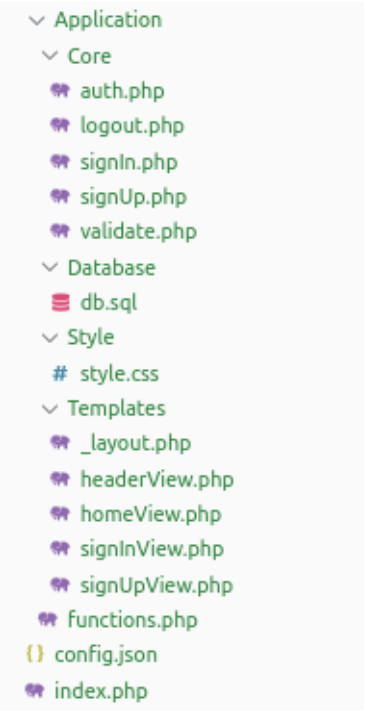
Bejelentkezés

Készítsen el egy bejelentkezési logikát az alábbiak szerint!



1. Készítse el a fájlstruktúrát a mellékelt kép szerint!

2. Az index.php elkészítése. Itt a megszokott módon a $page változó által navigálunk az egyes oldalak között. Változás, hogy a program indulásakor elindítjuk a munkamenetet, ami ilyenkor még ‘inicializálatlan’.

index.php > ...

```php
<?php

define('APPPATH', 'Application/');

$confPath='config.json';
$page = @$_GET['page'] ? $_GET['page'] : 'home';

session_start();

require_once APPPATH.'functions.php';

switch ($page)
{
    case 'signUp':  require_once APPPATH.'Core/signUp.php';     break;
    case 'signIn':  require_once APPPATH.'Core/signIn.php';     break;
    case 'auth':    require_once APPPATH.'Core/auth.php';       break;
    case 'validate':require_once APPPATH.'Core/validate.php';   break;
    case 'logout':  require_once APPPATH.'Core/logout.php';     break;
}

require_once APPPATH.'Templates/_layout.php';
```

Behúzzuk a functions.php-t, a switchel az egyes logikákat, madj a végén a _layout-al indulunk a megjelenítés felé.

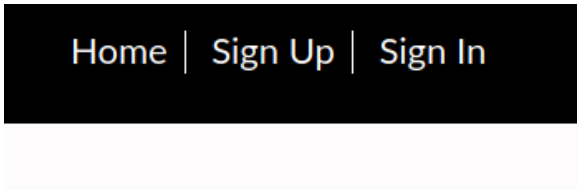
3. A megszokott **_layout.php** fájl továbbra is a teljes html struktúráért felelős.

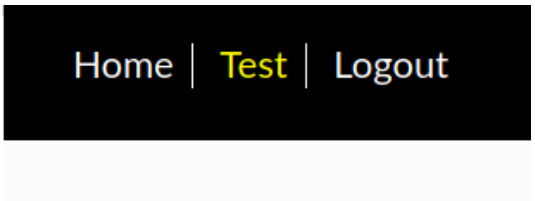
Application > Templates > _layout.php > ...

