

Střední průmyslová škola Třebíč

Maturitní práce

Webová aplikace pro truhlářství

Profilová část maturitní zkoušky

Studijní obor: Informační technologie

Třída: ITA4

Školní rok: 2023/2024 Vojtěch Mastný

Zadání práce

Webová aplikace bude obsahovat úvodní stránku, informace o firmě, služby, galerii, kontakty s mapou, formulář pro dotazy, vzorové kalkulace ceny, propojení se sociálními sítěmi a log-in systém. Webová aplikace bude využívat jazyky HTML, CSS, MySQL, PHP. Web bude dále zahrnovat základní zabezpečení, dále možnost přidávání obsahu pro administrátora a změny v kalkulaci.

ABSTRAKT

Webová stránka pro truhlářství, která obsahuje blog příspěvků, fotogalerii, vzorové kalkulace, kontakty a log-in systém. Webová stránka je vytvořená za pomocí jazyků HTML, CSS a PHP.

KLÍČOVÁ SLOVA

maturitní práce, webová stránka, truhlářství

ABSTRACT

Website for the joinery industry that includes blog posts, photo gallery, sample quotes, contacts and a log-in system. The website is created using HTML, CSS and PHP.

KEYWORDS

graduation thesis, website, joinery

PODĚKOVÁNÍ

Děkuji Ing. Ondřeji Stejskalovi za vedení při tvoření ročníkového práce. Dále Natálii Pistovčákové za rady při tvoření dokumentace k ročníkové práci.

V Třebíči dne 27. března 2024 podpis autora

PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl v ní všechny prameny, literaturu a ostatní zdroje, které jsem použil.

V Třebíči dne 27. března 2024

podpis autora

Obsah

[Úvod 6](#_Toc162019369)

[1 Teoretická část 7](#_Toc162019370)

[1.1 Použité technologie 7](#_Toc162019371)

[1.1.1 HTML 7](#_Toc162019372)

[1.1.2 CSS 8](#_Toc162019373)

[1.1.3 PHP 9](#_Toc162019374)

[1.1.4 PhpMyAdmin 11](#_Toc162019375)

[1.1.5 MySQL 11](#_Toc162019376)

[1.2 Hrozby 12](#_Toc162019377)

[1.2.1 SQL Injection 12](#_Toc162019378)

[1.2.2 Cross-Site Scripting 14](#_Toc162019379)

[1.2.3 Brute Force útok 15](#_Toc162019380)

[1.2.4 DoS útok 17](#_Toc162019381)

[2 Praktická část část 19](#_Toc162019382)

[2.1 Domovská stránka (index.php) 19](#_Toc162019383)

[2.2 Styl domovské obrazovky (style.css) Chyba! Záložka není definována.](#_Toc162019384)

[2.3 Změna obrázků a hamburger menu (script.js) 21](#_Toc162019385)

[2.4 Blog (blog.php) 22](#_Toc162019386)

[2.4.1 Úprava příspěvku (edit\_post.php) **Chyba! Záložka není definována.**](#_Toc162019387)

[Závěr 39](#_Toc162019388)

[Seznam použitých zdrojů 40](#_Toc162019389)

[Seznam použitých symbolů a zkratek 42](#_Toc162019390)

[Seznam obrázků 43](#_Toc162019391)

[Seznam tabulek Chyba! Záložka není definována.](#_Toc162019392)

[Seznam příloh Chyba! Záložka není definována.](#_Toc162019393)

Úvod

Tato maturitní práce se zabývá vývojem webové aplikace na míru pro truhlářství, která byla zadána jako zakázkový projekt. Volba tohoto tématu vychází z mého hlubokého zájmu o technologie a jejich praktické využití, a to i přes počáteční nedostatek zkušeností s vývojem webových stránek. Webová aplikace, kterou vytvořím, bude navržena tak, aby vyhovovala specifickým potřebám zákazníka a jejím cílem je zefektivnit komunikaci s klienty a prezentaci truhlářských výrobků.

V následujících kapitolách budu systematicky popisovat jednotlivé fáze vývoje, technologie použité při tvorbě aplikace a věnovat se analýze potřeb truhlářské firmy a  způsobu, jakým byla aplikace přizpůsobena těmto požadavkům.

Webová aplikace vytváří komplexní prostředí prostřednictvím různých jazyků a technologií. Základní stavební kameny této aplikace budou spočívat v použití jazyku HTML, který slouží k definici struktury celého webu. Důraz bude kladen na vytvoření kvalitního uživatelského rozhraní a vizuálního designu, a to díky jazyku CSS, který umožňuje elegantní stylování obsahu.

Přihlašovací proces, klíčový pro interakci uživatelů s aplikací, bude řešen s využitím jazyka PHP. Tímto způsobem se zajistí bezpečné a spolehlivé zpracování přihlašovacích údajů uživatelů. Tato webová aplikace bude založena na solidním technologickém základu a umožňuje uživatelům efektivní a intuitivní interakci s obsahem díky kombinaci těchto jazyků a moderních technologií.

# Teoretická část

V této části se zaměřím na popis jednotlivých technických a teoretických prostředků využívaných v mé práci.

## Použité technologie

V této části popíšu jednotlivé technologie pro tvoření webových stránek

### HTML

HyperText Markup Language (HTML), představuje jazyk označování, který slouží   
k vytváření dokumentů obsahujících hypertextové odkazy a pokročilé formátování. Funkce HTML jsou rozmanité a umožňují manipulaci s obsahem webových stránek. [1]

Základními stavebními kameny HTML jsou tagy. Tyto tagy rozdělujeme na párové  
a nepárové. Párové tagy se využívají např. pro nadpisy, odkazy, odstavce apod.

*<h1>Lorem ipsum dolor</h1>*

<h1> až <h6>: Tagy pro nadpisy, kde `<h1>` je nejvyšší úrovně a `<h6>` je nejnižší.

Zato nepárové používáme např. pro vložení obrázků, nebo zalomení řádku.

*<img src="pictures/spst.png" width="450" height="320" alt="SPŠT logo">*

*<img>*:vkládá obrázek do stránky.

*`src="pictures/spst.png"`*:Cesta k obrázku.

*`width="450"`*:Šířka obrázku (450 pixelů).

*`height="320"`*: Výška obrázku (320 pixelů).

*`alt="SPŠT logo"*`: Alternativní text pro případ, že obrázek není načten (SPŠT logo).

Struktura základní HTML stránky zahrnuje několik klíčových částí:

<!DOCTYPE html>: Tato deklarace definuje verzi HTML, kterou dokument používá.

<html>: Tento kořenový element obaluje celý obsah HTML dokumentu.

<head>: Sekce `<head>` obsahuje metadata o dokumentu, jako například kódování znaků, náhledové informace a odkazy na externí styly.

<meta charset="UTF-8">: Určuje použité kódování (UTF-8) pro zpracování znaků.

<meta name="viewport" content="width=device-width, initial-scale=1.0">: Nastavuje zobrazení na zařízeních s různými šířkami obrazovky.

<title>: Určuje název stránky, který se zobrazuje v záhlaví nebo na kartě prohlížeče.

<body>: Tato část obsahuje samotný obsah dokumentu, jako text, obrázky, odkazy, atd.

<p>: Tag pro odstavce textu.

Vytváření formulářů je další klíčovou funkcí HTML, umožňující uživatelům např. odeslat e-maily, reagovat na dotazníky nebo provádět objednávky z online katalogu. HTML rovněž umožňuje vytvářet tabulky, které poskytují větší kontrolu   
nad formátováním dokumentů a uspořádáním obsahu. Definování barev pozadí   
pro dokument, tabulku, řádek nebo buňku přispívá k celkové vizuální atraktivitě. [1]

Odkazy, další klíčový prvek HTML, umožňují navigaci mezi sekcemi dokumentu, odkazy na jiné stránky a integrovaní audio a video souborů. [1]

HTML pouze slouží jako základní struktura webové stránky, pro vizuální styl se dále využívá jazyk CSS, viz. Kapitola 1.1.2

### CSS

Cascading Style Sheets (CSS) představuje jazyk, který stanovuje základní vizuální podobu webových stránek a uspořádání jejich prvků. Jeho použití spočívá ve spolupráci s HTML, XHTML nebo XML, kde detailně popisuje, jak by měly být prezentovány jednotlivé složky. Komponenty, jež nejsou specifikovány pomocí CSS, zůstávají v původní podobě. [2]

Struktura webového prostoru, respektive dokumentu, vytvořená pomocí HTML, zůstává oddělena od jeho vizuální podoby. Tímto způsobem vzniká možnost pro jednodušší a sofistikovanější úpravy, což přispívá k rychlejšímu zobrazení obsahu stránek. [2]

CSS můžeme propojit s HTML souborem 3 způsoby.

Zápisem do hlavičky HTML souboru.

<style>

p { color: green; }

</style>

Zápisem přímo do konkrétního prvku

<p style="color: green;"> Lorem ipsum dolor </p>

‘style="color: green;"‘: Změnění barvy písma na zelené.

Propojení s externím CSS souborem.

