλφαριθμητική μορφή.
- Το πεδίο password που περιέχει τον κωδικό του χρήστη σε αλφαριθμητική αλλά μη αναγνώσιμη μορφή.

```
MariaDB [Login_Form]> describe users_upgraded;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
username	varchar(255)	NO		NULL	
email	varchar(255)	NO		NULL	
password	varchar(255)	NO		NULL	
last_ip	varchar(45)	YES		NULL	
logged_now	tinyint(1)	YES		NULL	

6 rows in set (0.003 sec)

Ενα στιγμιότυπο του πεδίου των κωδικών οι οποίοι αποθηκεύονται με την χρήση της hash function SHA-224 είναι το εξής:

```
MariaDB [Login_Form]> select password from users;
```

password
c73e86b7ab345a466e1fee86ecf11969ff758a0da6f48be3f3419964
47c93b392455db957524a39dddb1d375ad2b850312680ae508672731
6121bfe52721028ebefd9f3f8f3e8aa3963f3341b5945cf949acf188

Το table questions εκτελεί την λειτουργία Human user verification όπως αναφέρθηκε και πιο πάνω και περιέχει τα εξής πεδία:

- Το πεδίο id με όμοιο τρόπο όπως στο table users.
- Το πεδίο question που αποθηκεύεται μία ερώτηση.
- Το πεδίο answer που αποθηκεύεται η σωστή απάντηση της αντίστοιχης ερώτησης.

```
MariaDB [Login_Form]> describe questions;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
question	varchar(255)	NO		NULL	
answer	varchar(255)	NO		NULL	

Οι ενδεικτικές ερωτήσεις που τοποθετήσαμε είναι:

```
MariaDB [Login_Form]> select * from questions;
```

id	question	answer
1	What is the color of the sky?	blue
2	How many legs does a human has?(number)	2
3	How many fingers does one human hand has?(number)	5
4	How many corners does a square have?(number)	4
5	How many corners does a triangle have?(number)	3
6	What is the color of red roses?	red
7	What is the color of yellow roses?	yellow
8	Write APPLE without the letter P	ALE
9	Write ORANGE without the letter G	ORANE
10	Write BANANA without the letter A	BNN

Η συγκριση της σωστής με την απάντηση του χρήστη γίνεται μέσω της php και είναι case insensitive.

## Παράδειγμα Log in και αντίμετρο Session hijacking

- Κατα την διαδικασία register ελέγχουμε αν το email υπάρχει ήδη στην βάση και τυπώνουμε το ανάλογο μήνυμα. (το πεδίο email δεν είναι κλειδί αλλά με αυτόν τον τρόπο είναι μοναδικό για κάθε χρήστη)  
Η βάση μετά την εισαγωγή του χρήστη someone:


```
MariaDB [Login_Form]> select * from users_upgraded;
```

id	username	email	password	last_ip	logged_now
1	username	email@gmail.com	a9a0b26eb4167f1c17c5e839288022ac37f918762588d3732723bb5c	192.168.2.3	0
2	username2	email@gmail2.com	a9a0b26eb4167f1c17c5e839288022ac37f918762588d3732723bb5c	192.168.2.3	0
3	anotheruser	email@newmail.com	9413e3c8a92a125683f00352bd39579ac9dab766019f20a9cd2ee93a	192.168.2.3	0
4	user	random@email.com	dd1869821ebfc4c9eb9b0b78b017a73aec9d4859786822cda4db0885	192.168.2.3	0
5	someone	someone@gmail.com	dd1869821ebfc4c9eb9b0b78b017a73aec9d4859786822cda4db0885	NULL	NULL


5 rows in set (0.001 sec)

- Μετά το login του someone, χρησιμοποιώντας το email και τον κωδικό, η βάση είναι:

### Login



someone@gmail.com



.....

What is the color of red roses?

Login

New here? [Sign Up!](#)

```
MariaDB [Login_Form]> select * from users_upgraded;
```

id	username	email	password	last_ip	logged_now
1	username	email@gmail.com	a9a0b26eb4167f1c17c5e839288022ac37f918762588d3732723bb5c	192.168.2.3	0
2	username2	email@gmail2.com	a9a0b26eb4167f1c17c5e839288022ac37f918762588d3732723bb5c	192.168.2.3	0
3	anotheruser	email@newmail.com	9413e3c8a92a125683f00352bd39579ac9dab766019f20a9cd2ee93a	192.168.2.3	0
4	user	random@email.com	dd1869821ebfc4c9eb9b0b78b017a73aec9d4859786822cda4db0885	192.168.2.3	0
5	someone	someone@gmail.com	dd1869821ebfc4c9eb9b0b78b017a73aec9d4859786822cda4db0885	192.168.2.3	1

```
5 rows in set (0.000 sec)
```

- Η σελίδα πλέον είναι προστατευμένη από επιθέσεις **session hijacking**. Γίνεται αναλυτικότερος έλεγχος των στοιχείων του χρήστη που προσπαθεί να εισέλθει στο σύστημα, λαμβάνοντας υπόψη και την IP διεύθυνση. Κώδικας στο login.php:

```
//update the table fields
function log_in($con,$mail){
    // get users ip
    $ip = $_SERVER['REMOTE_ADDR'];
    //this is not user input so no need to sanitize
    $update_q = "update users_upgraded set last_ip='$ip', logged_now=1 where email='$mail'";
    if(mysqli_query($con,$update_q)){
        header("Location: homepage.php");
    }else{
        $_SESSION['update_error'] = 1;
        header("Location: login.php");
    }
}
```