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="<?= APPPATH ?>Style/style.css" type="text/css">
</head>
<body>
    <?php require_once APPPATH.'Templates/headerView.php' ?>
    <?php require_once APPPATH."Templates/{$page}View.php" ?>
</body>
</html>
```

Ne felejtsük el belinkelni a CSS állományt!

4. A header.php ebben a feladatban már dinamikusan működik annak megfelelően, hogy a felhasználó be van-e jelentkezve, vagy sem.



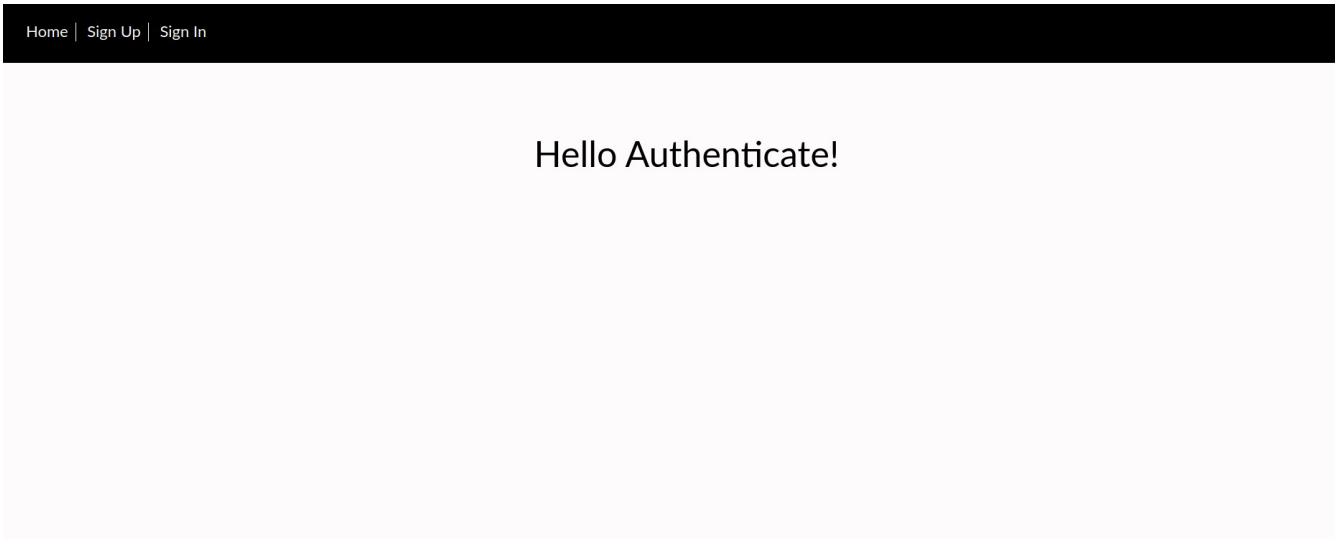


Application > Templates > headerView.php > ...

```php
<header class="main-header">
    <a href="index.php" class="hdr-btn">Home</a>

    <?php if( @!$_SESSION['userName']): ?>
        <a href="index.php?page=signUp" class="hdr-btn">Sign Up</a>
        <a href="index.php?page=signIn" class="hdr-btn">Sign In</a>
    <?php else: ?>
        <span class="hdr-btn hdr-usr"><?= $_SESSION['userName'] ?></span>
        <a href="index.php?page=logout" class="hdr-btn">Logout</a>
    <?php endif ?>

</header>
```

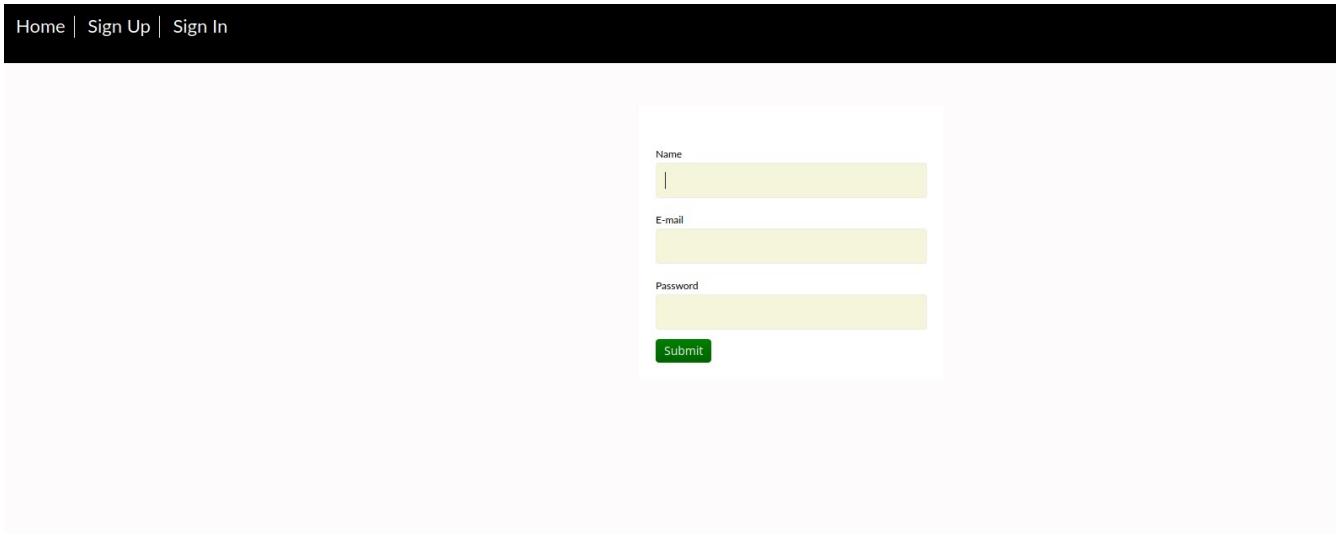
Látható, hogy egy elágazás dönt arról, hogy az egyes menügombok közül melyiket jelenítse meg a fejléc. Egyedül a Home gomb állandó, ha felhasználó nincsen bejelentkezve, a fejléc a Home, Sign Up és Sign In gombokat jeleníti meg, ellenkező esetben a Home, Felhasználó neve és a Logout opciókat.

5. Amikor az oldalt először látogatjuk meg, akkor a $page az alapértelmezett ‘home’ értéket kapja és az ennek megfelelő View-t jeleníti meg.



6. Ha a Sign Up-ra navigálunk az alábbi kép fogad:

Ez egy sima regisztrációs felület, név email-cím és jelszó inputokkal. A kód:



Az összes input-ot állítsuk required-re, és legyen egy autofocus mezőnk! A metódus POST legyen! Látható, hogy van az űrlapban egy $error változó. Ezt a signUpView logikájában, a signUp.php-ben deklaráljuk.

Application > Core > signUp.php > ...