<head>

<link rel="stylesheet" type="text/css" href="style.css" />

</head>

### PHP

Hypertext Preprocessor (PHP) je serverový skriptovací jazyk, určený pro specifické potřeby webových stránek. Na rozdíl od jiných jazyků, jako je například JavaScript, které běží na straně klienta, PHP provádí všechny své operace na straně serveru. Výsledkem jsou sestavené stránky ve formátu HTML, které jsou poté zobrazeny uživateli. [3]

Jedním z klíčových rysů PHP je jeho charakter jako Open Source technologie, což znamená, že je k dispozici zdarma a může být distribuováno bez omezení. [3]

PHP se vyznačuje schopností být nasazeno napříč rozličnými platformami   
a serverovými konfiguracemi, což z něj činí univerzálně použitelný nástroj. Tato nezávislost na specifických prostředích nabízí vysokou míru adaptability, což umožňuje jeho efektivní implementaci v širokém spektru vývojářských projektů. Tato otevřenost a flexibilita PHP zajišťuje, že se může snadno integrovat do různých serverových nastavení, což rozšiřuje možnosti jeho využití. [3]

PHP má 3 rozdílné typy proměnných podle rozsahu:

Globální - je deklarována mimo tělo funkcí nebo bloků kódu a je přístupná ze všech částí programu.

$globalniPromenna = 10;

function pristupKGlobalniPromenne() {

global $globalniPromenna;

echo $globalniPromenna;

}

Lokální - je deklarována uvnitř těla funkce nebo bloku kódu a je přístupná pouze   
z tohoto konkrétního kontextu.

function lokalniPromenna() {

$lokalniPromenna = 5;

echo $lokalniPromenna;

}

Statická - uchovává svou hodnotu i po opuštění kontextu, kde byla vytvořena. Při opětovném volání funkce si pamatuje svůj stav.

function statickaPromenna() {

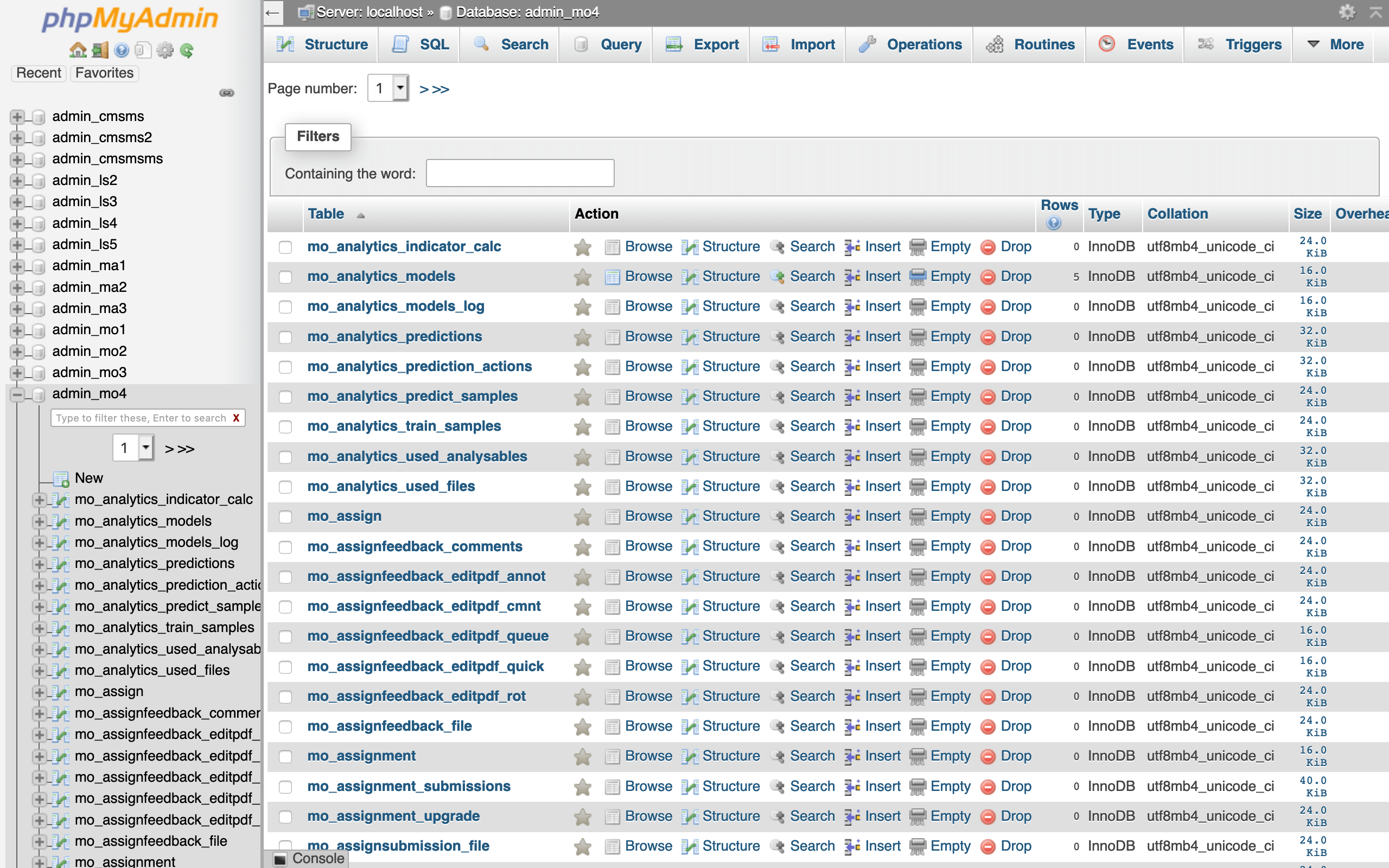
static $pocitadlo = 0;

$pocitadlo++;

echo $pocitadlo;

}

### PhpMyAdmin

phpMyAdmin je open-source aplikace napsaná v PHP, která poskytuje grafické rozhraní pro správu databází MySQL a MariaDB. PhpMyAdmin zjednodušuje množství úloh spojených s údržbou a zlepšením databází tím, že poskytuje jednoduché grafické rozhraní (GUI) pro vykonávání obvyklých úkonů, jako jsou nastavení   
Nabízí uživatelům možnost provádět tyto činnosti efektivněji, aniž by bylo potřeba podrobně rozumět SQL jazyku.

Obr. . phpMyAdmin prostředí [6]

Výsledkem je, že správci a vývojáři mohou bez obtíží manipulovat s databázemi, což vede k lepší správě, konzistenci dat a vylepšení výkonnosti databázových systémů. [4]

### MySQL

The Structured Query Language (SQL) je jazyk navržený pro práci s relačními databázemi, který umožňuje uživatelům vytvářet databáze, odstraňovat záznamy, provádět dotazy a modifikovat data. MySQL, který operuje jako systém pro správu relačních databází (RDBMS), hraje důležitou roli v chodu webových stránek a online aplikací díky své schopnosti efektivně zpracovávat a ukládat velké množství informací. Díky svým přednostem je MySQL často vybráno pro ukládání   
a vyhledávání dat, včetně osobních informací uživatelů, jako jsou uživatelská jména, šifrovaná hesla a data registrace. [5]

## Hrozby

Je zde mnoho hrozeb, které mohou negativně ovlivnit webové stránky. Proto v této kapitole některé představím a jak se proti nim bránit.

### SQL Injection

SQL injection (SQLi) je často využívaná technika útoku, která zahrnuje vložení škodlivého SQL kódu do vstupních polí webových formulářů nebo URL adres za účelem neoprávněné manipulace s databází a získání citlivých informací, které by jinak neměly být přístupné. Tento útok umožňuje útočníkovi získat přístup k citlivým údajům, jako jsou například hesla, osobní informace uživatelů nebo interní firemní data. [7]

Dopad SQLi na obchodní sféru může být značný. Realizovaný útok může způsobit neoprávněné prohlížení uživatelských seznamů, vymazání celých tabulek a dokonce   
i získání administrátorských oprávnění k databázi. [7]

Příklad SQLI: [8] {

*uName = getRequestString("username");*

*uPass = getRequestString("userpassword");*

*sql = 'SELECT \* FROM Users WHERE Name ="' + uName + '" AND Pass ="' + uPass + '"'*

*uName = getRequestString("username")*: Získává uživatelské jméno z vstupního požadavku.

*uPass = getRequestString("userpassword")*: Získává heslo uživatele z vstupního požadavku.

