|  |  |
| --- | --- |
| D:\UKTC\UKTC tempatets\Logo\UKTC-logo-color.png | **Национална професионална гимназия по компютърни технологии и системи**  **гр.Правец при технически университет – СОФИЯ** |

**Дипломна работа**

ДЪРЖАВЕН ИЗПИТ ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ – ЧАСТ ПО ТЕОРИЯ НА ПРОФЕСИЯТА

на

Ученик: Николай Малинов Колибаров

ученик от 18125 (XII) клас

Тема: **Уеб сайт за експонатите в Лувъра**

**ПРОФЕСИОНАЛНО НАПРАВЛЕНИЕ:** 481 КОМПЮТЪРНИ НАУКИ

**ПРОФЕСИЯ:** 481020 системен програмист

**СПЕЦИАЛНОСТ:** 4810201 системно програмиране

Ученик: ………………….. Ръководител: ………………………

*Николай Колибаров Елена Първанова*

Правец, 2023

Съдържание

[**Увод** 4](#_Toc133416200)

[**Глава 1 Литературен обзор** 5](#_Toc133416201)

[**Frontend:** 5](#_Toc133416202)

[1. HTML5: 5](#_Toc133416203)

[2. JavaScript: 6](#_Toc133416204)

[3. CSS: 8](#_Toc133416205)

[4. Bootstrap: 9](#_Toc133416206)

[5. SASS: 10](#_Toc133416207)

[**Backend:** 11](#_Toc133416208)

[1. PHP: 11](#_Toc133416209)

[3. MySQL: 13](#_Toc133416210)

[4. XAMPP: 14](#_Toc133416211)

[**I D E:** 15](#_Toc133416212)

[1. Visual Studio Code: 15](#_Toc133416213)

[**Глава 2 Теоретична част** 17](#_Toc133416214)

[1. Структура на директориите 17](#_Toc133416215)

[на проекта: 17](#_Toc133416216)

[2. Структура на директориите 19](#_Toc133416217)

[на проекта: 19](#_Toc133416218)

[**Глава 3 Инженерно решение на поставената задача** 26](#_Toc133416219)

[1. Описание на работата на кода 26](#_Toc133416220)

[На фиг.1.1 е показан *PHP* код, който се грижи за регистрацията на потребител. 26](#_Toc133416221)

[Ето разглеждане на това какво прави кодът: 26](#_Toc133416222)

[1. *session\_start()* инициализира нова сесия или продължава съществуваща. 26](#_Toc133416223)

[2. *$\_GET* се използва за извличане на стойностите на параметрите *"name", "email"* и *"password"* от *URL*. 26](#_Toc133416224)

[3. Създава се масив, наречен *$registerErrors*, за да се съхранят всички грешки при регистрацията, които могат да възникнат. 26](#_Toc133416225)

[4. С помощта на условния оператор *if* се проверява дали дължината на паролата е по-малка от 5 символа. Ако е така, се добавя грешка в масива *$registerErrors*. 26](#_Toc133416226)

[5. Ако има грешки при регистрацията, те се съхраняват в масива *$registerErrors*, като се изтриват предишните грешки (ако има такива), използвайки функцията *unset().* След това грешките се записват в сесията, използвайки *$\_SESSION*. Накрая, потребителят се пренасочва към страницата *"signIn.php"* и се приключва изпълнението на скрипта с помощта на функцията *die().* 26](#_Toc133416227)

[6. Ако няма грешки при регистрацията, тогава се изпълнява блока *else*. В него се осъществява връзката към базата данни с помощта на *PDO* *класа* на *PHP*. След това паролата се криптира с функцията *password\_hash().* Връзката към базата данни се използва, за да се въведат данните на новия потребител, използвайки *SQL* *заявка* *INSERT INTO*. След успешното добавяне на потребителя се пренасочва към страницата *"register.php"* и се приключва изпълнението на скрипта с помощта на функцията *die().* 26](#_Toc133416228)

[7. Ако възникне грешка при връзката към базата данни, се извежда съобщение за грешка. 26](#_Toc133416229)

[Фиг.3.1 – PHP код за регистрация 27](#_Toc133416230)

[на потребители 27](#_Toc133416231)

[PHP кодът, показан на фиг.3.2, е отговорен за потвърждение на входа на потребителите в уеб приложение. Първоначално, сесията се стартира с функцията *session\_start().* След това се извличат имейл и парола от заявката *GET*, подадена от формата за вход. Сървърното име, потребителското име и паролата от базата данни се определят в началото на кода, а след това се прави опит за свързване с базата данни. Ако свързването бъде успешно, се извличат данните на всички потребители от таблицата *user*. 28](#_Toc133416232)

[След това се извършва обхождане на всички потребители с цикъл *for*. За всеки потребител се проверява дали имейлът и паролата отговарят на тези, въведени от потребителя в формата за вход. Вместо да се сравнява паролата чрез оператора ==, този код използва функцията *password\_verify()*, която сравнява подадената парола с хешираната парола, която се съхранява в базата данни. Ако имейлът и паролата съвпадат, сесията се обновява с информацията за успешен вход и след това потребителят се пренасочва към началната страница на приложението. 28](#_Toc133416233)

[Ако не бъде намерен потребител с подадените имейл и парола, се създава съобщение за грешка за вход и потребителят се пренасочва към страницата за регистрация. 28](#_Toc133416234)

[Фиг.3.2 – PHP код за влизане 29](#_Toc133416235)

[в акаунт 29](#_Toc133416236)

[Кодът,показан на фиг.3.3, е за генериране на динамична уеб страница, която показва снимки на експонатите в музея. Първоначално се изгражда връзка към базата данни, след което се използва заявка за да се изберат редове от таблицата *"exhibits"* в зависимост от определен филтър, който се получава от формата, която се предава чрез *POST*. Ако не се предаде филтър, то всички редове ще бъдат взети. След това се използва *while* *цикъл*, за да се обходят всички избрани редове и да се изградят *HTML елементи* за визуализация на тези редове като картички с рисунки на експонатите. Има възможност да се избере един експонат и да се види повече информация за него, като се препраща към друга страница с форма, която се предава метод *GET*. Когато се избира експонат, неговото име и идентификационен номер се изпращат към другата страница във формата. 30](#_Toc133416237)

[Кодът на фиг.3.4. зарежда изложбите от базата данни и показва всяка една в отделна картичка, като при клик върху нея се препраща към подробности за конкретния експонат, извлича детайли за него от базата данни, като изобразява снимката му и неговото име, категория и място на произход. 31](#_Toc133416238)

[*PHP* кодът на фиг.3.5 създава таблица с информация за потребители (admin panel) от базата данни *"thelouvre".* Таблицата съдържа четири колони*: "user Id", "name", "email" и "password".* 32](#_Toc133416239)

[Кодът първо установява връзка с базата данни чрез *PDO обект*. След това проверява дали в *URL* има заявка за изтриване на потребител от таблицата и, ако има, извършва изтриването чрез *SQL* заявка към базата данни. 32](#_Toc133416240)

[След това кодът извлича информацията от таблицата *"user"* чрез *SQL* заявка, която изпълнява и връща данните като масив. След това цикълът преминава през всички редове в масива и използва *HTML* таблица, за да ги покаже в уеб страницата. 32](#_Toc133416241)

[В последната колона на таблицата, за всеки ред се показва бутон *"Delete",* който изпраща заявка за изтриване на потребител към същата страница с помощта на *GET* метода. 32](#_Toc133416242)

[Фиг.3.5 - PHP код за създаване на таблица 33](#_Toc133416243)

[с информация за потребителите за страницата *admin.php* 33](#_Toc133416244)

[Кодът на фиг.3.6 е във файл на езика за стилове *Sass* с името *"main.scss".* В него се дефинират променливи за цветове (например, *$primary, $secondary, $light, $dark* и *$link-color*), които се използват по-късно в темата на *Bootstrap*, която се импортира в края на файла. 33](#_Toc133416245)

[Освен това, в кода се дефинират и някои допълнителни стилове за елементите на формулярите и за елемент с клас *"bgr-r-image".* В *HTML* документите, които използват този файл, ще се използва *Bootstrap* тема със специфични цветове и стилове за формуляри и елементи на страницата. 33](#_Toc133416246)

[Фиг.3.6 – Код за добавяне 34](#_Toc133416247)

[на класове от *Sass* файл 34](#_Toc133416248)

[Кодът на фиг.3.7 представлява *JavaScript* скрипт, който се използва за създаване на интерактивна карта на музей Лувър в Париж. В него се използва библиотеката за *JavaScript* карти *Leaflet.js*. Картата се създава като се използва *L.map()* функцията за създаване на мапа с определени параметри, като например координатната система *(CRS),* минимално ниво на увеличение и други. След това се задават координатите на картинката, която ще се използва като слой на картата, като се използва *L.imageOverlay()* функцията. Картата включва маркери на различни изложби в музея, като за всяка изложба се задава съответния маркер и текст за *popup*.Картата се центрира и увеличава с *L.setView()* функцията, която задава координатите и нивото на увеличение на картата. 34](#_Toc133416249)

[2. Структура на базата данни 37](#_Toc133416250)

[и таблиците в нея: 37](#_Toc133416251)

[**3.** **SQL CREATE заявки за** 39](#_Toc133416252)

[**създаване и попълване на таблици** 39](#_Toc133416253)

[**Глава 4 Анализ на получените резултати, приложимост и изводи** 41](#_Toc133416254)

[**1.** **Работа на уебсайта** 41](#_Toc133416255)

[**2.** **Заключение** 49](#_Toc133416256)

[**Използвана литература** 50](#_Toc133416257)

[Списък с използваните означения и съкращения 51](#_Toc133416258)

# **Увод**

Настоящата дипломна работа има за цел създаването на сайт, което служи за информация на хора, които желаят да посетят музея Лувъра. Създадено е със средствата на маркиращият език ***HTML (HyperText Markup Language)*** и стилизиращият език ***CSS (Cascading Style Sheet).***

Сайтът съдържа най-важната информация свързана с Лувъра, както и някои популярни експонати. Системата е направена с помощта на ***Visual Studio Code*** и използва Бази Данни чрез ***MySQL-***поддръжка.

Налични са всички по-важни функции, които се изискват за такъв тип сайт, а именно:

1. Регистрация на ***потребители*** чрез потребителско име, имейл и парола.
2. Вписване на ***потребители*** чрез потребителско име и парола.
3. Вписване като обикновен ***потребител*** или ***администратор***, от което зависи и нивото на достъп и вида операции, които могат да се изпълняват.

Потребителският ***интерфейс*** е опростен, за да не затруднява работата на ***потребителите***.

# **Глава 1 Литературен обзор**

В настоящия проект са използвани следните технологии, езици, среда за разработка ***(I D E – Integrated Developement Enviroment)*** и работни рамки ***(FrameWorks):***

Frontend:

1. HTML5(HyperText Markup Language)
2. JavaScript
3. CSS(Cascading Style Sheet)
4. Bootstrap
5. SASS

Backend:

1. PHP(Hypertext Preprocessor)
2. LeafLet
3. MySQL
4. XAMPP(X-operating system, Apache, Mysql, Php, Perl)

I D E(Integrated Developement Enviroment):

1. Visual Studio Code

## **Frontend:**

### HTML5:

**Какво е HTML5?**

***HTML5*** е модерен и все по-използван и навлизащ маркиращ език за структуриране и представяне на съдържание в ***интернет***.

Езикът добавя много нови синтаксови функции, като ***video, audio и canvas***, които подобряват работата с мултимедийно и графично съдържание.

***HTML5*** се използва често в приложенията за мобилни устройства, мобилни сайтове и операционни системи, като ***Firefox OS, Ubuntu Touch, Tizen и други.***

Групата разработчици ***WHATWG(Web Hypertext Application Technology Working Group)*** започва работа по новия стандарт през 2004 година. По това време, ***HTML 4.01*** не е бил обновяван от 2000-та година и организацията ***W3C(World Wide Web Consortium)*** се фокусира в бъдещата разработка на ***XHTML 2.0***.HTML кодът се поставя във файл с разширение ***\*.html***.На **фиг .1.1** е показан примерен код за извеждане на текста “Hello World!” в уеб страница, чрез ***HTML5***.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <p>Hello World!</p>

</body>

</html>

**Фиг.1.1 –** Примерен код на ***HTML5***

### JavaScript:

**Какво е JavaScript?**

***JavaScript*** е един от най-използваните ***скриптови езици*** за програмиране в ***интернет***. Разработен е през 1995 година от ***Брендан Айк*** и осигурява моментното и динамично взаимодействие между ***потребител и браузър***. Днес смело може да заявим, че почти няма нещо, което ***JavaScript*** не може да направи. Благодарение на него уеб страниците, които използваме в момента, изпълняват много повече функции от това просто да зареждат данни.

**За какво се използва JavaScript?**

Функциите на ***JavaScript*** са се разширили значително през годините. Първоначално езикът просто е регулирал начинa, по който потребителят си взаимодейства с уебсайта: онлайн въпросници и форми за контакт, подаване на обратна връзка под формата на известия и важни съобщения и др. Днес обаче езикът е еволюирал до такава степен, че с него може дори да управлявате дрон. ***JavaScript*** кодът се поставя във файл с разширение ***\*.js***.На **фиг.1.2** е показан примерен код за извеждане на текста “Hello World!” в конзолата на уеб страница, чрез ***JavaScript***.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        console.log("Hello World!");

    </script>

</body>

</html>

**Фиг.1.2** – Примерен код на ***JavaScript***

### CSS:

**Какво е CSS?**

***CSS*** е език за описване на презентацията и стиловете на елементите в един ***HTML/XML документ***. ***CSS*** е една от основните технологии, използвани в уеб, редом с ***HTML*** и ***JavaScript***.