```php
<?php

$error = @$_GET['error'] ? 'Email is already used!' : '';
```

Application > Templates > signUpView.php > ...

```php
<form action="index.php?page=auth" method="POST">
    <div class="login-form">
        <p><?= $error ?></p>
        <label for="">Name</label>
        <input type="text" name="name" autofocus required>

        <label for="">E-mail</label>
        <input type="email" name="email" required>

        <label for="">Password</label>
        <input type="password" name="pass" required>

        <input type="submit" class="btn btn-grn" value="Submit">
    </div>
</form>
```

7. Miután elküldtük az adatokat az auth.php-nak, az megpróbálja beszúrni az új felhasználót az adatbázisba. Az email primary key, ha egyezés van, az auth error-t dob vissza. A kód:

Application > Core > auth.php > ...

```php
<?php

$name   = $_POST['name'];
$email  = $_POST['email'];
$pass   = password_hash( $_POST['pass'],  PASSWORD_DEFAULT );


$config = getConfig($confPath);
$pdo    = getConnection($config);


if (!SignUP($pdo, $name, $email, $pass ))
{
    header('Location: index.php?page=signUp&error=1');
    die;
}

header('Location: index.php');

die;
```

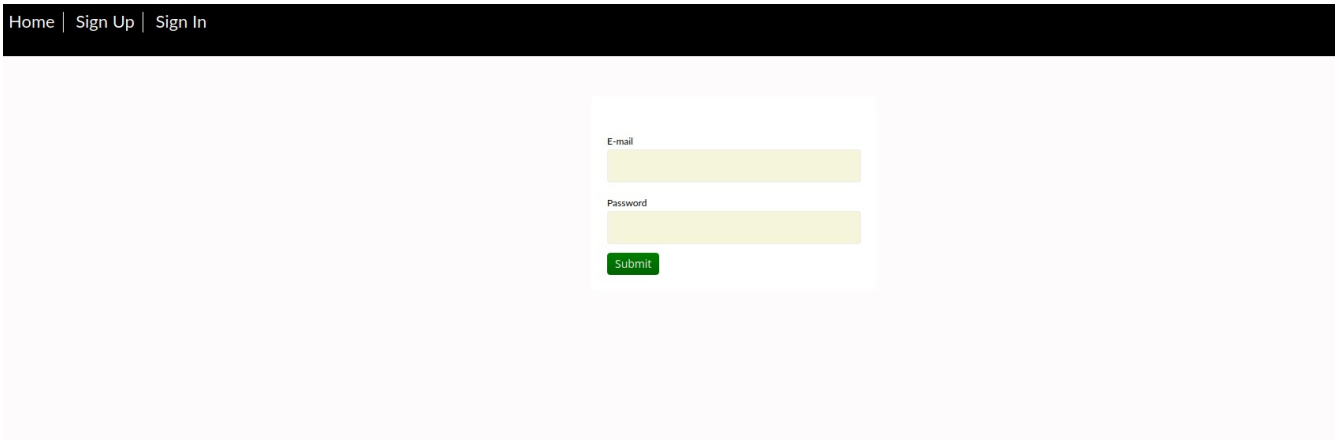
A szokásos függvények mellett (getConfig, getConnetcion), Most egy SignUp függvénnyel dolgozunk. Ha a függvény sikeresen lefut igazzal, ellenkező esetben hamissal tér vissza. Ha hamissal, az oldalt visszairányítjuk az űlaphoz, és elhelyezünk egy error kódot az url-ben.