*sql = 'SELECT \* FROM Users WHERE Name ="' + uName + '" AND Pass ="' + uPass + '"'*: Sestavuje SQL dotaz pro vyhledání záznamu v databázi, kde uživatelské jméno a heslo odpovídají získaným hodnotám. }

SQL injekce obvykle spadá do tří kategorií: [7]

* In-band SQLi – Při útoku In-band SQLi útočník využívá stejný kanál komunikace jak pro provedení útoku, tak pro sběr výsledků. Tato metoda se vyznačuje jednoduchostí a efektivitou a řadí se mezi nejčastější typy SQLi útoků. Existují dvě podvarianty této strategie:
* Error-based SQLi – spočívá v aktivitách, které vyvolávají vznik chybových zpráv ze strany databáze. To umožňuje útočníkovi získat přístup k informacím o struktuře databáze prostřednictvím těchto chybových zpráv.
* Union-based SQLi – tato metoda využívá UNION SQL operátoru k spojení několika select příkazů z databází do jediné HTTP odpovědi. Taková odpověď může obsahovat data, která mohou být zneužita útočníkem.
* Inferential (Blind) SQLi – útočník posílá datové zátěže na server a pečlivě sleduje jeho odpovědi a reakce, aby získal další informace o jeho struktuře. Tato strategie je známá jako Blind SQLi, jelikož útočník nepřímo získává data z databáze webové stránky, což mu neumožňuje přímo identifikovat informace o útoku prostřednictvím komunikačního kanálu.
* Blind SQL injekce vyžadují odezvu a interakci serveru, což je důvodem jejich obvykle pomalejšího průběhu, ale přesto mohou způsobit stejnou úroveň škod. Tyto formy SQL injekcí mohou být rozděleny do následujících kategorií:
* Boolovské – útočník posílá do databáze SQL dotaz a následně čeká na odpověď, což nutí aplikaci reagovat. Reakce aplikace se odvíjí od toho, zda je dotaz pravdivý či nepravdivý. Na základě této odpovědi se obsah HTTP odpovědi buď změní nebo zůstane beze změny. Tím umožňuje útočníkovi zjistit, jestli jeho dotaz vyústil v pravdivý nebo nepravdivý výsledek.
* Časově závislé - útočník využívá SQL dotaz, který donutí databázi k odkladu odpovědi po určitou dobu. Tímto zpožděním může útočník monitorovat dobu, kterou databáze potřebuje k odpovědi, a na základě této informace posoudit, zda je dotaz pravdivý či nepravdivý. Na základě výsledků se generuje HTTP odpověď, která může být buď okamžitá nebo zpožděná. Díky tomu může útočník určit správnost dotazu bez nutnosti přístupu k databázovým datům.
* Out-of-band SQLi – Takovýto typ útoku lze uskutečnit pouze v situaci, kdy jsou aktivní určité funkce na databázovém serveru, který webová aplikace využívá. Často se tento způsob útoku používá jako alternativa k In-band a Inferenčním SQLi technikám.
* Out-of-band SQLi útoky jsou používány v situacích, kde útočník nemá možnost využít ten samý komunikační kanál pro zahájení útoku a následný sběr dat, nebo pokud je cílový server příliš pomalý či nestabilní pro běžný průběh útoku. Pro přenos informací zpět k útočníkovi se tyto metody spoléhají na vytváření speciálních DNS nebo HTTP požadavků ze strany serveru.

Pro ochranu před útoky typu SQL injection je klíčové provádět důkladnou kontrolu vstupních dat a využívat techniku parametrizace dotazů, včetně použití předpřipravených procedur. Aplikace by se neměla spoléhat pouze na data získaná od uživatelů. Je důležité, aby vývojáři přijali opatření k zabezpečení všech dat, která do systému vstupují, bez ohledu na jejich původ, včetně těch, které pocházejí z různých typů webových formulářů. Kritickým krokem je odstranění libovolných prvků v kódu, které mohou ohrozit bezpečnost, jako jsou například apostrofy. Navíc je doporučeno vyloučit možnost, aby byly chyby databáze zpřístupněny veřejnosti na webových stránkách, aby se zabránilo jejich potenciálnímu využití pro odhalení citlivých informací o struktuře databáze. [9]

### Cross-Site Scripting

Cross-Site Scripting (XSS) patří mezi druhy injekčních útoků, kdy nebezpečné skripty jsou začleňovány do webů, které jsou vnímány jako bezpečné. Tato situace nastává, když útočníci využijí slabiny webové aplikace k distribuci škodlivého obsahu, obvykle skriptu, který se vykonává ve webovém prohlížeči cílového uživatele. Možnosti pro takové útoky se často vyskytují, jelikož mnoho webových aplikací zpracovává vstupy od uživatelů bez předchozího ověření nebo řádného zakódování, což vede k jejich zranitelnosti. [10]

Útočník může využít XSS útok k tomu, aby poslal škodlivý kód do prohlížeče uživatele, který o nebezpečí netuší. Protože prohlížeč nemá žádný mechanismus, jak rozlišit nebezpečný skript od důvěryhodného obsahu, automaticky jej spustí. Tento postup umožňuje útočníkovi získat přístup k citlivým údajům uživatele, jako jsou cookies a session tokeny, které jsou součástí prohlížeče a vztahují se k aktuálně navštívenému webu. Kromě toho může škodlivý kód upravovat i vzhled a strukturu webové stránky tím, že mění její HTML kód. [10]

XSS můžeme dělit do 3 kategorií: [10]

* Reflektované útoky typu XSS – jedná se o útoky, kde skript vložený útočníkem je poslán zpět uživateli webovým serverem, což se může objevit v různých formách, včetně chybových hlášení, výstupů z hledání, nebo jiných typů odpovědí, které integrují jeden nebo více uživatelem zaslaných vstupů jako část požadavku na server. Tento typ útoku je následně předán cílovým osobám prostřednictvím alternativních metod, například prostřednictvím emailu nebo přes různé webové stránky. Když je uživatel oklamán k interakci   
  s nebezpečným odkazem, k odeslání formuláře připraveného pro tento účel, nebo jednoduše při prohlížení kompromitované webové stránky, škodlivý skript je přenesen na webové místo s bezpečnostním nedostatkem, které následně odesílá skript zpět do prohlížeče uživatele. Prohlížeč pak vykoná tento skript pod předpokladem, že pochází z ověřeného zdroje.
* Uložené útoky typu XSS – jedná se o situace, kdy škodlivý kód je vložen   
  a trvale zachován na cílovém serveru, například v jeho databázích, na diskuzních fórech, v záznamech o návštěvách nebo v oblastech určených pro komentáře. Takový skript je pak poskytnut obětem, když si vyžádají informace, které byly dříve uloženy, což způsobí jejich infekci.
* Blind Cross-site Scripting – tento ptok obvykle nastává, když útočníkův škodlivý kód je uložen na serveru a poté je odeslán zpět směrem k uživateli prostřednictvím aplikace na straně serveru. Například v případech, kdy se využívají formuláře pro sběr zpětných vazeb, může útočník prostřednictvím takového formuláře poslat škodlivý obsah. Jakmile pak uživatel   
  s administrátorskými právy nebo jiný pracovník zodpovědný za zpracování těchto formulářů prostřednictvím systému na serveru zkontroluje obsah poslaný útočníkem, dojde k aktivaci škodlivého skriptu.

Obecně lze efektivně předcházet XSS zranitelnostem pomocí kombinace následujících opatření: [11]

* Filtrovat vstupy při jejich příjmu – na místě, kde systém sbírá informace od uživatelů, je klíčové uplatňovat důkladnou kontrolu vstupů v souladu   
  s očekáváními nebo pravidly pro validní data.
* Kódovat data při jejich výstupu – když systém odesílá data, která uživatelé mohou ovlivnit, ve formě HTTP odpovědí, je potřeba tato data šifrovat tak, aby se eliminovala možnost jejich vyhodnocení jako spustitelného kódu. Podle situace, v jaké se data odesílají, může být zapotřebí využít kombinaci šifrovacích metod pro HTML, URL adresy, JavaScript a CSS.
* Používat vhodné hlavičky odpovědi – aby se minimalizovalo riziko XSS útoků   
  v případech, kdy HTTP odpovědi nezahrnují HTML ani JavaScript, lze použít specifické hlavičky, jako jsou Content-Type a X-Content-Type-Options. Tyto hlavičky pomáhají zajistit, že prohlížeče interpretují odpovědi podle vašich intencí.
* Politika zabezpečení obsahu - pro další omezení možných XSS útoků lze implementovat Politiku zabezpečení obsahu (CSP), která slouží jako dodatečná ochranná vrstva.

### Brute Force útok

Útok typu brute force, který se často využívá k porušení bezpečnostních opatření,   
je zodpovědný za pět procent všech doložených případů narušení bezpečnosti, podle určitých odhadů. Tato metoda se zakládá na systematickém pokusu o odhadnutí přihlašovacích údajů uživatelů s úmyslem získat neautorizovaný přístup do systému. Navzdory její zdánlivé jednoduchosti, útoky typu brute force často vedou k úspěchu. [12]

Někteří útočníci využívají aplikace a skripty jako prostředky k provádění brute force útoků. Tyto nástroje systematicky testují mnoho kombinací hesel s cílem obejít autentizační procesy. V jiných případech se snaží najít validní relační ID k získání přístupu k webovým aplikacím. Účelem může být krádež informací, infikování webových stránek malwarem či narušení poskytovaných služeb. [12]

I přesto, že někteří stále provádějí brute force útoky ručně, v současné době jsou téměř všechny útoky automatizovány pomocí botů. Osoby disponují seznamy běžně používaných přihlašovacích údajů nebo reálných uživatelských údajů získaných   
z bezpečnostních průlomů nebo temného webu. Boti systematicky navštěvují webové stránky, kde zkoušejí tyto seznamy přihlašovacích údajů, a informují osobu, která provádí útok, v případě úspěchu získání přístupu. [12]

Brute force útoky rozdělujeme do několika typů: [12]

* Jednoduchý útok hrubou silou – tento typ útoku spočívá v systematickém odhadu bez nutnosti spoléhat se na vnější logiku.
* Hybridní útok - využívá externí logiku k nalezení potenciálně úspěšných variant hesel a poté systematicky testuje širokou škálu kombinací s cílem získat přístup.
* Slovníkový útok – usiluje o odhalení uživatelských jmen nebo hesel prostřednictvím prohledávání slovníku možných kombinací.
* Útoky s využitím rainbow tabule – pracují s předem vypočítanými tabulkami pro reverzní získání kryptografických hash funkcí.
* Útok hrubou silou obrácený – Využívá běžná hesla k útokům na širokou škálu potenciálních uživatelských jmen, přičemž kombinuje tyto hesla s již známými kombinacemi hesel a uživatelských jmen na různých webových stránkách. Tento přístup vychází z faktu, že mnoho uživatelů má tendenci používat identické přihlašovací údaje na různých online službách.