При създаване на ***HTML*** страница, съдържанието й се описва с ***HTML*** код, а презентацията на това съдържание, тоест как ще изглежда то в браузъра, се описва с ***CSS*** код.

***CSS*** кодът се поставя във файл с разширение ***\* .css***. След това този файл може да се използва във всяка една страница от уеб сайта. Така че стилът ще се намира на едно място и при промяна ще се отразява на всички уеб страници. На **фиг.1.3** е показан примерен код за стилизация със ***CSS*** на параграф, в уеб страница, с текст “Hello World!”.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style>

        p{

            font-family: 'Times New Roman', Times, serif;

            color: red;

            font-size: 30px;

        }

    </style>

</head>

<body>

    <p>Hello World!</p>

</body>

</html>

**Фиг.1.3** – Примерен код на ***CSS***

### Bootstrap:

**Какво е Bootstrap?**

***Bootstrap*** е платформа, която е комбинация от ***HTML***, ***CSS*** и ***JavaScript*** код. ***Bootstrap*** напълно поддържа ***CSS3*** и ***HTML5***. Последеният факт е много ценен, тъй като ***CSS3*** и ***HTML5*** са стандарт за уеб дизайн през последните години. В интерес на истината – ***Bootstrap*** се превръща в стандарт за уеб разработки. Той е съвместим с всички основни ***уеб браузъри***. ***Bootstrap*** сайтовете се адаптират много лесно за всякакви устройства. Използвайки комбинация от ***JavaScript***, ***CSS*** и ***интелигентни мрежи***, ***Bootstrap*** позволява да се изпълни почти всеки елемент от дизайна, който може да се изиска. На **фиг.1.4** е показано как да вградим ***Bootstrap*** в дадена уеб страница чрез ***CDN (Content Delivery Network)***, което ни позволява да го използваме без да е необходимо сваляне.

<!-- Последно компилиран и минимизиран CSS -->  
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">  
  
<!-- jQuery библиотека -->  
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>  
  
<!-- Последно компилиран JavaScript -->  
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

**Фиг.1.4 –** Вграждане на ***Bootstrap*** в уеб страница чрез

***CSS***, ***JQuery*** и ***JavaScript***

### SASS:

**Какво е *SASS*?**

***SASS*** е предпроцесорен скриптов език, който се интерпретира или компилира в ***CSS***. ***SassScript*** е самият скриптов език. ***SASS*** се състои от два синтаксиса. Оригиналният синтакси e подобен на ***Html***. По-новият синтаксис, ***SCSS (Sassy CSS)***, използва блоково форматиране като това на ***CSS***. Той използва скоби за обозначаване на кодови блокове и точка и запетая за разделяне на правила в блок. Синтаксисът с отстъп и ***SCSS*** файловете традиционно получават съответно разширенията ***.sass*** и ***.scss***. На **фиг.1.5** е представен примерен код на ***SASS***.

$primary: #212124;

$secondary: #c51f1f;

$light: #F5F7F8;

$dark: #212124;

$link-color: $light;

$link-decoration: none;

$theme-colors: (

  "primary":    $primary,

  "secondary":  $secondary,

  "light":      $light,

  "dark":       $dark

);

**Фиг.1.5** – Примерен код на ***SASS***

за добавяне на цветове към сайта

## **Backend:**

### PHP:

**Какво е PHP?**

***PHP*** е скриптов език. Използва се предимно в ***интернет*** среда за изпълнение на широк кръг от услуги.

***PHP*** се разпространява под отворен лиценз ***(PHP License),*** който позволява безплатно разпространяване на програмния код на интерпретатора на езика, както и създаването на производни интерпретатори под други лицензи с уговорката, че тези интерпретатори не могат да включват ***PHP*** в името си. Фактът, че ***PHP*** се разпространява свободно, го прави удачен избор за изграждане на уеб сървър, базиран изцяло на свободни продукти ***– GNU/Linux, Apache, MySQL/PostgreSQL*** и др.

***PHP*** файловете могат да съдържат текст, ***HTML***, ***CSS***, ***JavaScript*** и ***PHP*** код и имат разширение ***\*.php***.На **фиг.1.5** е показан примерен код на ***PHP*** за извеждането на текста “Hello World!” в уеб страница.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>

        <?php

        echo 'Hello, World!';

        ?>

    </h1>

</body>

</html>

**Фиг.1.5** – Примерен код на ***PHP***

1. **LeafLet:**

**Какво е LeafLet?**

***Leaflet*** ***е*** ***JavaScript*** ***библиотека*** с отворен код, използвана за изграждане на приложения за уеб картографиране. Пуснат за първи път през 2011 г., той поддържа повечето мобилни и настолни платформи, като поддържа ***HTML5*** ***и*** ***CSS3***. Сред потребителите му са ***FourSquare, Pinterest и Flickr***.

***Leaflet*** позволява на разработчиците без ***GIS*** опит много лесно да показват уеб карти, хоствани на публичен сървър, с незадължителни наслагвания с мозайки. Може да зарежда данни за функции от ***GeoJSON файлове***, да ги стилизира и да създава интерактивни слоеве, като например маркери с изскачащи прозорци при щракване. Разработен е от ***Володимир Агафонкин***, който се присъедини към ***Mapbox*** през 2013 г.

var map = L.map('map', {

        crs: L.CRS.Simple,

        minZoom: -5

      });

      var bounds = [[0,0], [1000,1000]];

      var image = L.imageOverlay('louvre-museum-map-level-1.jpg', bounds).addTo(map);

      var sol = L.latLng([ 145, 175.2 ]);

      L.marker(sol).addTo(map).bindPopup('Something');

      map.setView( [70, 120], 1);

    }

    window.onload = function() {

      initMap();

**Фиг.1.6 –** Примерен код на **Leaflet**

### MySQL:

**Какво е MySQL?**

***MySQL***, система за управление на релационни ***бази данни*** с отворен код ***(RDBMS),*** съществува от 1995 г. Създадена е от ***MySQL AB***, която по-късно става ***Oracle*** ***Corporation***. Софтуерът използва ***SQL*** като основен език за данни и съхранява ***данни*** в таблици на дисковото устройство на сървъра. Данните могат да се съхраняват свободно в определени граници или обвързани със схема, която определя как трябва да бъдат структурирани.

***MySQL*** е софтуер с отворен код и може да бъде изтеглен безплатно. Няма ограничения за броя ***сървъри***, ***потребители*** или ***бази данни***, които можете да създадете с него, докато абонаментът ви остава активен.

***MySQL*** е популярна ***RDBMS*** за уеб приложения, защото е лесна за използване, бърза и надеждна. Освен това софтуерът има голяма потребителска общност и са налични много опции за търговска поддръжка. ***MySQL*** също така предлага голяма гъвкавост, като ви позволява да съхранявате данни в различни формати и да дефинирате свои собствени схеми.На **фиг.1.7** са показани примерни заявки за създаване на таблица и съответно добавяне на стойности към нея в ***MySQL***.

CREATE TABLE table\_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
   ....  
);

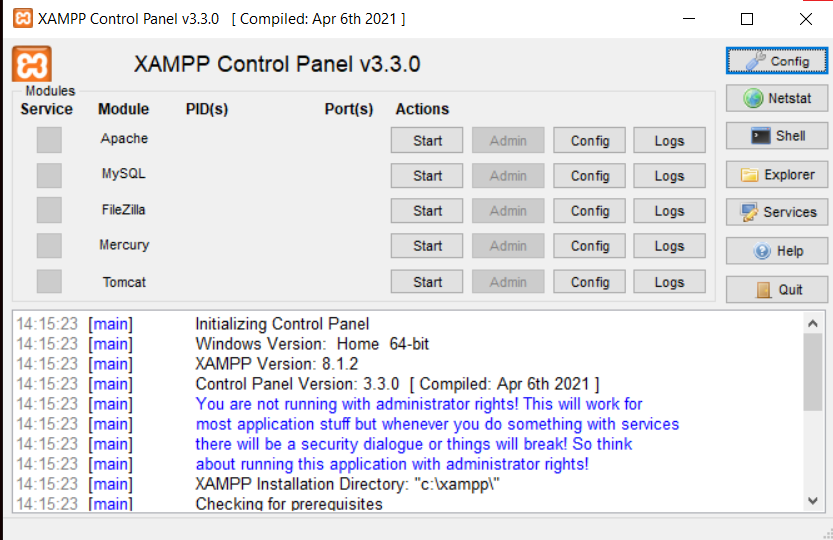
INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

**Фиг.1.7** – Примерни заявки в ***MySQL***

### XAMPP:

**Какво е XAMPP?**

***XAMPP*** е съкращение за междуплатформени, ***Apache, MySQL, PHP*** и ***Perl*** и ви позволява да създавате сайт офлайн, на локален уеб сървър на вашия компютър. Това просто и леко решение работи на ***Windows, Linux*** и ***Mac*** – оттук и ***„кросплатформената“*** част. Може да бъде срещнато и под наименование ***WEB-Stack***, което не е синоним на конкретното използвания набор, а по-скоро обобщаващо наименование за всички пакети от подобен вид.На **фиг.1.8** е представен интерфейсът на ***XAMPP*** контрол панелът.



**Фиг.1.8** – Интерфейсът на ***XAMPP*** контрол панела

Тъй като не е самостоятелно приложение, ***XAMPP*** предоставя два основни компонента за неговото инсталиране – ***Apache***, който се използва за създаване на локалния сървър, и ***MySQL***, който можете да използвате като база данни за вашия уебсайт.

Може би се чудите защо и как разработчиците използват локален сървър . Отговорът е прост – позволява им да създадат локално копие на сайта, в което могат да изпробват нови актуализации на плъгини, преди да ги внедрят в неговата версия на живо. По този начин те могат да предотвратят и открият потенциални грешки и проблеми, които могат да възникнат.

**Какво е Apache?**

***Apache*** е най-широко използваният софтуер за уеб сървър. Разработен и поддържан от ***Apache Software Foundation***, ***Apache*** е софтуер с отворен код, достъпен безплатно. Той работи на 67% от всички уеб сървъри в света. Той е бърз, надежден и сигурен. Той може да бъде силно персонализиран, за да отговори на нуждите на много различни среди чрез използване на разширения и модули.

Така че основно уеб сървърът е софтуерът, който получава вашата заявка за достъп до уеб страница. Той изпълнява няколко проверки за сигурност на вашата HTTP заявка и ви отвежда до уеб страницата. В зависимост от страницата, която сте заявили, страницата може да поиска от сървъра да стартира няколко допълнителни модула, докато генерира документа, който да Ви служи. След това Ви връчва искания от вас документ.

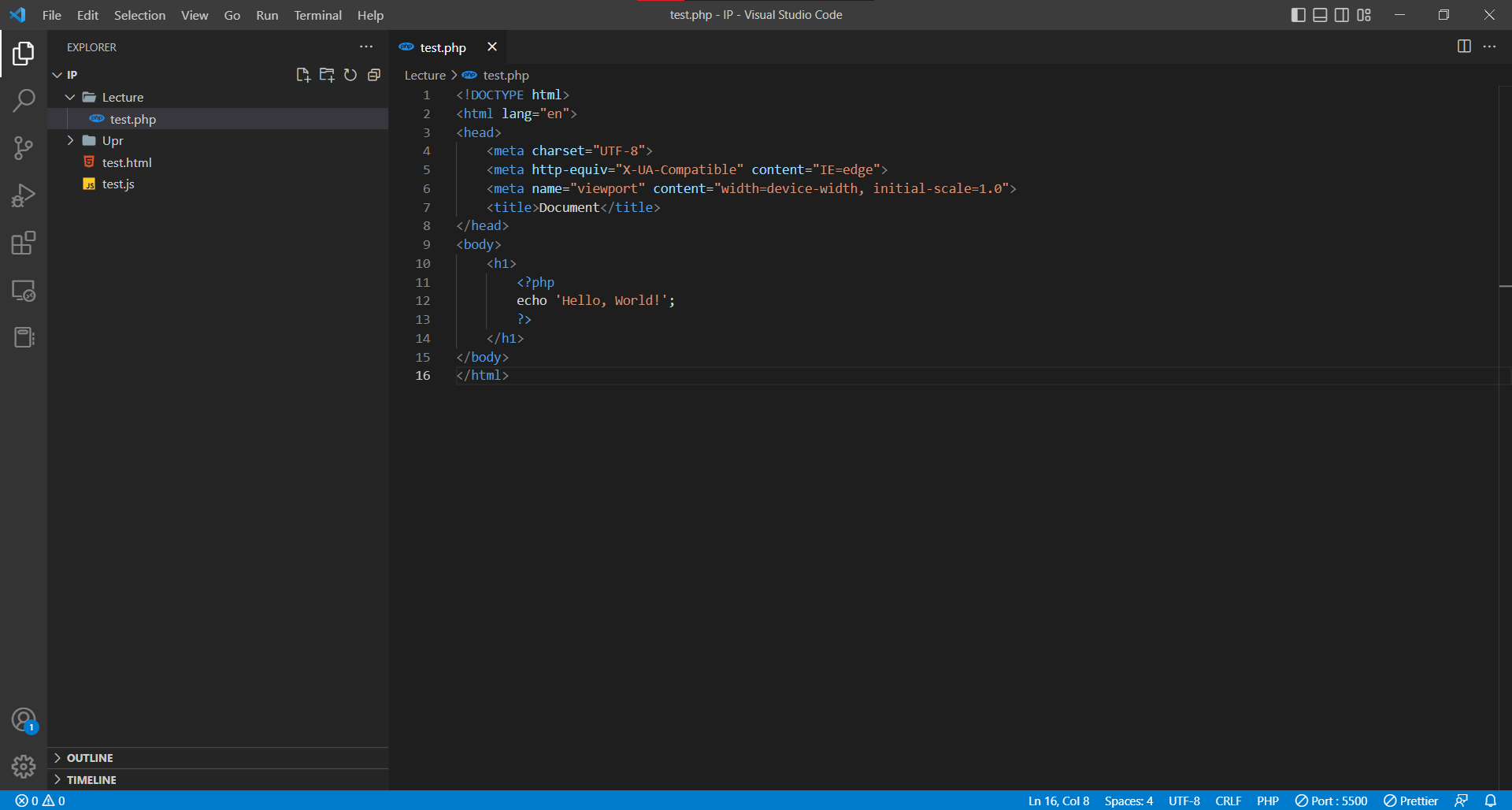
## **I D E:**

### Visual Studio Code:

**Какво е Visual Studio Code?**

***Visual Studio Code*** е редактор на програмен код за ***Windows, Linux*** и ***OS X***.Това е първият редактор на ***Microsoft***, който може да се ползва под ***Linux*** и ***macOS***. Поддържа богат набор от инструменти за разработване като ***дебъгване***, вграден ***Git Control, IntelliSense,*** и др. Той също така дава възможност за персонализиране, което означава, че потребителите могат да променят темата на редактора, клавишните комбинации, настройките и др. Редакторът е е безплатен и публично достъпен за преглед.