- Το αρχείο session\_auth.php πλέον επικυρώνει την σωστή σύνδεση του χρήστη ,ελέγχοντας την IP αυτού που προσπαθεί να έχει πρόσβαση στην homepage.php και αυτού που εκτέλεσε την διαδικασία μέσω του login.php σε συνδυασμό με την μεταβλητή logged\_now.

```
//need to check here if ip is the same and if logged_now = 1
$ip = $_SERVER['REMOTE_ADDR'];
$email = $_SESSION['email'];

$q = "select username, last_ip, logged_now from users_upgraded where email='$email'";
$res = mysqli_query($con,$q);
$row = mysqli_fetch_array($res,MYSQLI_ASSOC);

if($row['logged_now'] && ($row['last_ip'] == $ip)){
    $_SESSION['username'] = $row['username'];
    //if condition is true till here then we connect normally
}else{
    header("Location: login.php");
    exit();
}
```

- Τέλος, το αρχείο logout.php εκτός από το να καταστρέφει το session, κάνει unset την μεταβλητή logged\_now σε 0.

```
<?php
require('db_conn.php');
session_start();
$email = $_SESSION['email'];
$update_q = "update users_upgraded set logged_now=0 where email='$email'";

if(!mysqli_query($con,$update_q)){
    header('Location: homepage.php');
}else{
    session_destroy();
    // Redirect to the login page:
    header('Location: login.php');
}
?>
```

# Apache Server

Στον server apache εκτελέσαμε τις εξής τροποποιήσεις:

## Hide Directory Tree

Μεταβήκαμε στον κατάλογο `/etc/httpd/conf` όπου περιέχεται το αρχείο διαμόρφωσης (configuration file) `httpd.conf`. Εκεί, κάτω από τις ρυθμίσεις για τον κατάλογο `/var/www/html/` αφαιρέσαμε την τιμή `Indexes` από το `Options` directive. Με αυτόν τον τρόπο κάποιος χρήστης δεν μπορεί να περιηγηθεί στους φακέλους του καταλόγου που έχει πρόσβαση ο server παρά μόνο να φορτώσει απευθείας ένα έγγραφο. Εάν το επιχειρήσει τότε επιστρέφεται ο κωδικός 403 Forbidden. Μετά την ρύθμιση το αρχείο είναι:

```
<Directory "/var/www/html">
#
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important. Please see
# http://httpd.apache.org/docs/2.4/mod/core.html#options
# for more information.
#
#Options Indexes FollowSymLinks #previous option
Options FollowSymLinks

#
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
AllowOverride None

#
# Controls who can get stuff from this server.
#
Require all granted
```

## XSS protection

Στο ίδιο αρχείο στο τέλος του προσθέτουμε την ρύθμιση `Header always set X-XSS-Protection "1; mode=block"`, όπως προαναφέρθηκε.

```
Header always set X-XSS-Protection "1; mode=block"
```

## Encrypted communication

Ρυθμίσαμε τον sever ώστε να δέχεται `https` requests και τα δεδομένα να κρυπτογραφούνται κατά την μεταφορά τους.

Αρχικά κατεβάσαμε ένα module του apache ώστε να υποστηρίζει το `SSL` πρωτόκολλο με την εντολή:

```
sudo yum install mod_ssl
```



Μετά από αυτό εμφανίστηκε ένα configuration file *ssl.conf* στον κατάλογο */etc/httpd/conf.d*. Προτού το επεξεργαστούμε πρέπει να παράξουμε το κλειδί και το πιστοποιητικό που είναι απαραίτητα για το SSL. Αυτό γίνεται με την εντολή:

```
openssl req -x509 -sha256 -nodes -newkey rsa:2048 -keyout gfsselfsigned.key -out gfcert.pem
```

Και συμπληρώνουμε αντίστοιχα ότι ζητείται.

Αποθηκεύουμε τα παραγόμενα αρχεία *gfsselfsigned.key*, *gfcert.pem* σε φάκελο που γνωρίζουμε την διεύθυνση. Μπορούμε τώρα να ολοκληρώσουμε τις ρυθμίσεις του αρχείου *ssl.conf*. Μέσα στο πεδίο του directive *VirtualHost*, όπως φαίνεται πιο κάτω, τοποθετούμε ή τροποποιούμε της ήδη υπάρχουσες ρυθμίσεις ως εξής:

```
<VirtualHost _default_:443>
DocumentRoot "/var/www/html"
SSLEngine on
SSLCertificateFile /etc/httpd/conf.d/ssl/gfcert.pem
SSLCertificateKeyFile /etc/httpd/conf.d/ssl/gfsselfsigned.key
SSLCertificateChainFile /etc/httpd/conf.d/ssl/gfcert.pem
```

Βάζοντας *https://* πριν την διεύθυνση της σελίδας μας ο sever ανταποκρίνεται κανονικά και η σύνδεση είναι πλέον κρυπτογραφημένη όπως φαίνεται και από τις πληροφορίες της σελίδας.