Způsoby, kterými se lze bránit proti brute force útokům mohou být: [13]

* Politika silných hesel – složitost hesla přímo ovlivňuje dobu potřebnou k jeho prolomení. Avšak, samotná délka není vždy rozhodujícím prvkem. Účinná ochrana spočívá v jedinečné kombinaci písmen, čísel a znaků.. [13]
* Vícefaktorová autentizace – pro zvýšení bezpečnosti při používání internetových aplikací, jako jsou cloudové služby pro sdílení souborů nebo systémy pro online platby, se doporučuje zavést dvoustupňové ověření. Tento postup účinně chrání před neautorizovaným přístupem, protože kromě tradičního hesla vyžaduje ještě jeden, dodatečný prvek ověření, což znemožňuje útočníkům získat přístup prostřednictvím útoků hrubou silou. Pro další zabezpečení lze využít například hardwarové tokeny nebo metody založené na biometrii, jako je skenování duhovky nebo rozpoznání hlasu, které poskytují ještě lepší ochranu proti pokusům o neoprávněný vstup.
* Omezení pokusů o přihlášení – omezení přístupu po sérii nezdařených pokusů o přihlášení je efektivním způsobem, jak předcházet útokům metodou brute force, jelikož přerušuje pokusy o neoprávněné vniknutí již v jejich počátku.
* Použití CAPTCHA – CAPTCHA představuje důležitou ochranu proti automatizovaným útokům. Tato technologie umožňuje efektivní rozlišení mezi opravdovými uživateli a robotickými skripty pomocí úloh, které lidé mohou snadno vyřešit, zatímco pro stroje tyto úkoly představují významnou výzvu.
* Monitorování a reakce na útoky hrubou silou - průběžné sledování logů je klíčové pro odhalení pokusů o útok hrubou silou na síť. analýza logů v reálném čase a nástroje SIEM umožňují detekci podezřelých vzorů a sledování neúspěšných pokusů o přihlášení.
* Systém detekce průniků (IDS) - implementace síťového systému detekce průniků (IDS) může účinně monitorovat webové stránky nebo sítě na neobvyklou nebo podezřelou aktivitu. IDS dokáže rychle identifikovat vzory naznačující útoky hrubou silou a včasným vyvoláním poplachu umožnit bezpečnostnímu týmu rychlou reakci a minimalizaci potenciálních hrozeb.

### DoS útok

Denial-of-Service (DoS) útok je útok zaměřený na vypnutí počítače nebo sítě, čímž se stává nepřístupným pro své zamýšlené uživatele. DoS útoky dosahují tohoto cíle zaplavením cíle provozem nebo odesláním informací, které způsobují pád systému.

V obou případech DoS útoky zbavují legitimní uživatele služby nebo prostředků, které očekávali. [14]

Oběti DoS útoků často zahrnují webové servery významných organizací, jako jsou banky, obchodní a mediální společnosti nebo vládní a obchodní organizace. Ačkoliv DoS útoky obvykle nevedou k odcizení nebo ztrátě důležitých informací nebo jiných aktiv, avšak mohou oběti stát velké množství času a peněz na jejich řešení. [14]

Existují dvě obecné metody DoS útoků: zaplavování služeb nebo pád služeb. Zaplavovací útoky nastávají, když systém přijímá příliš mnoho provozu, což způsobuje zpomalení, a nakonec zastavení serveru. [14]

Nejvíce užívané útoky jsou: [14]

* Přetékání bufferu – nejčastější typ DoS útoku. Princip spočívá v odeslání většího množství provozu na síťovou adresu, než byli programátoři schopni systém zpracovat. Kromě ní patří sem i útoky uvedené níže, a to včetně dalších, které jsou navrženy k využití chyb specifických pro určité aplikace nebo sítě.
* ICMP flood – využívá špatně nakonfigurovaná síťová zařízení odesíláním falšovaných paketů, které pingují každý počítač v cílové síti, namísto jediného konkrétního stroje. Síť je pak spuštěna tak, aby zesílila provoz. Tento útok je také znám jako smurf útok nebo ping smrti.
* SYN flood – odesílá žádost o připojení k serveru, ale nikdy nesestaví kompletní handshaking. Pokračuje až do doby, kdy jsou všechny otevřené porty zahlceny žádostmi a pro legitimní uživatele nejsou žádné dostupné.

# Praktická část

V této kapitole se zaměřím na samotné fungování webové stránky a její jednotlivé komponenty.

## Domovská stránka (index.php)

Jedná se o úvodní stránku, kterou uživatelé navštíví po příchod na web. Samotný kód se skládá z několika sekcí:

Deklarace HTML dokumentu - začíná deklarací HTML dokumentu pomocí značky *<html lang="cs">.* Atribut *lang="cs*" označuje, že jazyk použitý na této stránce je čeština. Tato deklarace je důležitá pro správnou interpretaci znaků a lokalizaci obsahu.

Sekce *<head>* - tato část obsahuje metadata dokumentu. Prvním meta elementem *<meta charset="UTF-8">* se určuje kódování znaků dokumentu, které je nastaveno na UTF-8. To zajišťuje, že stránka bude správně zobrazovat speciální znaky a symboly. Druhý meta element *<meta name="viewport" content="width=device-width, initial-scale=1.0" />* definuje zobrazení stránky na různých zařízeních, jako jsou mobilní telefony, tablety a počítače. Tím se zajistí, že obsah bude správně škálován a bude se dobře přizpůsobovat různým velikostem obrazovek. Poslední částí v *<head>* sekci je odkaz na externí CSS soubor pomocí *<link rel="stylesheet" href="assets/styles/style.css">*. Tento soubor obsahuje stylování pro různé prvky na stránce, což pomáhá zajistit konzistentní a atraktivní vzhled.

Hlavička <header> - Tato část obsahuje navigační lištu s odkazy na různé stránky webu. Každý odkaz je zahrnut v <a> značce, která umožňuje uživatelům přecházet na danou stránku. V této hlavičce najdeme odkazy na domovskou stránku   
*<a href="index.php" class="nav-link">Domů</a>*, galerii *<a href="gallery.php" class="nav-link">Galerie</a>*, blog *<a href="blog.php" class="nav-link">Blog</a>*, kalkulačku *<a href="calcu.php" class="nav-link">Kalkulace</a>* a kontaktní stránku *<a href="kontakty.html" class="nav-link">Kontakty</a>*.

Obr. . Domovská stránka

V hlavičce stránky se nachází část kódu, která dynamicky mění text odkazu   
v navigační liště v závislosti na stavu přihlášení uživatele. Tato část kódu používá podmíněné bloky PHP, aby určila, zda je uživatel přihlášený. Pokud je uživatel přihlášený (což je určeno podle existence relační proměnné *$\_SESSION['loggedin']*,zobrazí se odkaz na odhlášení s textem "Odhlásit" a přesměrováním na stránku "*logout.php*". V opačném případě, když uživatel není přihlášený, zobrazí se odkaz na přihlášení s textem "Přihlášení" a přesměrováním na stránku "*login.php*". Tato dynamická funkcionalita umožňuje uživatelům snadno navigovat mezi přihlášením  
a odhlášením na stránce a poskytuje jim relevantní možnosti v závislosti na jejich aktuálním stavu.

Div s logem - tento *<div>* obsahuje obrázek, který slouží jako logo stránky. Obrázek je zahrnut pomocí značky *<img>* s atributem *src*, který odkazuje na umístění obrázku, a atributem *width*, který určuje šířku obrázku.

## Změna obrázků, hamburger menu (script.js)

První část kódu začíná definicí proměnných hamburger a navMenu, které získávají odkazy na prvky DOM pomocí metody *document.querySelector()*. Prvek s třídou *.hamburger* představuje ikonu menu, zatímco prvek s třídou .nav-menu představuje navigační menu.