***Visual Studio Code*** е базиран на ***Electron***, който е базиран на ***Chromium***, използван да разгръща ***io.js*** приложения за десктопа. ***Visual Studio Code*** използва ***Blink layout engine***, за интерфейс. На **фиг.1.9** е представен потребителският интерфейс на ***Visual Studio Code*.**



**Фиг.1.9** – Потребителският интерфейс

на ***Visual Studio Code***

# **Глава 2 Теоретична част**

***HTML (HyperText Markup Language)*** – език за маркиране, използван за създаване и структуриране на съдържанието на уеб страница. ***HTML*** се използва за определяне на структурата и оформлението на уеб страница с помощта на елементи като заглавия, абзаци, изображения и връзки.

***CSS (Cascading Style Sheets)*** - стилов език, използван за описание на представянето на документ, написан на ***HTML***. ***CSS*** позволява на уеб разработчиците да определят външния вид и усещането на уебсайт, включително цветове, шрифтове и оформление.

***PHP (Hypertext Preprocessor)*** - сървърен скриптов език, използван за създаване на динамични уеб страници. ***PHP*** често се използва във връзка с база данни за създаване на интерактивни уебсайтове и уеб приложения.

***JavaScript*** - скриптов език, използван за създаване на интерактивни уеб страници и динамични уеб приложения. ***JavaScript*** позволява на уеб разработчиците да добавят интерактивност към своите уебсайтове, включително неща като анимации, валидиране на въвеждане от потребителя и динамични актуализации на съдържанието.

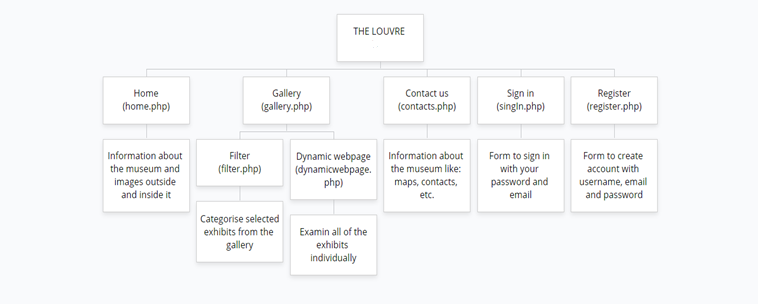
***Bootstrap*** - популярна рамка за разработка отпред, използвана за изграждане на адаптивни и ориентирани към мобилните устройства уебсайтове. ***Bootstrap*** предоставя предварително изградени ***UI*** компоненти и инструменти за оформление, което улеснява уеб разработчиците да създават модерни и професионално изглеждащи уебсайтове.

***Visual Studio Code*** е редактор на програмен код за ***Windows*** ,***Linux*** и ***OS*** ***X***. Това е първият редактор на ***Microsoft***, който може да се ползва под ***Linux*** и ***macOS***.

1. Структура на директориите

на проекта:

Проектът е разположен в директория, като в нея се съдържа отделните файлове отговарящи за неговата работа, както следва :

В избрана от потребителя папка /директория/ на компютъра се намира всяка една страница от менюто на уебсайта. На **фиг.2.1** е представен ***Sidemap*** на проекта.

**Фиг.2.1** –***Sidemap*** на проекта

1. Структура на директориите

на проекта:



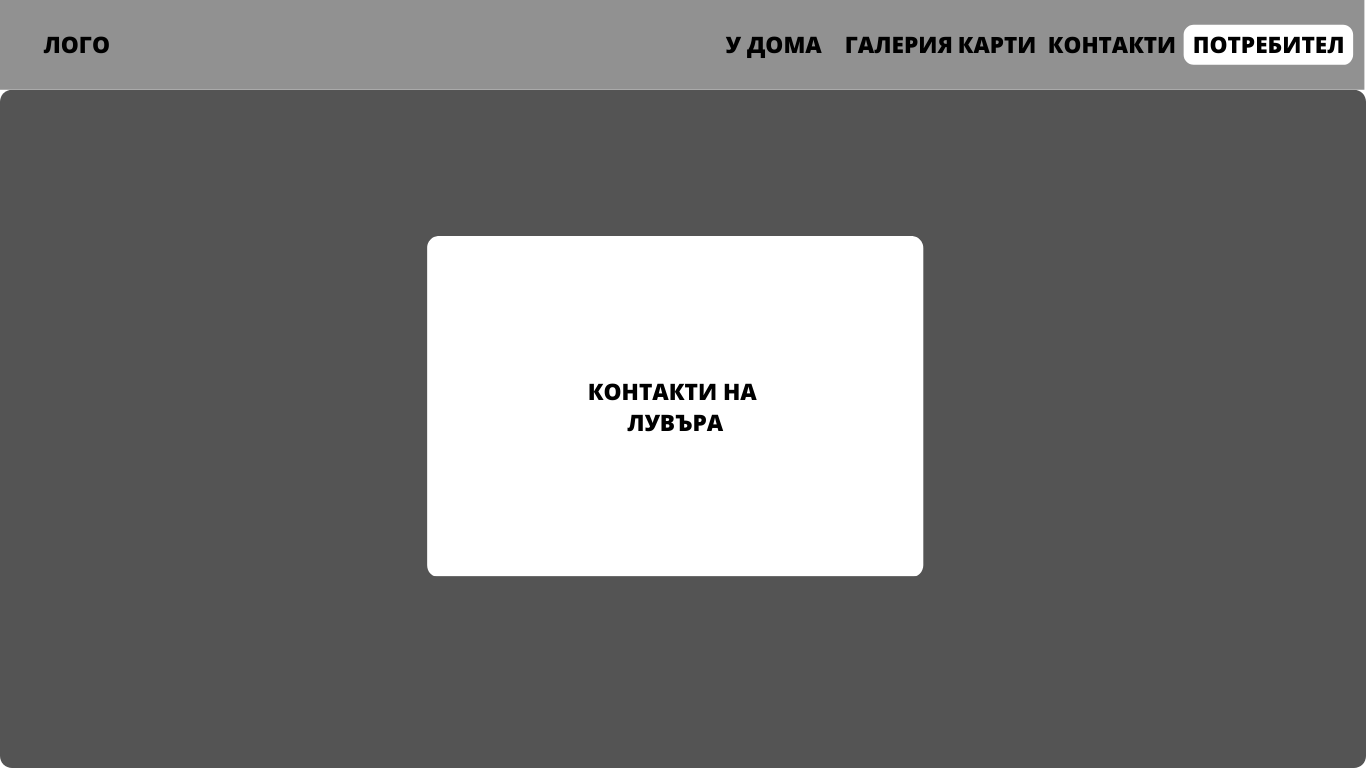
**Фиг.2.2** – ***Home page(home.php)***

Първата страница на уебсайта – ***home.php* (фиг.2.2).** Съдържа най-различни снимки на Лувъра,както и информация за него. В полето отгоре има бутони за отвеждане до галерията.



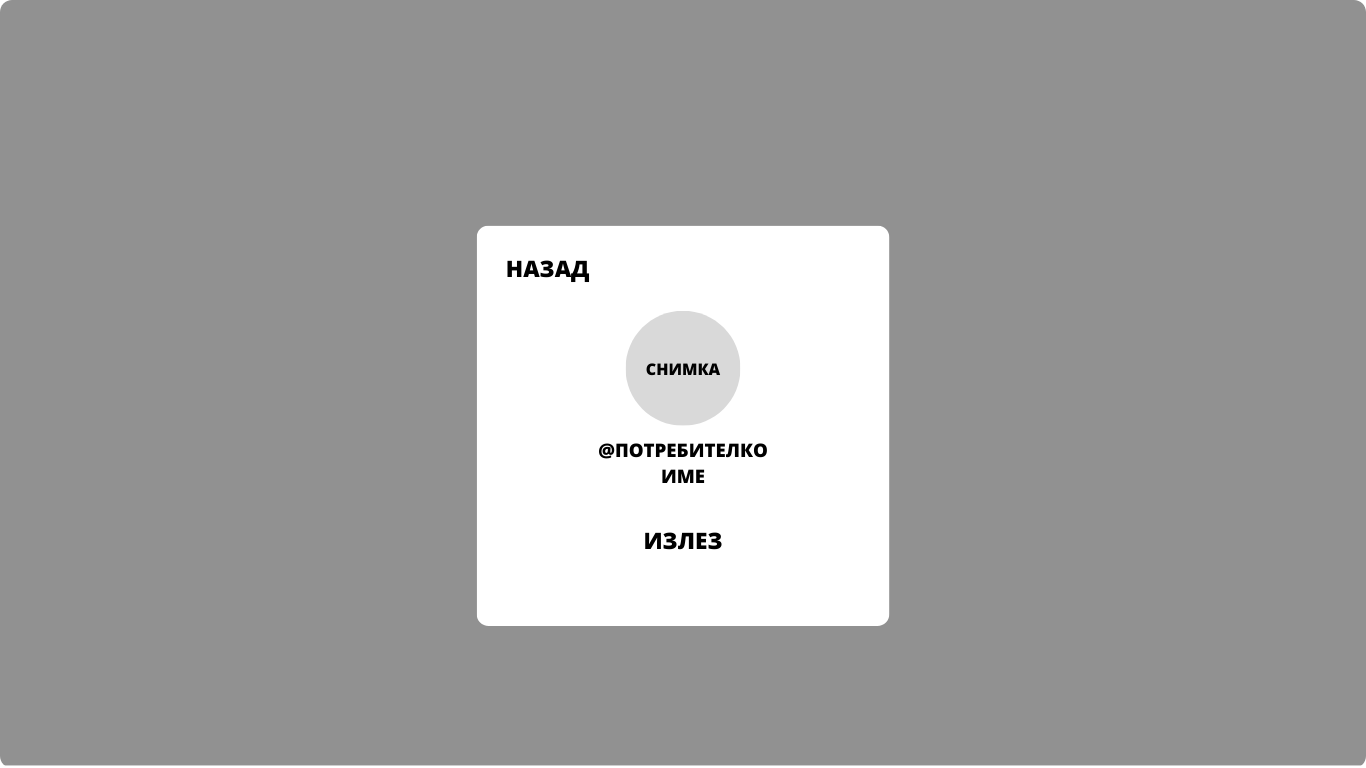
**Фиг.2.3 – *Gallery(gallery.php)***

Втората страница на проекта е ***gallery.php* (фиг.2.3).** По-надолу тя съдържа снимки на експонати от Лувъра, върху които потребителят може да кликне, за да види повече информация за експоната.