```php
function SignUP( PDO $pdo, $name, $email, $pass )
{

    try
    {
        $smt = $pdo->prepare('INSERT INTO `users`(`name`,`email`,`password`) VALUES (:name, :email, :password)');

        $smt->bindParam(':name', $name);
        $smt->bindParam(':email', $email);
        $smt->bindParam(':password', $pass);

        if (!$smt->execute())
        {
            throw new RuntimeException($smt->errorInfo()[2]);
        }

        return true;
    }
    catch (RuntimeException $e)
    {
        return false;
    }
}
```

Ha igazzal, az oldalt visszairányítjuk a főoldalra – lehetne visszajelzést adni a sikeres regisztrációról.

Látható, hogy egy **`password_hash`** nevű függvényt használunk. Ez a jelszót a megadott kriptografikus kódolási algoritmussal lekódolja. Szabály, hogy a jelszót még az adatbázisban is kódolva tároljuk. Régebben erre pl. az md5 kódolást használták (`md5('password')`), de a kódolási eljárásban előfordulhattak ütközések, vagyis, hogy két jelszó hash-elt kódja azonos volt és az előállított kulcs nagyon gyorsan legenerálható a mai számítógépek teljesítménye mellett. Ennél egyel újabb az `sha1()` kódolás, ami egy 20byte-os kimenetet generál, de már ez is elavultnak számít 2017 óta. A `password_hash` minden kódolás esetében más és más karakterláncot állít elő. Ez a módszer már tartamaz minden információt arról, hogy hogyan is lett legenerálva, ellentétben az md5-ös kódolással. Ez szükséges a visszaalakításhoz.

8. Ezek után a Sign In fülön tudunk bejelentkezni a weboldalra:



A második form egyszerűen újrahasznosítja az első kódját.

Application > Templates > signInView.php > ...

```php
<form action="index.php?page=validate" method="POST">
    <div class="login-form">
        <p><?= @$error ?></p>

        <label for="">E-mail</label>
        <input type="email" name="email" required>

        <label for="">Password</label>
        <input type="password" name="pass" required>

        <input type="submit" class="btn btn-grn" value="Submit">
    </div>
</form>
```

A `$error` itt is megjelenítésre kerül, de itt már a signIn.php gondoskodik annak előállításáról:

Application > Core > signIn.php > ...

```php
<?php

$error = @$_GET['error'] ? 'Email or password is invalid' : '';
```

9. Elküldve a formot, az meghívja a validate.php-t ami, ami a zsokásos függvények mellett a SignIn() függvényel hitelesíti az adatokat.

Application > Core > validate.php > ...

```php
<?php

$email  = $_POST['email'];
$pass   = $_POST['pass'];


$config = getConfig($confPath);
$pdo    = getConnection($config);

$user = SignIn($pdo,$email, $pass );

if (!$user)
{
    header('Location: index.php?page=signIn&error=1');
    die;
}

$_SESSION['userName'] = $user['name'];

header('Location: index.php');

die;
```

10. Ha a SignIn() függvény a kapott adatok alapján igaz értékkel tér vissza, vagyis a validáció sikeres volt, a $_SESSION globálisban elhelyezzük a felhasználó nevét (az elhelyezni kívánt adatok tetszőlegesek. Jellemzően csak egy jelzés értékű változó, mondjuk logged=1 van letárolval, mert a php minden lekérésnél olvas az adatbázisból, amit egy userEntity osztály tárol és kezel).

```php
function SignIn($pdo, $email, $pass )
{
    try
    {
        $smt = $pdo->prepare('SELECT * FROM `users` WHERE `email` LIKE :email');

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

        if(!$smt->execute())
        {
            throw new RuntimeException($smt->errorInfo()[2]);
        }

        $hash = $smt->fetch();


        if( password_verify($pass, $hash['password']) )
        {
            return $hash;
        }

        return false;
    }
    catch (RuntimeException $e)
    {
        return false;
    }
}
```

A SignIn függvényben kiolvassuk az adott email-hez gartozó sort, majd összehasonlítjuk a visszafejtett jelszót a kapottal. Ezt a `password_verify()` függvénnyel tehetjük meg. Visszatérési értéke igaz, ha van gyezés, hamis különben.

11. Sikeres bjelentkezés után megjelenik a felhaszáló neve a fejlécben és a logout fül. A logout-ra kattintva a logout.php hívódik meg, ami törli a $_SESSION tömb tartalmát, így a nyomonkövethetőség elvész, majd átirányít.

Application > Core > logout.php

```php
<?php

$_SESSION = [];

header("Location: index.php");

die;
```