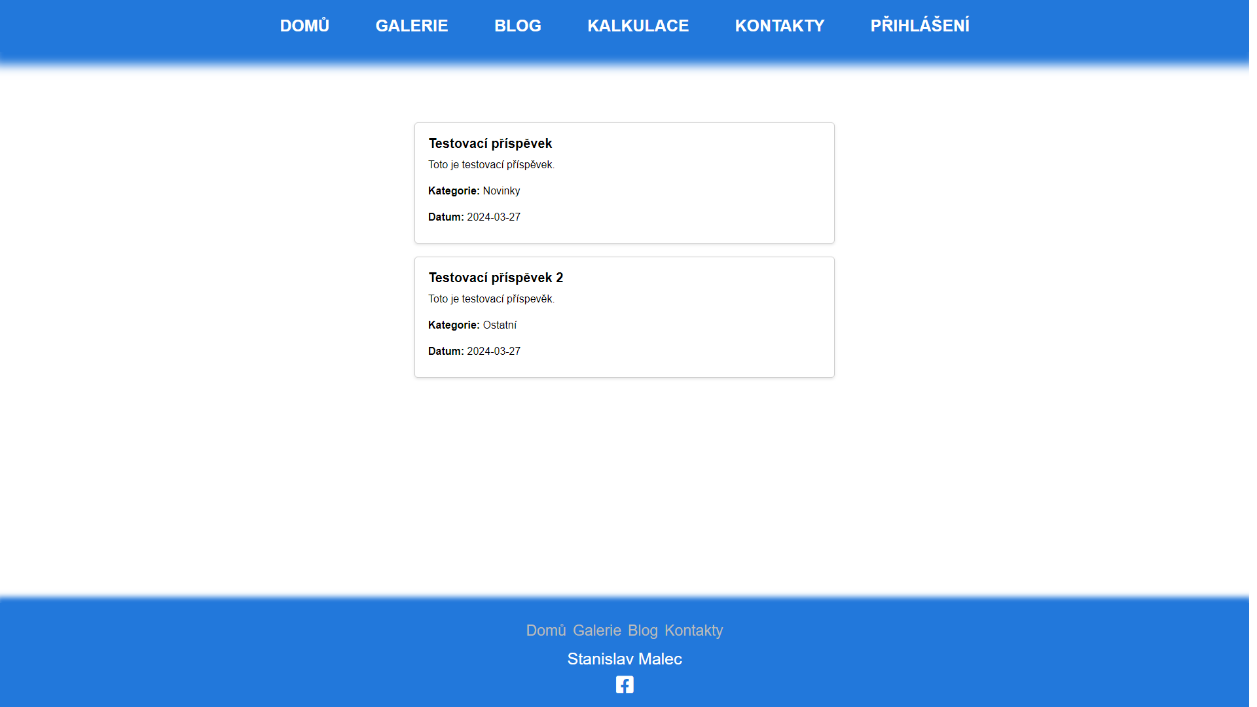
Poté následuje událostní posluchač na ikonu hamburgeru, který reaguje na kliknutí. Pokud je ikona hamburgeru kliknutá, provede se anonymní funkce. Tato funkce přepíná přítomnost třídy active na ikoně hamburgeru a navigačním menu pomocí metody *classList.toggle().*

Následuje část kódu, která vyhledává všechny prvky s třídou .nav-link pomocí metody *document.querySelectorAll()* a pro každý z nich přidává událostní poslouchač   
na kliknutí. Pokud je jakýkoli odkaz v navigačním menu kliknut, provede se anonymní funkce. Tato funkce odstraňuje třídu active z ikony hamburgeru a navigačního menu, což zajišťuje, že navigační menu bude skryté po kliknutí na odkaz.

Poslední část kódu definuje funkci changePhoto(), která slouží k přepínání snímků   
v karuselu obrázků. Tato funkce nejprve vyhledá prvky obsahující data atribut *[data-slides] a [data-active]*. Poté vypočítá index aktuálně zobrazeného snímku a určí index následujícího snímku. Pokud aktuálně zobrazený snímek je posledním snímkem   
v seznamu, funkce přepne na první snímek a naopak. Nakonec funkce aktualizuje data atributy snímků tak, aby byl viditelný následující snímek a aktuálně zobrazený snímek ztratil atribut [data-active].

Kód vytváří intervalovou událost, která periodicky volá funkci changePhoto() každých 4000 milisekund (4 sekundy), čímž dochází k automatickému přepínání snímků   
v karuselu bez zásahu uživatele.

## Blog (blog.php)

Tato stránka slouží pro zobrazení jednotlivých příspěvků.

Obr. . Stránka blogu

Na začátku kódu je zahájena session voláním *session\_start()*. Tato funkce umožňuje práci se session proměnnými, které uchovávají informace mezi různými HTTP požadavky. To je klíčové pro identifikaci, zda je uživatel přihlášen, a následné přizpůsobení zobrazovaného obsahu a dostupných možností na webu.

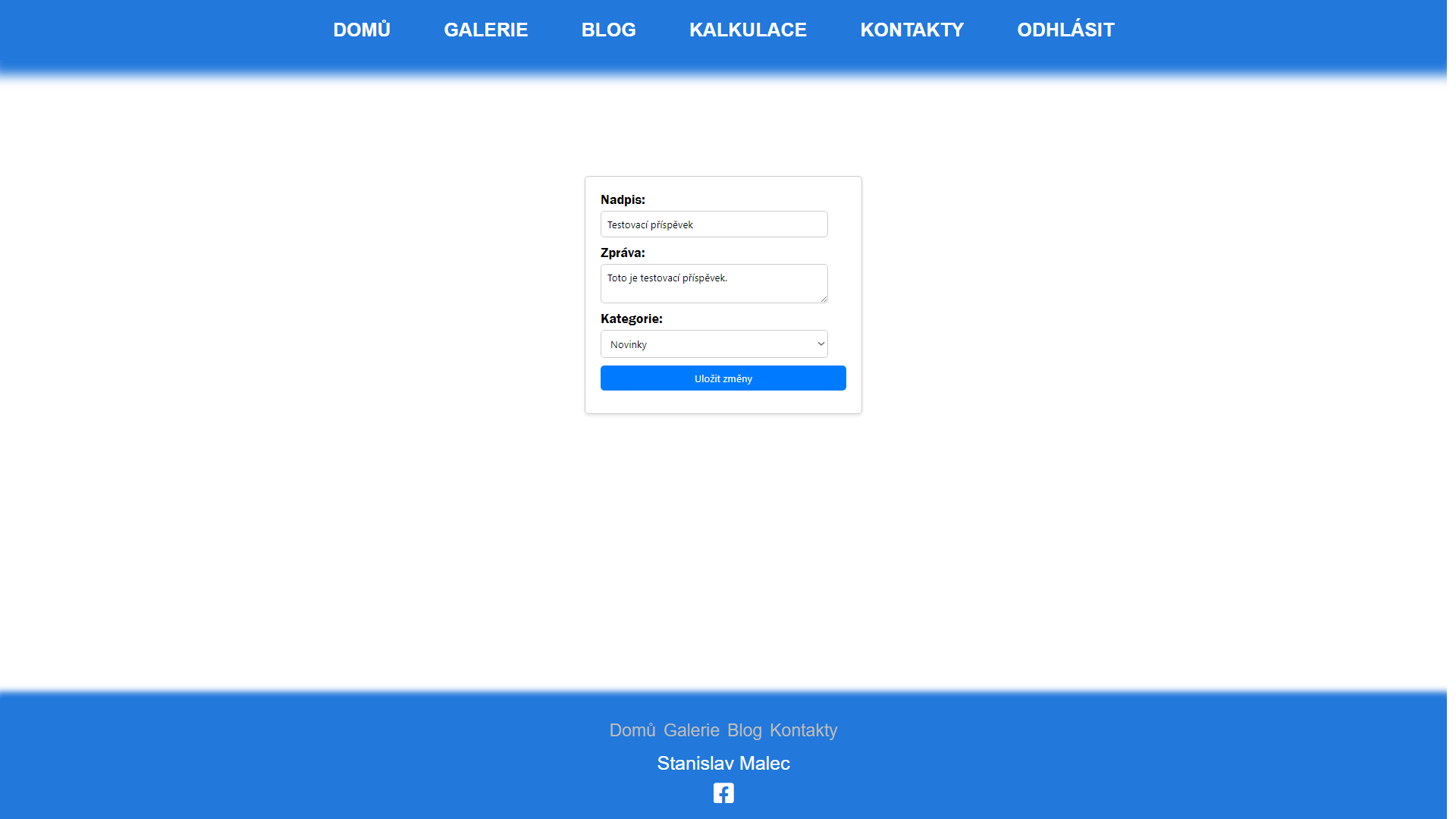
Načítání kategorií - pomocí objektu *$db*, který je instancí třídy Database, získává kód přístup k databázi. Skript připraví a spustí dotaz pro načtení všech kategorií z tabulky categories. Výsledky dotazu jsou uloženy do pole $categories, které se později využívá pro dynamické generování seznamu kategorií v rozbalovacím menu ve formuláři pro přidání nového příspěvku.

Mazání příspěvků - v případě, že je odeslán POST požadavek s akcí delete\_post, kód získá ID příspěvku z dat formuláře a připraví dotaz pro jeho smazání z tabulky blog. Pomocí metody pro vazbu parametrů se ID bezpečně přiřadí k dotazu, čímž se minimalizuje riziko SQL injekcí. Pokud je dotaz úspěšně proveden, dojde   
k přesměrování na stránku *blog.php*; v opačném případě se zobrazí chybová hláška.