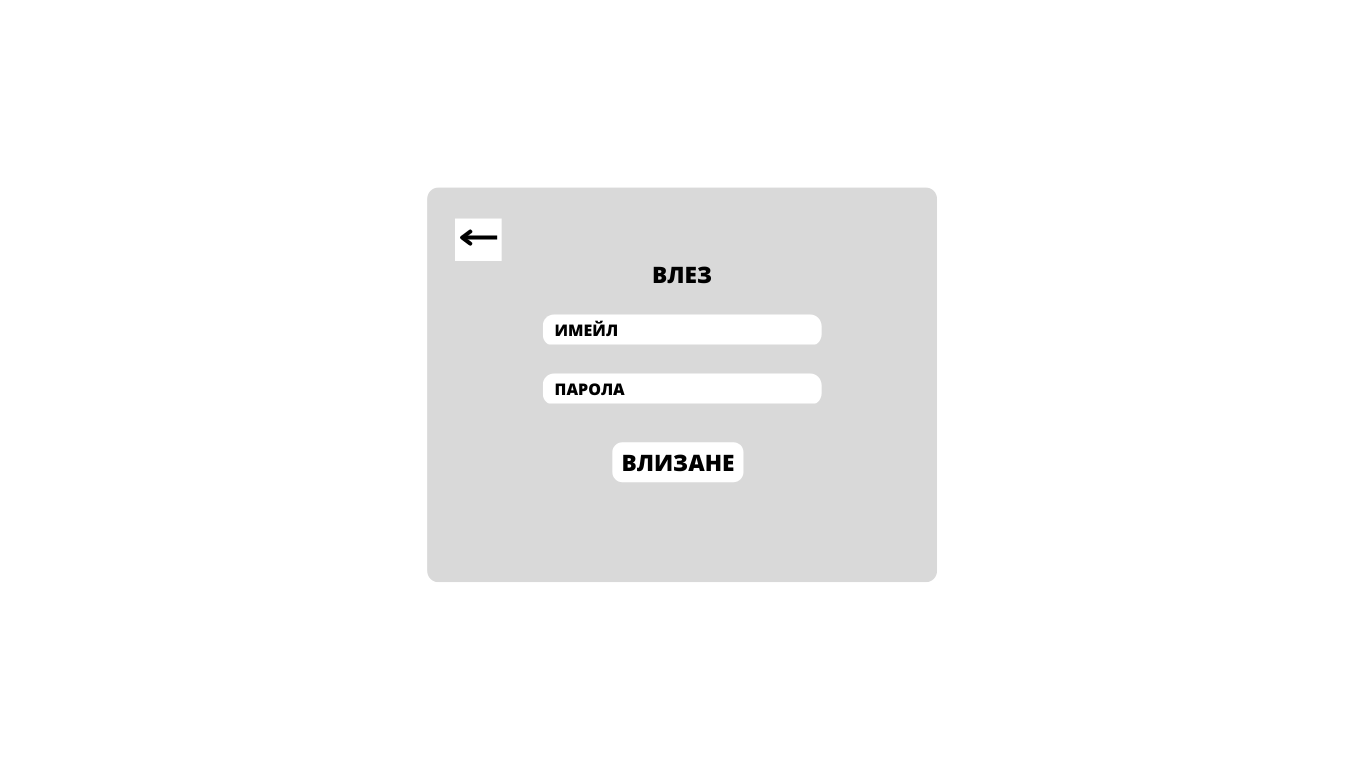
**Фиг.2.4 – *Contacts(contacts.php)***

Третата страница от проекта е ***contacts.php* (фиг.2.4).**Тя съдържа допълнителна информация като контакти за връзка на музея и т.н.



**Фиг.2.5 – *User Profile(userProfile.php)***

На **фиг.2.5** може да видите страницата ***userProfile.php***. В нея потребителят може да види своя профил и да излезе от акаунта си.

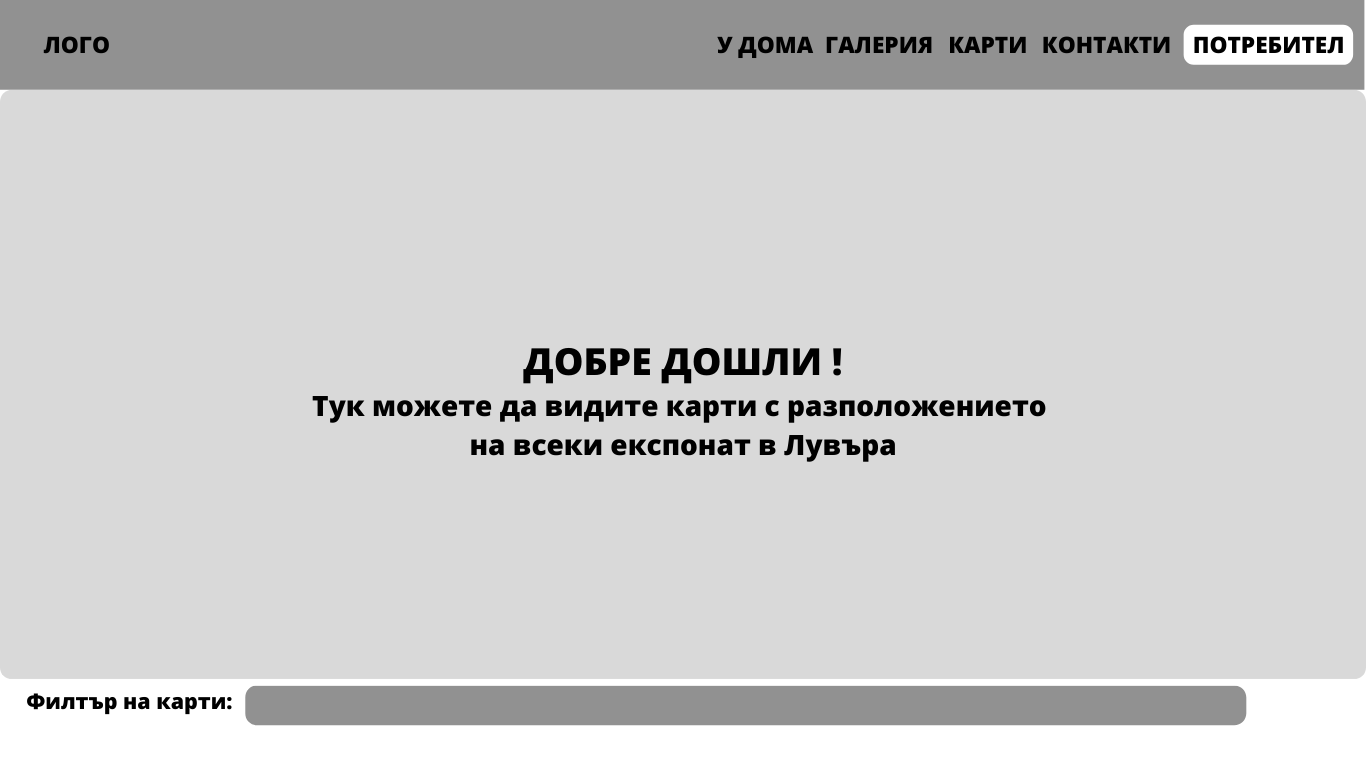


**Фиг.2.6 – *Register(register.php)***

На **фиг.2.6** може да видите страницата ***register.php.*** В нея потребителят може да влезе в профила си с имейл и парола. След което да натисне бутона за влизане, за да влезе в акаунта си.

**Фиг.2.7 – *Sign In(signIn.php)***

****На **фиг.2.7** може да видите страницата ***signIn.php.*** В нея потребителят може да се регистрира с потребителско име, имейл и парола. След което да натисне бутона за регистриране, за да се запази информацията за него в базата данни.



**Фиг.2.8 – *Cards (cards.php)***

На **фиг.2.8** може да видите страницата ***cards.php.*** В нея потребителят може да разгледа интерактивни карти с разчертано разположение на всички експонати по стаи.

# **Глава 3 Инженерно решение на поставената задача**

1. Описание на работата на кода

На фиг.3.1 е показан *PHP* код, който се грижи за регистрацията на потребител.

Ето разглеждане на това какво прави кодът:

1. *session\_start()* инициализира нова сесия или продължава съществуваща.
2. *$\_GET* се използва за извличане на стойностите на параметрите *"name", "email"* и *"password"* от *URL*.
3. Създава се масив, наречен *$registerErrors*, за да се съхранят всички грешки при регистрацията, които могат да възникнат.
4. С помощта на условния оператор *if* се проверява дали дължината на паролата е по-малка от 5 символа. Ако е така, се добавя грешка в масива *$registerErrors*.
5. Ако има грешки при регистрацията, те се съхраняват в масива *$registerErrors*, като се изтриват предишните грешки (ако има такива), използвайки функцията *unset().* След това грешките се записват в сесията, използвайки *$\_SESSION*. Накрая, потребителят се пренасочва към страницата *"signIn.php"* и се приключва изпълнението на скрипта с помощта на функцията *die().*
6. Ако няма грешки при регистрацията, тогава се изпълнява блока *else*. В него се осъществява връзката към базата данни с помощта на *PDO* *класа* на *PHP*. След това паролата се криптира с функцията *password\_hash().* Връзката към базата данни се използва, за да се въведат данните на новия потребител, използвайки *SQL* *заявка* *INSERT INTO*. След успешното добавяне на потребителя се пренасочва към страницата *"register.php"* и се приключва изпълнението на скрипта с помощта на функцията *die().*
7. Ако възникне грешка при връзката към базата данни, се извежда съобщение за грешка.

<?php

    session\_start();

    $name = $\_GET["name"];

    $email = $\_GET["email"];

    $password = $\_GET["password"];

    $registerErrors = array();

    if(strlen($password) < 5){

        $registerErrors[] = "Invalid password length!";

    }

    if(count($registerErrors) != 0){

        unset($\_SESSION["registerErrors"]);

        $\_SESSION["registerErrors"] = $registerErrors;

        header("Location: signIn.php");

        die();

    }else{

        $servername = "localhost";

        $dbusername = "root";

        $dbPassword = "";

        try {

            $conn = new PDO("mysql:host=$servername;dbname=thelouvre", $dbusername, $dbPassword);

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

            $hashed\_password = password\_hash($password, PASSWORD\_DEFAULT);

            $sql = "INSERT INTO user (name, `pass`, `email`, `userType`) VALUES (?,?,?,?)";

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

            $stmt->execute([$name, $hashed\_password, $email, 'user']);

            header("Location: register.php");

            die();

        } catch(PDOException $e) {

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

        }

    }

?>

Фиг.3.1 – PHP код за регистрация

на потребители

PHP кодът, показан на фиг.3.2, е отговорен за потвърждение на входа на потребителите в уеб приложение. Първоначално, сесията се стартира с функцията *session\_start().* След това се извличат имейл и парола от заявката *GET*, подадена от формата за вход. Сървърното име, потребителското име и паролата от базата данни се определят в началото на кода, а след това се прави опит за свързване с базата данни. Ако свързването бъде успешно, се извличат данните на всички потребители от таблицата *user*.

След това се извършва обхождане на всички потребители с цикъл *for*. За всеки потребител се проверява дали имейлът и паролата отговарят на тези, въведени от потребителя в формата за вход. Вместо да се сравнява паролата чрез оператора ==, този код използва функцията *password\_verify()*, която сравнява подадената парола с хешираната парола, която се съхранява в базата данни. Ако имейлът и паролата съвпадат, сесията се обновява с информацията за успешен вход и след това потребителят се пренасочва към началната страница на приложението.

Ако не бъде намерен потребител с подадените имейл и парола, се създава съобщение за грешка за вход и потребителят се пренасочва към страницата за регистрация.

<?php

    session\_start();

    $logMail = $\_GET["email"];

    $logPass = $\_GET["password"];

    $servername = "localhost";

    $dbUsername = "root";

    $dbPassword = "";

    $dbName = "thelouvre";

    try {

        $connection = new PDO("mysql:host=$servername;dbname=$dbName",$dbUsername,$dbPassword);

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

        $PDOstatement = $connection->prepare("SELECT \* FROM user");

        $PDOstatement -> execute();

        $PDOstatement -> setFetchMode(PDO::FETCH\_ASSOC);

        $data = $PDOstatement -> fetchAll();

        for($i = 0; $i < count($data); $i++){

            $currentRow = $data[$i];

            $id = $currentRow["userId"];

            $username = $currentRow["name"];

            $email = $currentRow["email"];

            $password = $currentRow["pass"];

            $userType = $currentRow["userType"];

            // if ($logMail == $email && $logPass == $password) {

            if(password\_verify($logPass, $password)){

                unset($\_SESSION["loginSuccess"]);

                $\_SESSION["loginSuccess"] = "loginSuccess";

                unset($\_SESSION["loggedUserID"]);

                $\_SESSION["loggedUserID"] = $id;

                unset($\_SESSION["username"]);

                $\_SESSION["loggedUser"] = $username;

                unset($\_SESSION["loggedUserRole"]);

                $\_SESSION["loggedUserRole"] = $userType;

                header("Location: ../home/home.php");

                die();

            }

        }

    } catch(PDOException $e) {

        echo "Connection failed: ". $e -> getMessAge();

    }

    unset($\_SESSION["loginError"]);

    $\_SESSION["loginError"] = "loginError";

    header("Location: register.php");

?>

Фиг.3.2 – PHP код за влизане

в акаунт

Кодът,показан на фиг.3.3, е за генериране на динамична уеб страница, която показва снимки на експонатите в музея. Първоначално се изгражда връзка към базата данни, след което се използва заявка за да се изберат редове от таблицата *"exhibits"* в зависимост от определен филтър, който се получава от формата, която се предава чрез *POST*. Ако не се предаде филтър, то всички редове ще бъдат взети. След това се използва *while* *цикъл*, за да се обходят всички избрани редове и да се изградят *HTML елементи* за визуализация на тези редове като картички с рисунки на експонатите. Има възможност да се избере един експонат и да се види повече информация за него, като се препраща към друга страница с форма, която се предава метод *GET*. Когато се избира експонат, неговото име и идентификационен номер се изпращат към другата страница във формата.

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "thelouvre";

$sql = "";

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

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

$filter = $\_POST['filter'];

            if ($filter == 'All') {

              $sql = "SELECT \* FROM exhibits";

            } else {

              $sql = "SELECT \* FROM exhibits WHERE Category='$filter'";

            }

          } else {

            $sql = "SELECT \* FROM exhibits";

          }

          $result = $conn->query($sql);

          if ($result->num\_rows > 0) {

            while ($row = $result->fetch\_assoc()) {

              echo '<div class="col-md-4">

                      <div class="thumbnail">

                        <form method="get" action="dynamicwebpage.php" class="mx-auto">

                          <img src="' . $row["FilePath"] . '" alt="' . $row["Name"] . ' " style="width:100%; height:250px">

                          <input style="display: none" name="drawingName" value="'. $row["Name"] .'">

                          <h2 class="card-text text-center my-2"><button class="buton border-0 bg-transparent text-white" type="submit" name="Id" value="'. $row["Id"] .'" class="nav-link text-dark"</button>' . $row["Name"] . '</h2>

                        </form>

                      </div>

                    </div>';

            }

          }

?>

**Фиг.3.3 –** Код за извеждане

на снимка на експонатите

Кодът на фиг.3.4. зарежда изложбите от базата данни и показва всяка една в отделна картичка, като при клик върху нея се препраща към подробности за конкретния експонат, извлича детайли за него от базата данни, като изобразява снимката му и неговото име, категория и място на произход.

<?php

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

$sql = "SELECT \* FROM exhibits WHERE Id=$drawingId";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

while ($row = $result->fetch\_assoc()) {

$\_SESSION["drawingName"] = $row["Name"];

echo '<div class="col">

<div class="card h-10 bg-dark">

<img class="mx-auto" style="width: 50%; height: 50%; align: center; margin-top: 100px; border-radius: 10px" src="'. $row["FilePath"] . '">

<div class="card-body text-center">

<form method="get" action="dynamicwebpage.php" class="mx-auto">

<h2 class="card-text text-center my-2"><button class="buton border-0 bg-transparent text-white" type="submit" name="Id" value="'. $row["Id"] .'" class="nav-link text-white"</button>' . $row["Name"] . '</h2>

<p class="d-inline-block me-2">Category: '. $row["Category"] .'</p>

<p class="d-inline-block">Country: '. $row["MadePlace"] .'</p>

</form>

</div>

</div>

</div>';

}

}

?>

**Фиг.3.4 –** Код за извеждане на

екпонатите в страницата ***dynamicwebpage.php***

*PHP* кодът на фиг.3.5 създава таблица с информация за потребители (admin panel) от базата данни *"thelouvre".* Таблицата съдържа четири колони*: "user Id", "name", "email" и "password".*

Кодът първо установява връзка с базата данни чрез *PDO обект*. След това проверява дали в *URL* има заявка за изтриване на потребител от таблицата и, ако има, извършва изтриването чрез *SQL* заявка към базата данни.

След това кодът извлича информацията от таблицата *"user"* чрез *SQL* заявка, която изпълнява и връща данните като масив. След това цикълът преминава през всички редове в масива и използва *HTML* таблица, за да ги покаже в уеб страницата.

В последната колона на таблицата, за всеки ред се показва бутон *"Delete",* който изпраща заявка за изтриване на потребител към същата страница с помощта на *GET* метода.

<?php

  $connection = new PDO('mysql:host=localhost;dbname=thelouvre', "root", "");

  $delete\_id = @$\_GET['delete\_id'];

  if ( $delete\_id ) {

    $query = $connection->prepare('DELETE FROM user WHERE userId = ?');

    $query->execute( [ $delete\_id ] );

  }

  $query = $connection->prepare('SELECT \* FROM  user ');

  $query->execute();

  $rows = $query->fetchAll();

?>

<div class="container table-responsive">

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

    <thead>

      <tr>

        <th>user Id</th>

        <th>name</th>

        <th>email</th>

        <th>password</th>

        <th>Action</th>

      </tr>

    </thead>

    <tbody>

    <?php

    foreach( $rows as $row ) {

      $row['userId'] = htmlspecialchars( $row['userId'], ENT\_QUOTES );

      $row['name'] = htmlspecialchars( $row['name'], ENT\_QUOTES );

      $row['email'] = htmlspecialchars( $row['email'], ENT\_QUOTES );

      $row['pass'] = htmlspecialchars( $row['pass'], ENT\_QUOTES );

    ?>

    <tr>

      <td><?= $row["userId"] ?></td>

      <td><?= $row["name"] ?></td>

      <td><?= $row["email"] ?></td>

      <td><?= $row["pass"] ?></td>

      <td>

        <a href="?delete\_id=<?= $row['userId'] ?>" class="btn btn-danger btn-sm">Delete</a>

      </td>

        </tr>

    <?php

        }

      ?>

    </tbody>

  </table>

</div>

Фиг.3.5 - PHP код за създаване на таблица

с информация за потребителите за страницата *admin.php*

Кодът на фиг.3.6 е във файл на езика за стилове *Sass* с името *"main.scss".* В него се дефинират променливи за цветове (например, *$primary, $secondary, $light, $dark* и *$link-color*), които се използват по-късно в темата на *Bootstrap*, която се импортира в края на файла.

Освен това, в кода се дефинират и някои допълнителни стилове за елементите на формулярите и за елемент с клас *"bgr-r-image".* В *HTML* документите, които използват този файл, ще се използва *Bootstrap* тема със специфични цветове и стилове за формуляри и елементи на страницата.

$primary: #212124;

$secondary: #c51f1f;

$light: #F5F7F8;

$dark: #212124;

$link-color: $light;

$link-decoration: none;

$theme-colors: (

  "primary":    $primary,

  "secondary":  $secondary,

  "light":      $light,

  "dark":       $dark

);

html {

  scroll-behavior: smooth;

}

$form-select-color: $light;

$form-select-bg: $dark;

@import "../node\_modules/bootstrap/scss/bootstrap";

select option {

  background-color: #212124 !important;

  color: rgb(216, 207, 207) !important;

}

.bgr-r-image {

  background-size: cover;

  background-position: center;

  background-repeat: no-repeat;

}

Фиг.3.6 – Код за добавяне

на класове от *Sass* файл

Кодът на фиг.3.7 представлява *JavaScript* скрипт, който се използва за създаване на интерактивна карта на музей Лувър в Париж. В него се използва библиотеката за *JavaScript* карти *Leaflet.js*. Картата се създава като се използва *L.map()* функцията за създаване на мапа с определени параметри, като например координатната система *(CRS),* минимално ниво на увеличение и други. След това се задават координатите на картинката, която ще се използва като слой на картата, като се използва *L.imageOverlay()* функцията. Картата включва маркери на различни изложби в музея, като за всяка изложба се задава съответния маркер и текст за *popup*.Картата се центрира и увеличава с *L.setView()* функцията, която задава координатите и нивото на увеличение на картата.

<script>

        const map = L.map('map', {

                crs: L.CRS.Simple,

                minZoom: -10

            });

            const yx = L.latLng;

            function xy(x, y) {

                if (Array.isArray(x)) {

                    return yx(x[1], x[0]);

                }

                return yx(y, x);

            }

            const bounds = [xy(0, 0), xy(320, 200)];

            const image = L.imageOverlay('../maps/images/louvre-museum-map level\_0.jpg', bounds).addTo(map);

            const ChristDetached = xy(118, 139);

            const StatueOfEbih = xy(175, 136);

            const ThePalaceOfDarius = xy(244, 136);

            const StatueOfMarcellus = xy(199, 29);

            const TheDyingSlave = xy(121, 58);

            const SculptureFromMaloIsland = xy(24, 29);

            const mChristDetached = L.marker(ChristDetached).addTo(map).bindPopup('<p class="text-center fw-bold m-0">Sculptures / France</p><div class="expo-container gray-expo-bgr text-black"><img src="../maps/images/Christ\_Detached.jpg"><b>Christ Detached from the Cross</b>201</div>');

            const mStatueOfEbih = L.marker(StatueOfEbih).addTo(map).bindPopup('<p class="text-center fw-bold m-0">Near Eastern<br>Antiquities</p><div class="expo-container yellow-expo-bgr text-black"><img src="../maps/images/Statue\_Ebih.jpg"><b>Statue of Ebih-Il</b>236</div>');

            const mThePalaceOfDarius = L.marker(ThePalaceOfDarius).addTo(map).bindPopup('<p class="text-center fw-bold m-0">Near Eastern<br>Antiquities</p><div class="expo-container yellow-expo-bgr text-black"><img src="../maps/images/palace\_Darius.jpg"><b>The Palace of Darius</b>308</div>');

            const mStatueOfMarcellus = L.marker(StatueOfMarcellus).addTo(map).bindPopup('<p class="text-center fw-bold m-0">Roman Antiquities</p><div class="expo-container blue-expo-bgr text-black"><img src="../maps/images/Statue\_Marcellus.jpg"><b>Statue of Marcellus</b>410</div>');

            const mTheDyingSlave = L.marker(TheDyingSlave).addTo(map).bindPopup('<p class="text-center fw-bold m-0">Sculptures /Europe</p><div class="expo-container gray-expo-bgr text-black"><img src="../maps/images/Dying\_Slave.jpg"><b>The Dying Slave</b>403</div>');

            const mSculptureFromMaloIsland = L.marker(SculptureFromMaloIsland).addTo(map).bindPopup('<p class="text-center fw-bold m-0">Arts of Africa,<br>Asia,Oceania and<br>the Americas</p><div class="expo-container text-black"><img src="../maps/images/Sculpture\_Malo\_Island.jpg"><b>Sculpture from<br>Malo Island</b>427</div>');

            map.setView(xy(170, 100), 1);

</script>

**Фиг.3.7** – Код за добавяне на карта

с ***Leaflet***

В главната папка на проекта се намират основните файлове, отговорни за работата на

уебсайта, а именно:

* home.php;
* gallery.php;
* dynamicwebpage.php;
* signin.php;
* register.php;
* logout.php;
* signInProcess.php;
* navigation.php;
* footer.php;
* userProfile.php;

1. Структура на базата данни

и таблиците в нея:

Базата данни на проекта се наричa **`thelouvre`** и съдържа следните таблици и полета във всяка от тях, както следва :

Име на таблицата: user

|  |  |  |
| --- | --- | --- |
| Име: | Тип: | Забележка: |
| userId | INT | Id на потребителя |
| name | VARCHAR | Име на потребителя |
| email | VARCHAR | Имейл на потребителя |
| pass | VARCHAR | Парола на потребителя |
| userType | VARCHAR | Роля на потребителя |

Име на таблицата: place

|  |  |  |
| --- | --- | --- |
| Име: | Тип: | Забележка: |
| Id | INT | Id на мястото |
| Room | VARCHAR | Стая на експонат |
| Floor | VARCHAR | Етаж на експонат |
| ExhibitsId | INT | Id на експонат |

Име на таблицата: exhibits

|  |  |  |
| --- | --- | --- |
| Име: | Тип: | Забележка: |
| Id | INT | Id на експонат |
| Name | VARCHAR | Име на експонат |
| Category | VARCHAR | Категория на експонатите |
| Place | VARCHAR | Id на мястото |
| MadePlace | VARCHAR | Страна на създаване |
| FilePath | VARCHAR | Пътят към снимка на експоната |

След като сте инсталирали XAMPP, следваме стъпките по долу :

1. Отворете браузър и отидете на URL <http://localhost/phpmyadmin/>.
2. След това щракнете върху раздела бази данни.
3. Създайте база данни с име “thelouvre” и след това щракнете върху раздела за импортиране.
4. Щракнете върху преглед на файла и изберете файл”thelouvre.sql”.
5. Щракнете върху Go.

## **SQL CREATE заявки за**

## **създаване и попълване на таблици**

Заявка за създаването на таблицата ***user***:

**CREATE** **TABLE** **user**(

userId **INT** **PRIMARY** **KEY** **AUTO\_INCREMENT**,

**name** **VARCHAR**(255),

email **VARCHAR**(255),

pass **VARCHAR**(255),

userType **VARCHAR**(255)

);

Примерна заявка за въвеждане на потребител в таблицата user:

**INSERT** **INTO** `user` (`name`, `email`, `pass`, `userType`) **VALUES**

('Потребител', 'user@mail.bg', '1234567890', 'user');

Заявка за създаването на таблицата ***exhibits***:

**CREATE** **TABLE** **user**(

id **INT** **PRIMARY** **KEY** **AUTO\_INCREMENT**,

**Name** **VARCHAR**(255),

Category **VARCHAR**(255),

Place **VARCHAR**(255),

madePlace **VARCHAR**(255),

FilePath **VARCHAR**(255)

);

Примерна заявка за въвеждане на потребител в таблицата ***exhibits***:

**INSERT** **INTO** `exhibits ` (`Name`, `Category `, `Place `, `MadePlace`, `FilePath`) **VALUES** ('The Mona Lisa', 'Portrait', '1', 'Italy', '../images/mona\_lisa.jpg');

Заявка за създаването на таблицата ***exhibits***:

**CREATE** **TABLE** **user**(

id **INT** **PRIMARY** **KEY** **AUTO\_INCREMENT**,

**Name** **VARCHAR**(255),

Category **VARCHAR**(255),

Place **VARCHAR**(255),

madePlace **VARCHAR**(255),

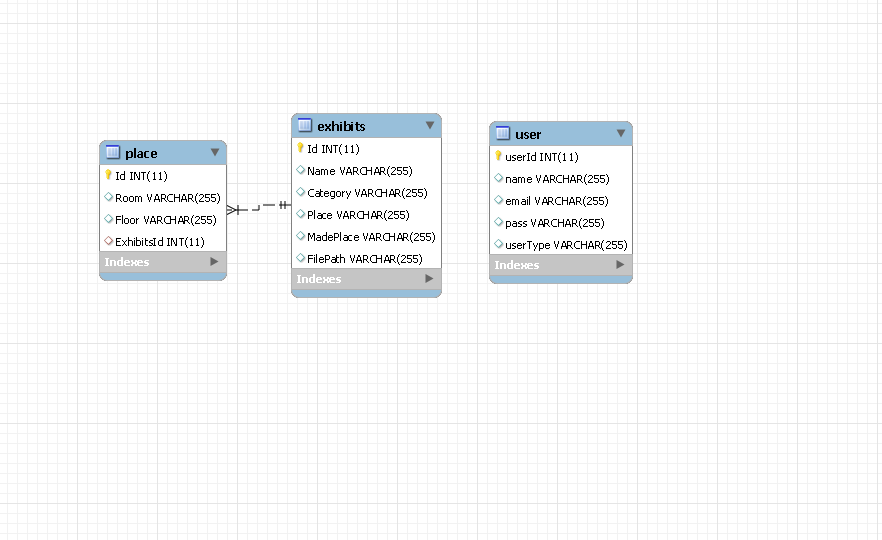
FilePath **VARCHAR**(255)

);

Примерна заявка за въвеждане на потребител в таблицата ***exhibits***:

**INSERT** **INTO** `place ` (`Id`, `Room`, `Floor `, `ExhibitsId `) **VALUES** ('1', '1', '1', '1');

На ***фиг.3.7*** е показана ***ER диаграма*** на таблиците в базата и свързванията на информацията в тях.