Zobrazení příspěvků blogu - skript dynamicky generuje obsah blogu. Z databáze jsou načteny informace o příspěvcích, včetně titulku, obsahu, kategorie a data publikace. Pro každý příspěvek se vygeneruje HTML kód, který tyto informace prezentuje.   
K určení názvu kategorie se využívá dříve načtené pole *$categories*. Uživatelům   
s administrátorskými právy se navíc zobrazí možnosti pro úpravu nebo smazání každého příspěvku.

Formulář pro přidání nového příspěvku - pro uživatele s administrátorskými právy se na stránce zobrazí formulář umožňující přidání nového příspěvku na blog. Formulář obsahuje pole pro zadání nadpisu, obsahu příspěvku a výběr kategorie z rozevíracího seznamu, jehož možnosti jsou generovány na základě pole *$categories*. Po odeslání formuláře jsou data zpracována dalším kódem.

### Editace příspěvků (edit\_post.php)

Tato stránka slouží jako formulář pro úpravu jednotlivých příspěvků v blogu. 

Obr. . Stránka pro úpravu příspěvku

Na začátku kódu se kontroluje, zda je uživatel přihlášen a zda má roli administrátora. To je docíleno pomocí session proměnných, kde *$\_SESSION['loggedin']* indikuje, zda je uživatel přihlášen, a *$\_SESSION['username']* obsahuje jméno uživatele. Pokud uživatel není přihlášen nebo není administrátor, je přesměrován na stránku blogu.

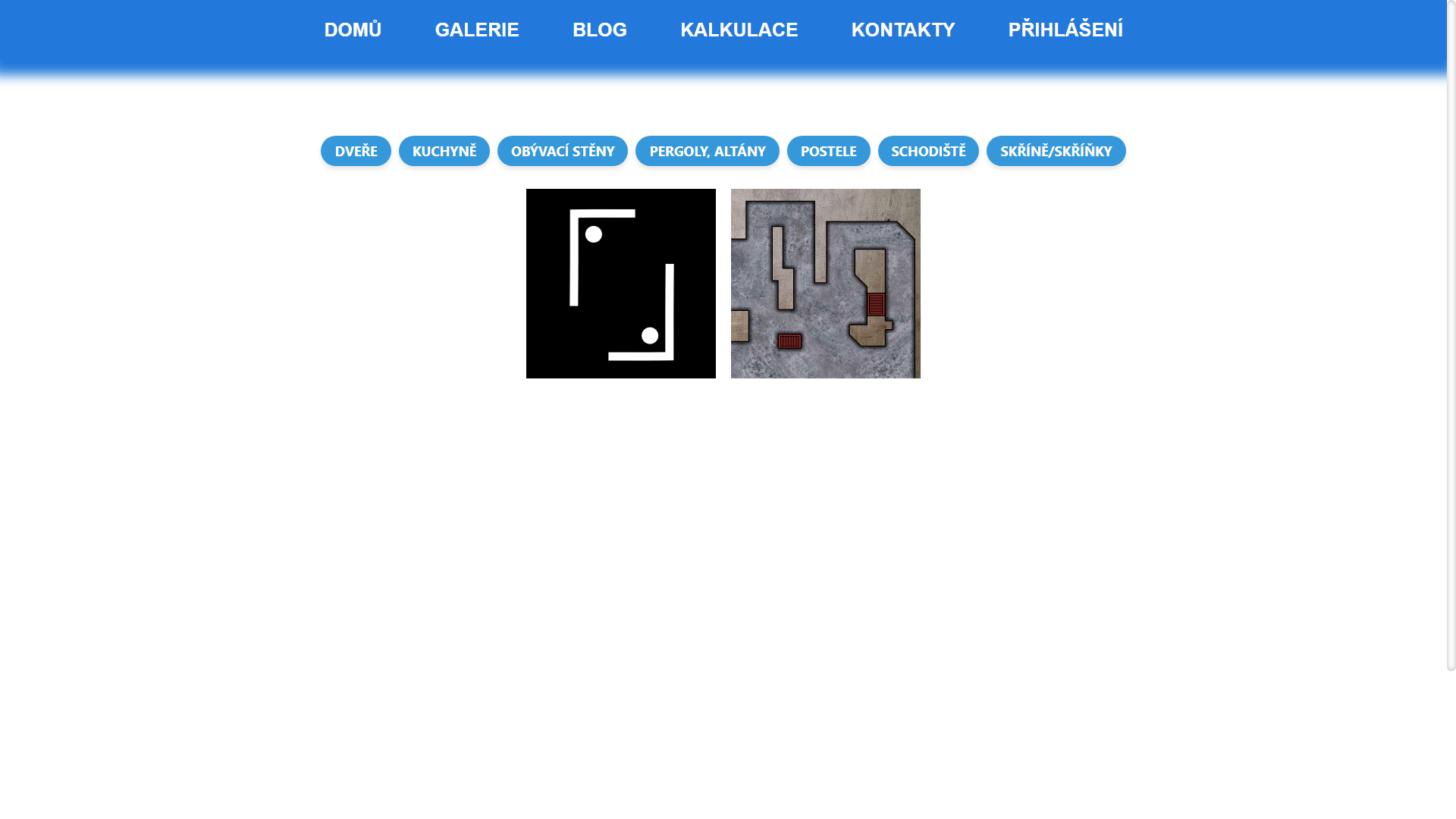
Načtení dat příspěvku - kód dále ověřuje, zda bylo v URL předáno ID příspěvku pomocí *GET* požadavku. Pokud není ID příspěvku správně poskytnuto, uživatel je přesměrován na stránku blogu. Pokud je ID dostupné, použije se pro načtení informací příspěvku z databáze. Výsledek dotazu je uložen do proměnné $post.

Zpracování úprav příspěvku - pokud je formulář pro editaci odeslán pomocí $\_POST['submit'], kód zobrazí nové hodnoty titulku, obsahu a kategorie z odeslaných dat formuláře. Tyto hodnoty jsou dále použity v připraveném dotazu pro aktualizaci příspěvku v databázi. V případě úspěchu je v session nastavena zpráva o úspěšné aktualizaci, v opačném případě je uživatel informován o chybě. Po dokončení je uživatel přesměrován zpět na stránku blogu

Generování uživatelského rozhraní – v další části se nachází formulář pro editaci příspěvku. Tento formulář je předvyplněn aktuálními údaji příspěvku.

## Galerie (gallery.php)

Tato stránka obsahuje fotografie, které jsou na stránku přidány.



Obr. . Stránka galerie

Na začátku kódu je zahájena session a propojení s databází, viz. Kapitola 2.4.

Rozhraní pro výběr kategorií - v sekci *<section class="gallery">* se nachází formulář s tlačítky pro výběr jednotlivých kategorií obrázků, které mají být zobrazeny. Uživatel zvolí jednu z možných kategorií a po odeslání formuláře se zobrazí obrázky patřící do vybrané kategorie. Pro zobrazení obrázků podle kategorií je implementován soubor category.php, viz. Kapitola 2.4.2

Možnost přidání obrázků pro Administrátora - pokud je uživatel přihlášen jako administrátor zobrazí se formulář, který umožňuje nahrání nového obrázku do galerie. Formulář obsahuje tlačítko pro nahrání soubrou z počítače Po vyplnění všech údajů   
a stisknutí tlačítka se prove příkaz post.php, viz. Kapitola 2.4.1.

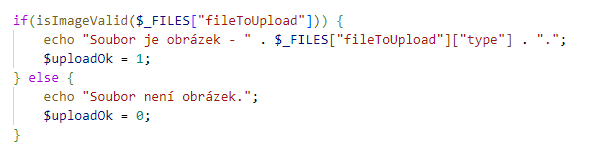
### Přidávání obrázků (post.php)

Tento kód se zaměřuje na přidávání obrázků do galerie. Samotný kód se dělí do několika sekcí:

Příprava cílové složky - kód nejprve ověří, zda byl formulář odeslán pomocí *$\_POST["submit"]*, pokud ano, načte kategorii z odeslaných dat formuláře a určí cílovou složku na základě této kategorie. Složka je vytvořena v adresáři *uploads/*,   
a pokud neexistuje, je automaticky vytvořena s přístupovými právy 0777, toto umožňuje čtení, zápis a spouštění souborů ve složce.

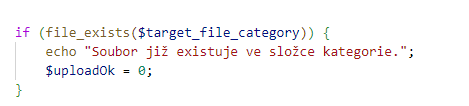
Zpracování nahrávaného souboru - kód extrahuje název souboru z globální proměnné *$\_FILES["fileToUpload"]["name"]* a vytvoří cílovou cestu souboru v rámci kategorie.

Následují validace pro ověření souboru, zda splňuje určité podmínky.

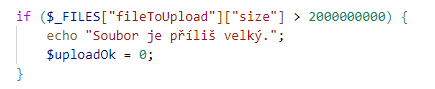
Kontrola, zda soubor je obrázek - funkce *isImageValid* ověřuje, zda má soubor platné vlastnosti obrázku pomocí funkce *getimagesize*, která zjišťuje rozměry souboru. Pokud *getimagesize* vrátí negativní hodnotu, znamená to, že soubor není obrázek   
a nastaví se *$uploadOk = 0* zároveň s vypsanou zprávou "Soubor není obrázek."