**Фиг.3.7** – ***ER диаграма***

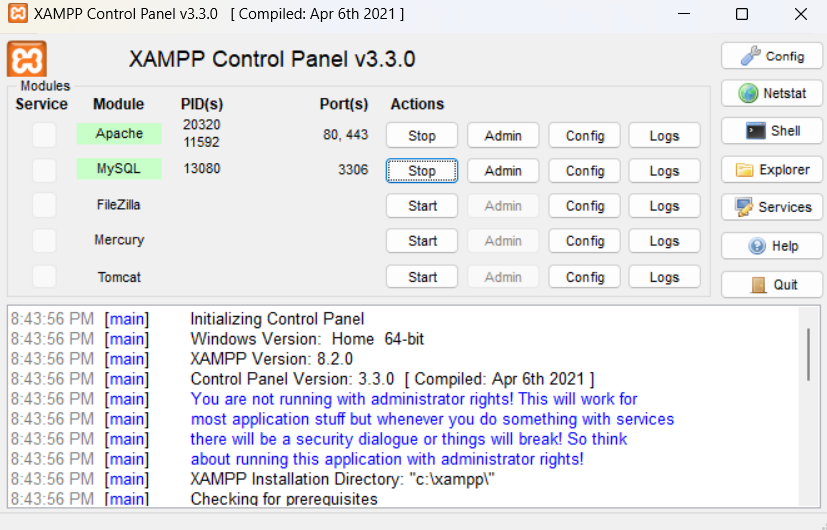
на базата данни

# **Глава 4 Анализ на получените резултати, приложимост и изводи**

## 

## **Работа на уебсайта**

За стартиране на уебсайта първоначално трябва да се стартират всички нужни сървъри през интерфейса на ***XAMPP***, а те са **фиг.4.1** : ***Apache Web Server*** и ***MySQL Database***.

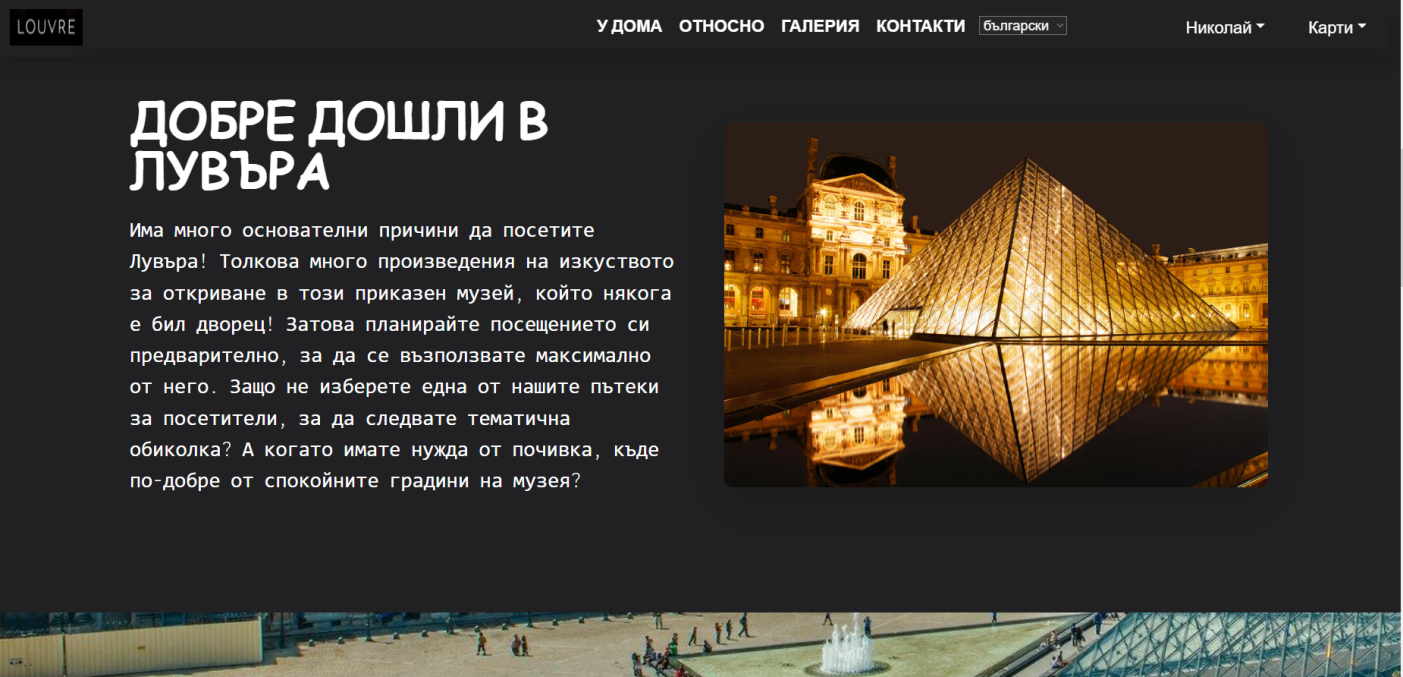


**Фиг.4.1** – Интерфейс на***XAMPP Control Panel***

След стартиране на уебсайта се показва следни екран(**фиг.4.2**). Когато дадем по-надолу ни излиза екрана от **фиг.4.3**:



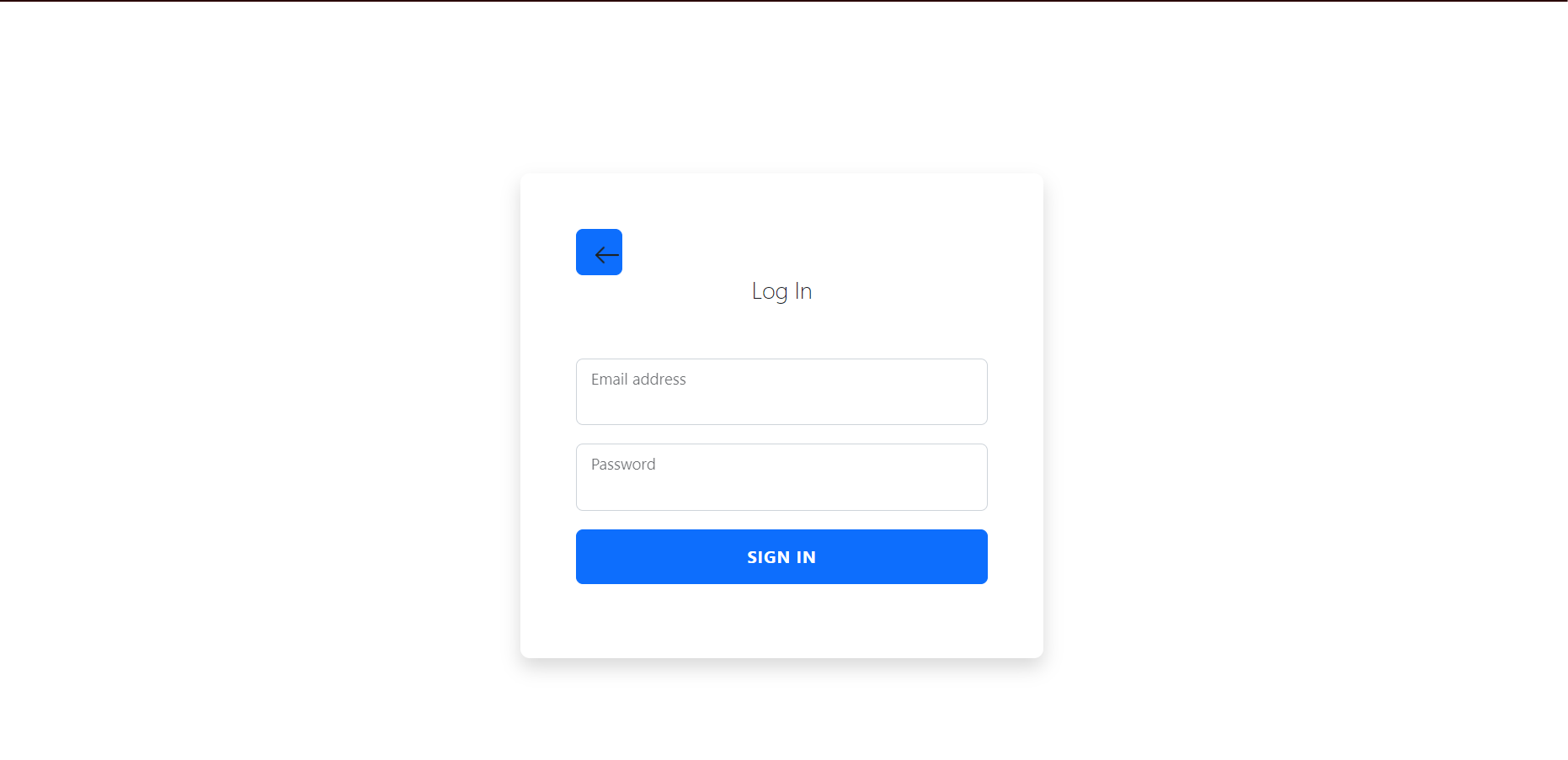
**Фиг.4.2** – Начален екран при стартиране



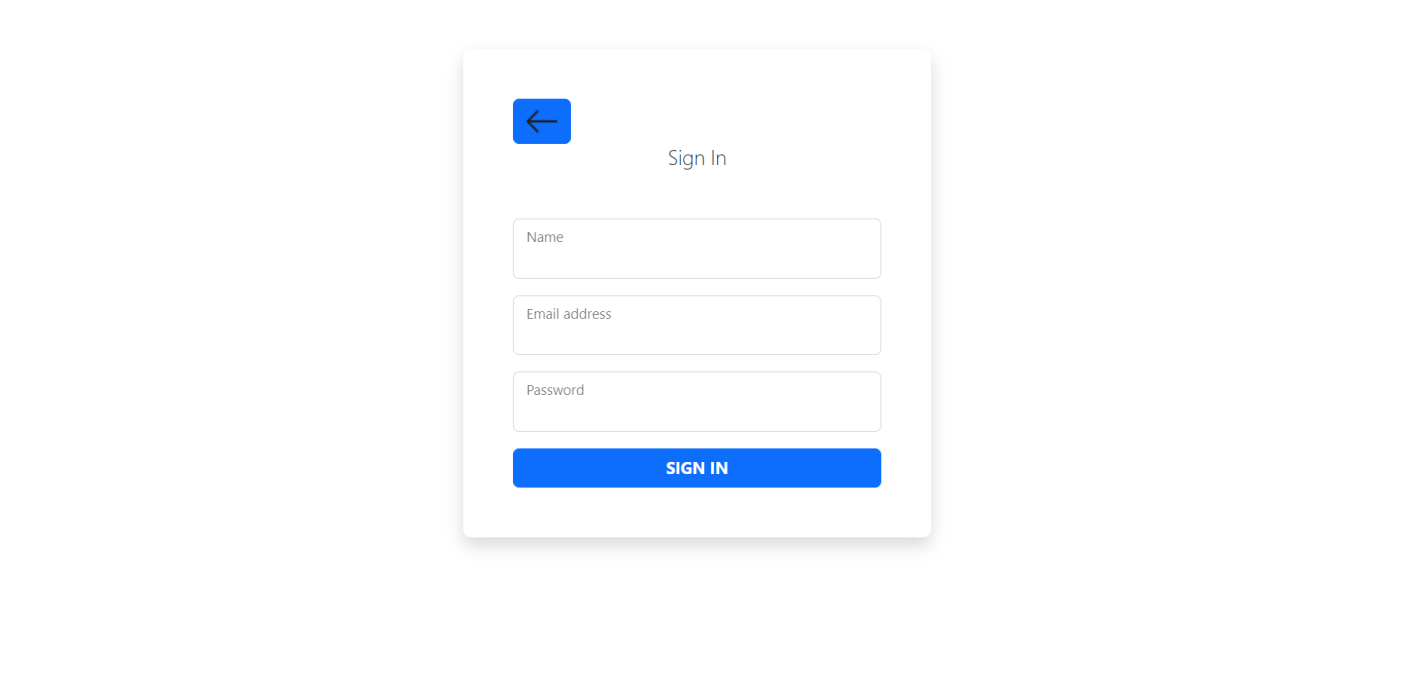
**Фиг.4.3** – Начален екран(Относно)

Това е началния екран. Отгоре има възможност за (***LOGIN*** и ***SIGNIN***). За вписване се влиза в ***LOGIN***. Ако потребителят е нов ***SINGIN*** бутонът му дава възможност за създаване на акаунт, а след това трябва да се впише.

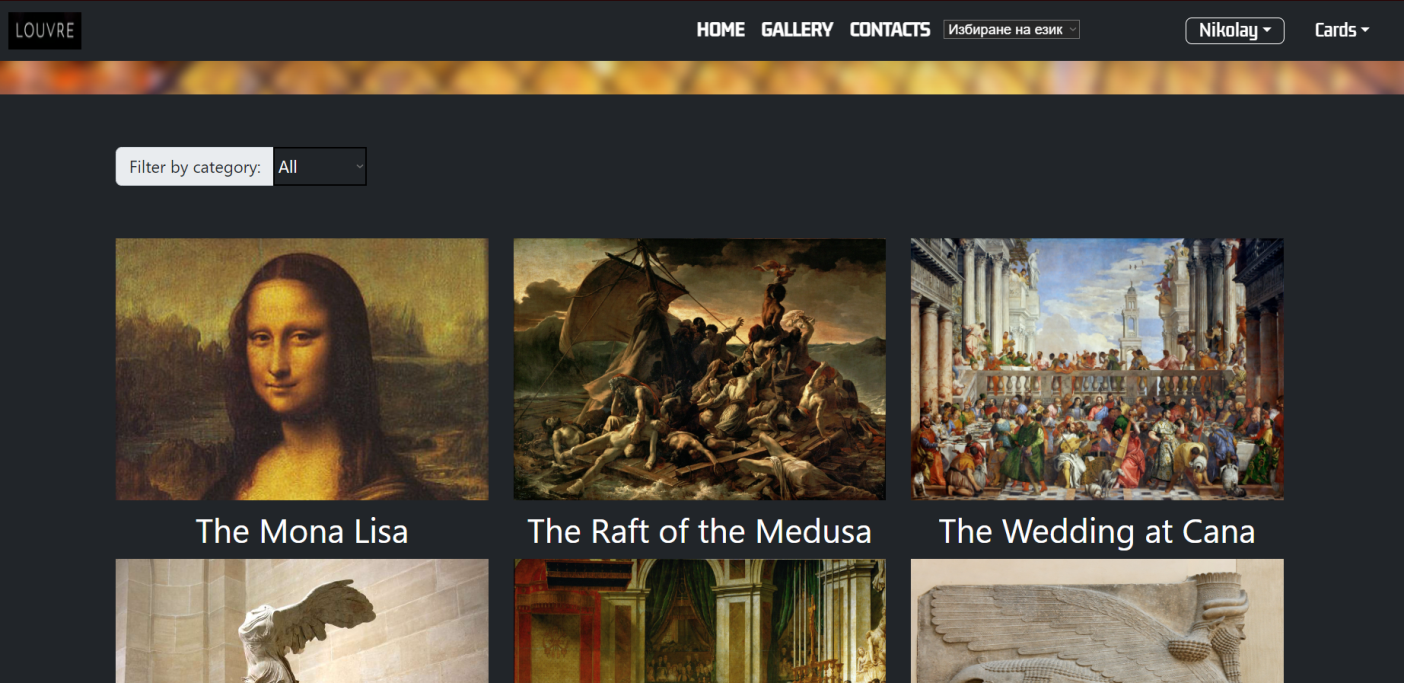
Така изглеждат ***LOG IN*** формата(**фиг.4.4**) и ***SIGN IN*** формата(**фиг.4.5**):



**Фиг.4.4** *–* ***LOG IN***

**Фиг.4.5** – ***SIGN IN***

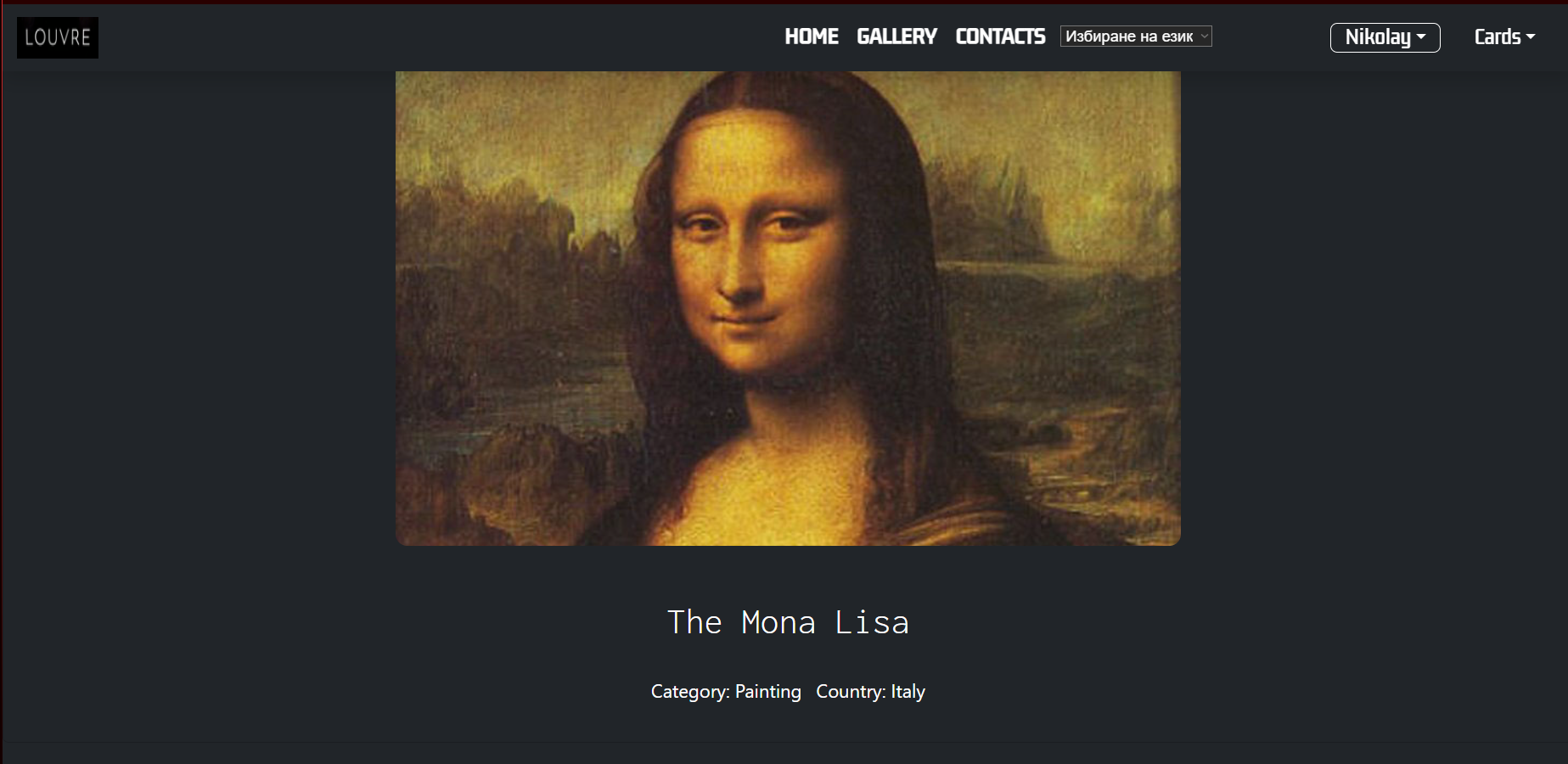
След влизане в акаунта отново се показва началния екран, като потребителят може да влезе в галерията и да разгледа експонатите(**фиг.4.6**):



**Фиг.4.6** – Галерия със снимки

на експонатите

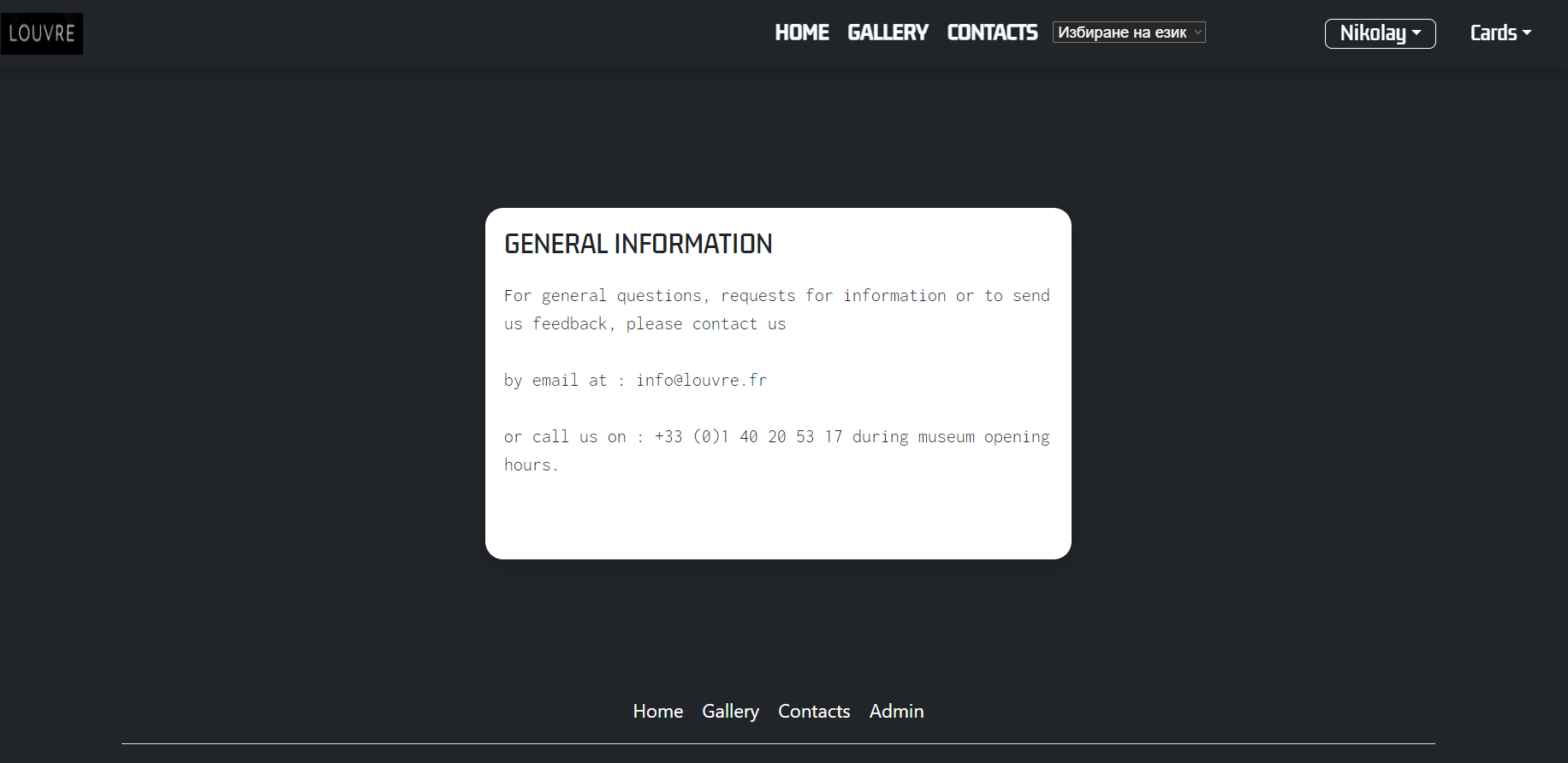
При натискането на снимката на някой от експонатите се отваря ***dynamic web page***(**фиг.4.7**), чрез които се отваря снимката в нов прозорец:



**Фиг.4.7** – Страница с информация

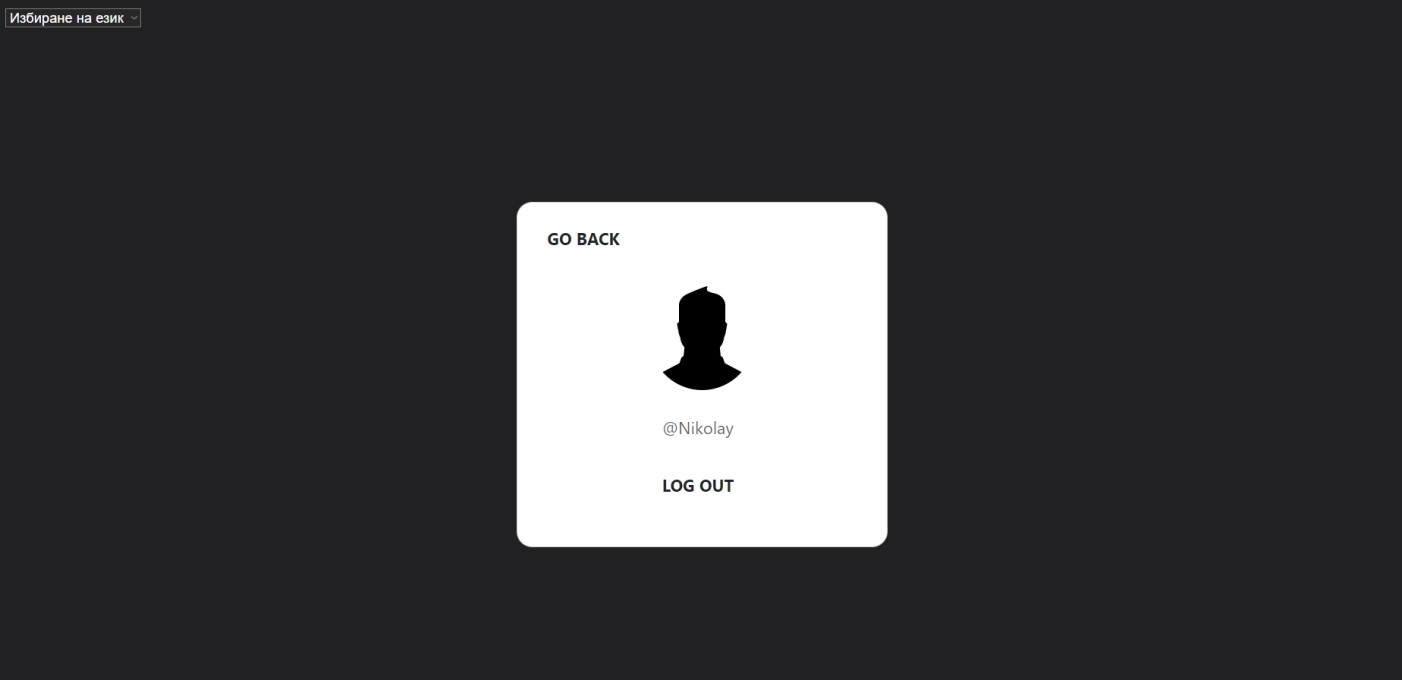
за експоната

В страницата ***contacts.php*** потребителят може да види информация като номер и имейл на музея – Лувъра. На **фиг.4.8** е показан интерфейса на тази страница.