Obr. . Kontrola za je soubor obrázek

Pokud je soubor validní obrázek, vypíše se potvrzení spolu s názvem souboru.

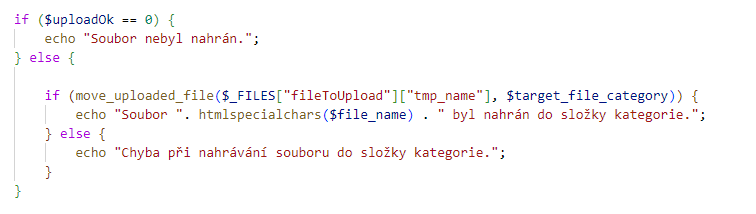
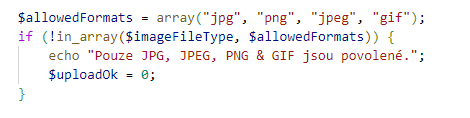
Kontrola existence souboru - pro vyhnutí se přepsání již existujícího souboru se provádí kontrola, zda soubor již ve složce existuje. Pokud existuje, *$uploadOk* se nastaví na 0 a vypíše se zpráva "Soubor již existuje ve složce kategorie."

Obr. . Kontrola za soubor již existuje

Kontrola velikosti souboru – v této části se kontroluje, zda velikost souboru nepřekračuje limit 25000 KB (200000000 bajtů). Toto omezení pomáhá předcházet problémům s přetížením nebo pomalým nahráváním velkých souborů. Pokud soubor přesahuje tuto velikost, *$uploadOk* je nastaven na 0 a vypíše se zpráva "Soubor je příliš velký."

Obr. . Kontrola velikosti souboru

Kontrola formátu souboru - zde je kontrolováno, zda je obrázek jeden z povolených typů: JPG, JPEG, PNG, nebo GIF. To je provedeno porovnáním přípony souboru. Tato validace chrání před nahráváním potenciálně škodlivých souborů, které se jeví jako obrázky. Pokud soubor není v povoleném formátu, $uploadOk se nastaví na 0 a vypíše se zpráva "Pouze JPG, JPEG, PNG & GIF jsou povolené."

Závěrečná kontrola a pokus o nahrání - po provedení všech výše uvedených kontrol se provede kontrola, zda proměnná *$uploadOk* stále zůstává nastavena na 1. Pokud ano, znamená to, že všechny validace proběhly úspěšně, a kód pokračuje pokusem o přesun souboru z dočasného umístění do cílové složky pomocí *move\_uploaded\_file*.

Obr. . Kontrola typu souboru

Zobrazení jednotlivých obrázků (category.php)

Obr. . Nahrání obrázku

V případě úspěchu je uživatel přesměrován zpět na stránku galerie, v opačném případě

se informuje o chybě při nahrávání.

Na začátku kódu je nejprve zkontrolováno, zda byla v URL adrese specifikována kategorie pomocí GET parametru category. Pokud ano, je zvolena odpovídající složka s obrázky; pokud ne, nevybere se žádná kategorie. Pokud byla zvolená kategorie a existuje příslušný adresář, kód načte všechny obrázky z daného adresáře do pole *$images*.

V kódu je pro každý načtený obrázek vytvořen div element s třídou *gallery-item*, který obsahuje obrázek *<img>*. Pokud je uživatel přihlášen jako admin u každého obrázku je dostupné tlačítko pro odstranění obrázku. Na obrázky je možné kliknout a toto kliknutí zobrazí obrázky v modal okně.

Pro zobrazení obrázků v modal okně je implementován JavaScript kód, který má následující funkce:

closeModal() - skryje modal okno tím, že nastaví jeho CSS vlastnost display na *none*. Tato funkce je volána, když uživatel klikne na tlačítko pro uzavření modal okna.

showImage(index) - zobrazí modal okno s obrázkem specifikovaným indexem. Nastaví *src* atribut elementu *modalImg* na cestu k obrázku, který odpovídá zadanému indexu v poli *images*. Poté nastaví *modalIndex* na zadaný index a zobrazí modal okno.

showNext() - přejde na následující obrázek v galerii. Vypočítá index nadcházejícího obrázku a implementuje animaci pro přechod. Po krátkém zpoždění aktualizuje zdroj obrázku a vrací animaci zpět do výchozího stavu.

showPrevious() - Přejde na předchozí obrázek v galerii.

### Odstranění obrázku (delete\_image.php)

Tento kód je navržen k tomu, aby umožňoval uživatelům s administrátorským přístupem smazat obrázky z galerie.

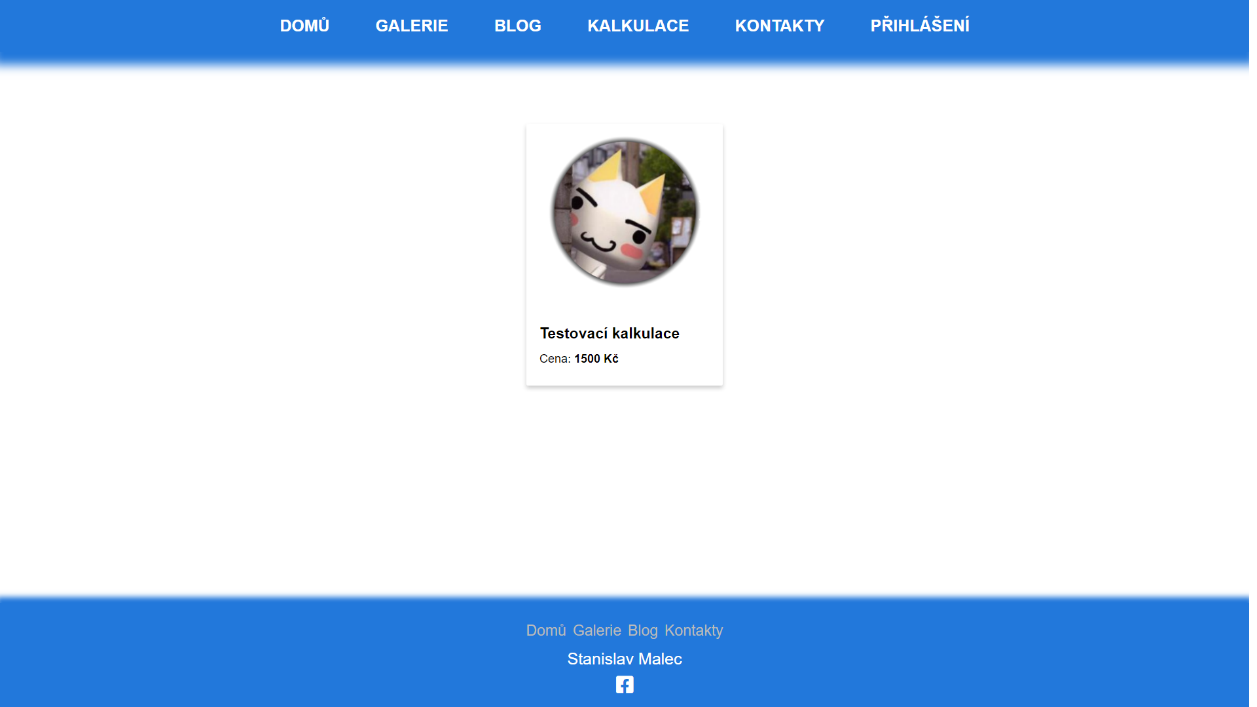
Kód ověřuje, zda byla odeslána POST data s klíčem *delete*. Toto je bezpečnostní kontrola, která zajišťuje, že kód pokračuje jen v případě, pokud opravdu byl proveden pokus o odstranění obrázku.

Pokud je uživatel oprávněn obrázek smazat, kód zpracuje cestu k obrázku a jeho název, které jsou předány v POST požadavku. Kód dále ověří, zda soubor na určené cestě existuje. Pokud je soubor nalezen, použije se PHP funkce *unlink*, která obrázek ze složky odstraní.

V případě úspěchu nebo neúspěchu informuje uživatele odpovídající zprávou. Úspěšné smazání je potvrzeno zprávou o odstranění, pokud soubor není nalezen nebo již byl dříve odstraněn, kód vypíše chybovou zprávu.

## Vzorové kalkulace (calcu.php)

Začátek kódu je zaměřen na inicializaci session a propojení s databází, viz. Kapitola 2.3.