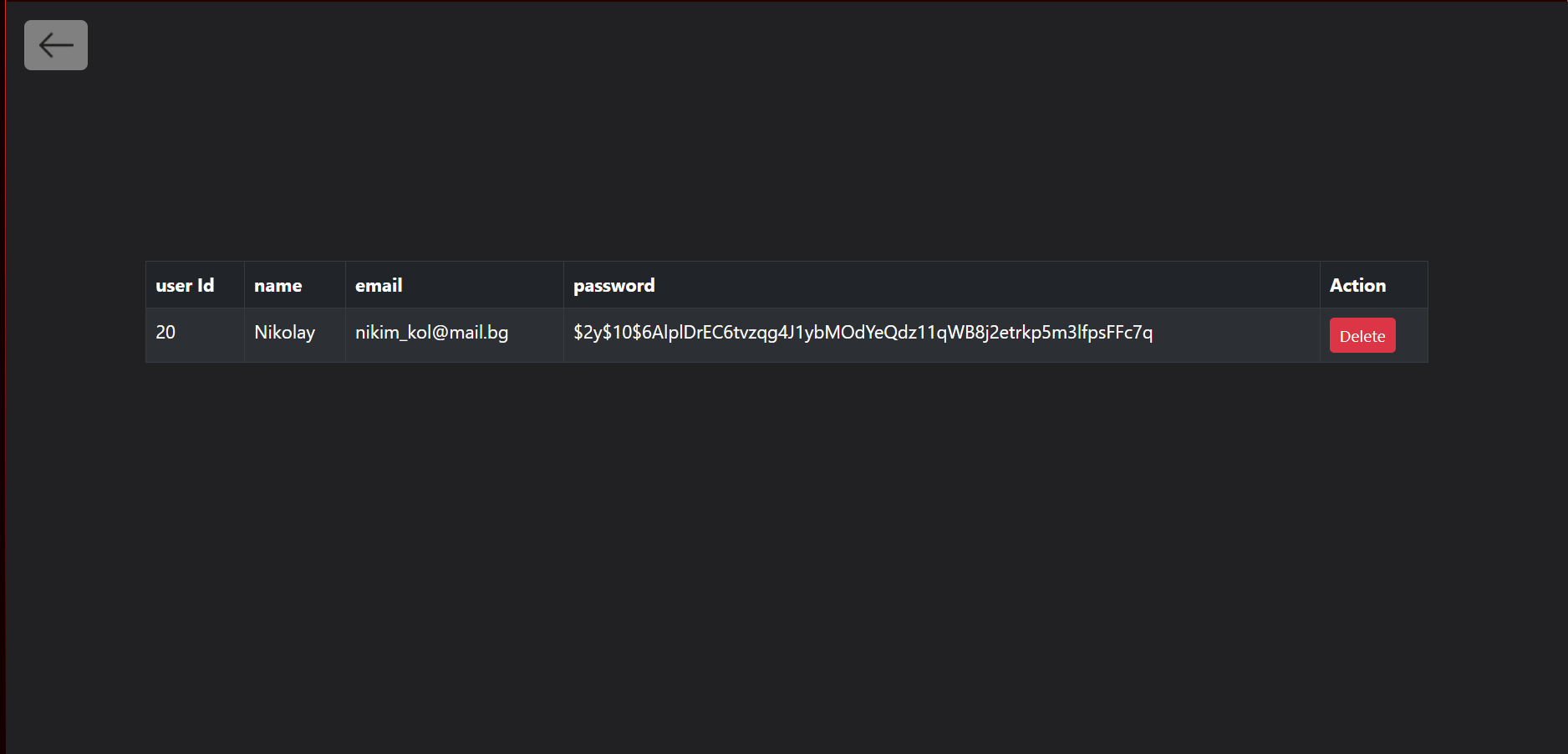


**Фиг.4.8** – Интерфейс на contacts.php

Потребителят може да избере и да влезе в своя профил (**фиг.4.9**) и да излезе от него от dropdown менюто с името му. Ако потребителят има admin права може да види и всички останали потребители в админ панела(**фиг.4.10**) и да изтрие някого.



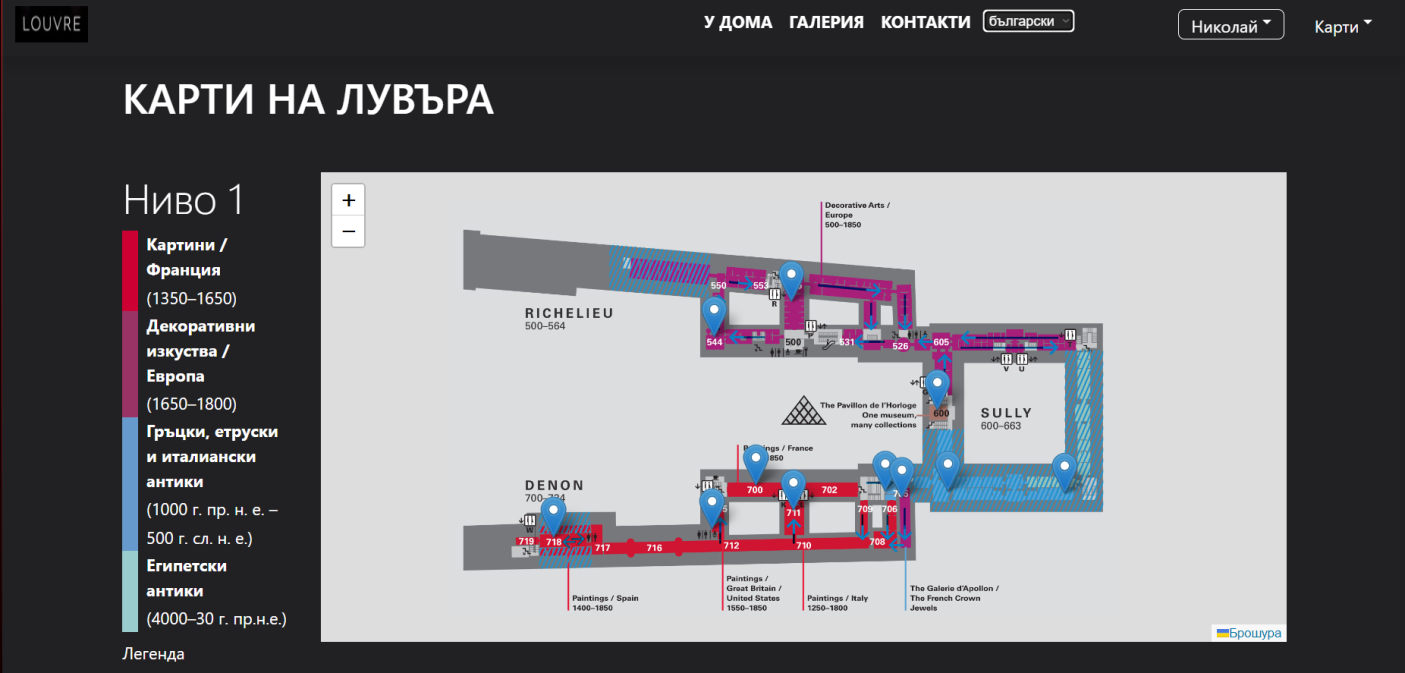
**Фиг.4.9** – User Profile



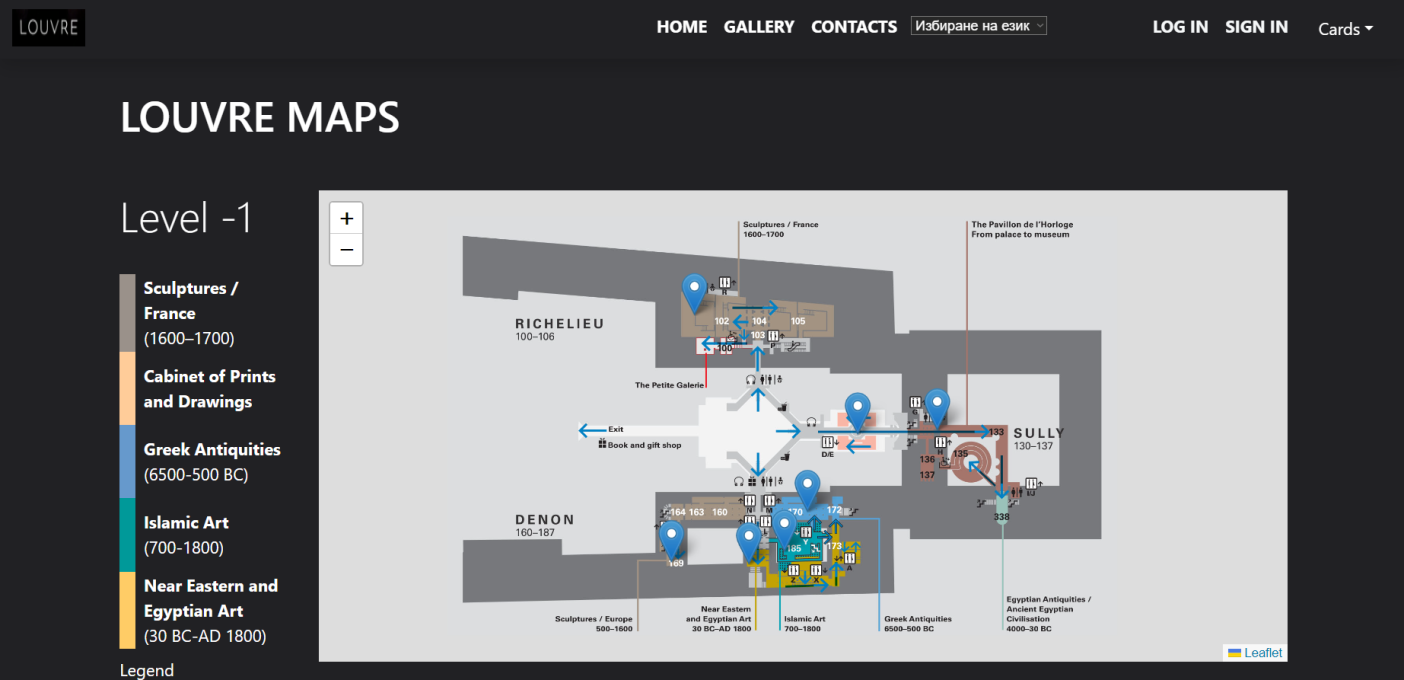
**Фиг.4.10** – Admin Panel

След това всеки потребител може да разгледа карти на Лувъра през dropdown менюто карти.(**фиг.4.11; фиг.4.12; фиг.4.13; фиг.4.14**).

**Фиг.4.11** – Карта(Ниво 0)



**Фиг.4.12** – Карта(Ниво 1)



**Фиг.4.12** – Карта(Ниво -1)



**Фиг.4.12** – Карта(Ниво 2)

## **Заключение**

Настоящата дипломна работа има за цел създаването на сайт, което служи за информация на хора, които желаят да посетят музея Лувъра. Създадено е със средствата на маркиращият език ***HTML (HyperText Markup Language)*** и стилизиращият език ***CSS (Cascading Style Sheet).***

Работата с приложението е изключително улеснена и интуитивна. Подходящо са избрани всички необходими технологии, за да се получи едно стабилно и бързо работещо проложение.

Бъдещи идеи за развитие на продукта са функционалности като:

1. Възможност за промяна на парола.
2. Възможност за промяна на потребителско име
3. Възможност за оставяне на ревю.

# **Използвана литература**

[1] “[HTML5 – Hypertext Markup Language – 5.0](http://www.w3.org/2014/10/html5-rec.html.en).”-Спецификация на HTML

[2] “[Cascading Style Sheets](http://www.w3.org/Style/CSS/#browsers).”- Спецификация на CSS

[3]” [Bootstrap 5.2.3"](https://blog.getbootstrap.com/2022/11/22/bootstrap-5-2-3/).” – Спецификация на Bootstrap

[4]”[*nodejs/node*](https://github.com/nodejs/node/blob/master/BUILDING.md)” – Спецификация на Node JS

[5]” [PHP 8.2.1 Released"](https://www.php.net/archive/2023.php#2023-01-05-2).” – Спецификация на PHP

[6]”[*XAMPP"*](https://sourceforge.net/projects/xampp/)” – Спецификация на XAMPP

[7]”["Release v1.9.3"](https://github.com/Leaflet/Leaflet/releases/tag/v1.9.3).” – Спецификация на LeafLet

[8]” [Sass: Syntactically Awesome Style Sheets](http://sass-lang.com/about.html)” – Спецификация на SASS

# Списък с използваните означения и съкращения

HTML – Hyper Text Markup Language

CSS - Cascading Style Sheets

PHP – Personal Home Page

SQL - Structured Query Language

HTTPS - Hypertext Transfer Protocol Secure

URL - Uniform Resource Locator

SASS - Syntactically Awesome Style Sheets

IDE - Integrated Development Environment

При форматирането на текста се спазват следните параметри:

* Обем на дипломния проект – минимум 30 страници
* Формат: А4; Брой редове в стр.: 30; Брой на знаците: 60 знака в ред
* Общ брой на знаците в 1 стр.: 1800 – 2000 знака
* Шрифт: Times New Roman, 12, разстояние между редовете 1,5