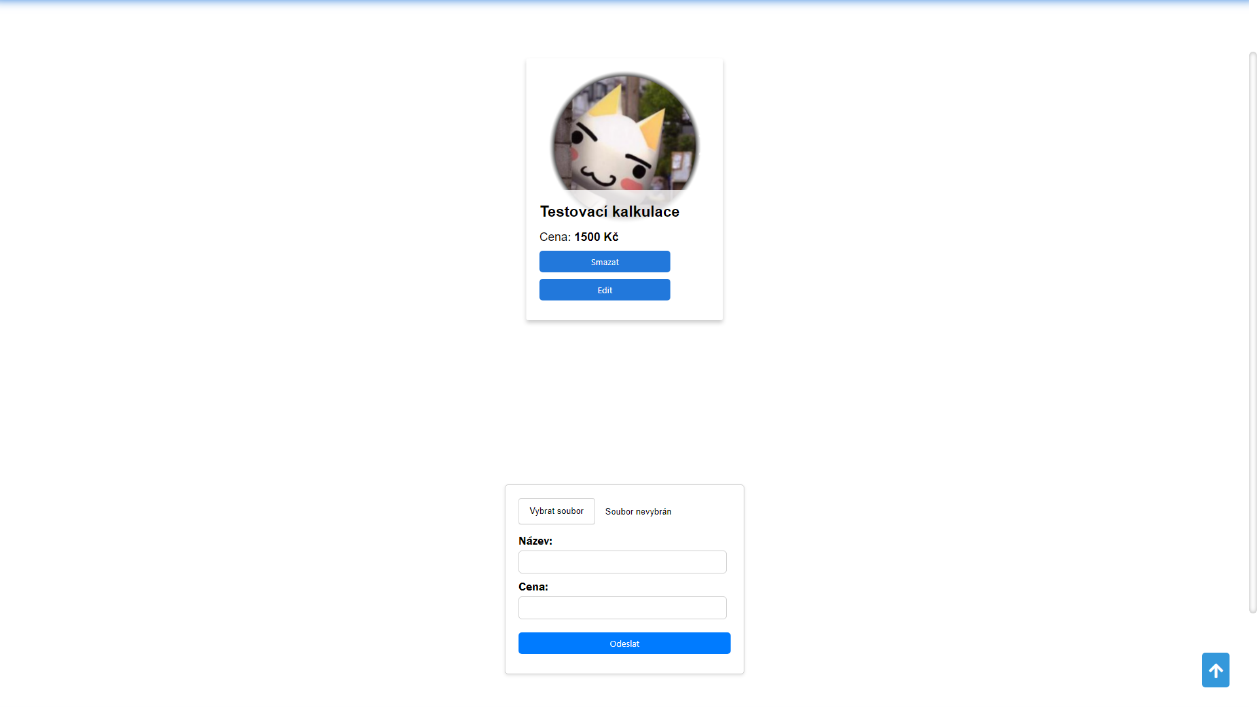


Obr. . Stránka pro vzorové kalkulace

Odstranění Položky a Přidruženého Obrázku:

Důležitou částí kódu je zpracování požadavku na smazání položky, které je iniciováno odesláním formuláře s akcí *delete\_post*. Kód nejprve ověří, zda má uživatel oprávnění k této akci. Pokud ano, z databáze je načtena cesta k obrázku spojenému   
s mazanou položkou, a následně je položka odstraněna z databáze. Pokud je položka úspěšně odstraněna, kód se pokusí odstranit i příslušný obrázek ze složky. V závislosti na výsledku této operace je do session proměnné *calc\_result* uložena zpráva informující o úspěchu nebo případných chybách, která jsou později zobrazena uživateli.

Dynamické Generování Obsahu:

Tělo skriptu pokračuje generováním HTML kódu pro zobrazení jednotlivých kalkulačních položek. Pro každou položku je vytvořen HTML kontejner, který obsahuje obrázek položky, její název a cenu. Pro uživatele s administrátorským přístupem jsou také dostupná tlačítka pro smazání a úpravu kalkulace. 

Obr. . Stránka pro kalkulace s administrátorskými právy

Formulář pro Přidání Nové Položky:

Pro uživatele s administrátorským přístupem je na konci stránky umístěn formulář, který umožňuje přidání nové kalkulační položky. Formulář obsahuje pole pro nahrání souboru, pole pro zadání názvu a ceny nové položky. Položka je následně odeslána do dalšího skriptu post\_calc.php

### Editace kalkulací (edit\_calc.php)

Úprava vzorových kalkulací funguje na stejné bázi jako úprava příspěvků, viz. Kapitola 2.3.1.

### Přidávání kalkulací (post\_calc.php)

Tento PHP skript umožňuje nahrávání obrázků na web a jejich ukládáním do databáze spolu s názvem a cenou.

Skript začíná kontrolou, zda byl soubor skutečně odeslán pomocí formuláře. Cesta pro uložení souboru je definována v proměnné $target\_dir, zatímco název souboru se získává z globální proměnné $\_FILES. Dále je sestavena celá cílová cesta souboru   
a je inicializována proměnná $uploadOk, která slouží jako indikátor, zda je soubor připraven k nahrání.

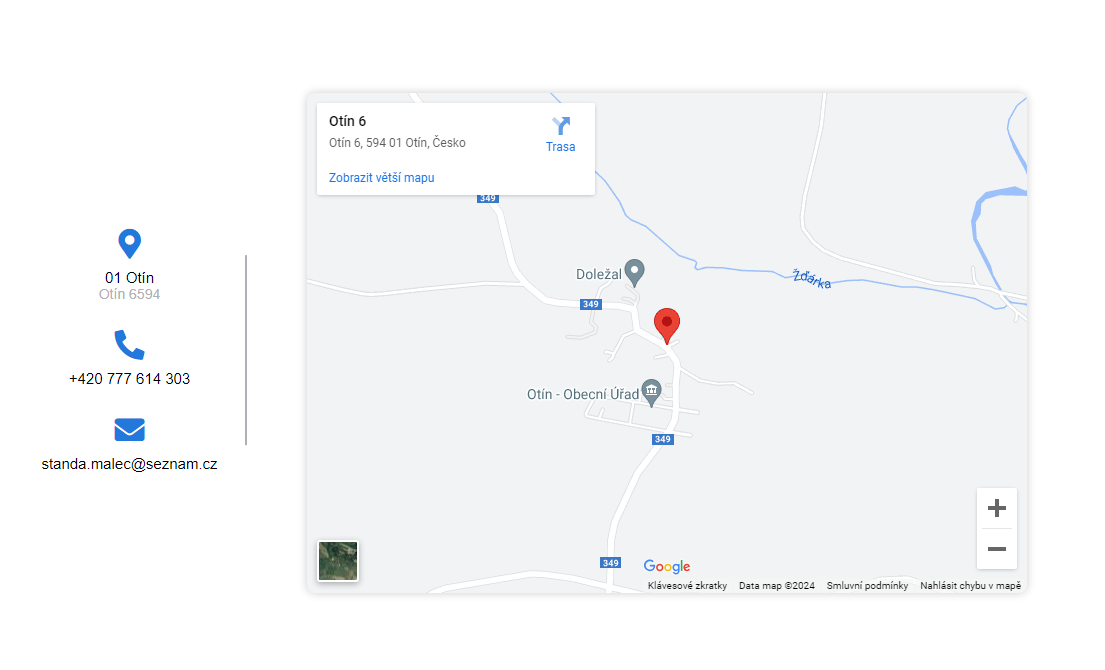
Validace souboru – viz. Kapitola 2.4.1.

Pokud všechny předchozí kontroly proběhnou úspěšně, skript pokračuje pokusem   
o nahrání souboru do cílového adresáře pomocí funkce *move\_uploaded\_file()*. Pokud je soubor úspěšně nahrán, skript pokračuje k uložení souvisejících informací do databáze.

Nahrání doprovází vytvoření záznamu v databázi, kde jsou uloženy názvy, ceny a názvy obrázků. K tomu slouží připravený SQL dotaz, do kterého jsou vloženy hodnoty z formuláře a názvu nahrávaného souboru. V případě úspěchu je uživatel přesměrován zpět na stránku kalkulací. V opačném případě je uživatel informován o chybě při nahrávání souboru do databáze.

## Kontakty (kontakty.php)

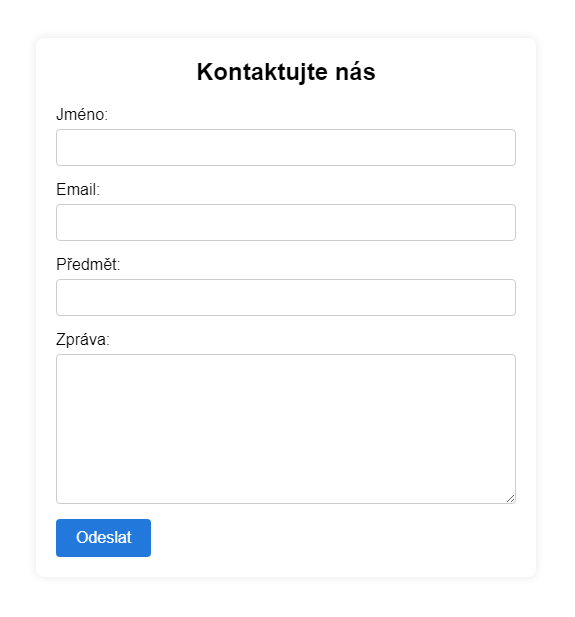
Tato stránka slouží k zobrazení informací o firmě jako jsou emailová adresa, telefonní číslo a adresa.

Sekce s informacemi o kontaktu obsahuje adresu, telefonní číslo a e-mailovou adresu. Každá z těchto informací je vizuálně odlišena ikonou, aby byli jednotlivé informace od sebe odlišitelné.

Obr. . Panel s informacemi

Tato sekce je navržena tak, aby poskytovala veškeré potřebné kontaktní údaje na jednom místě.

Vložená Google Mapa umožňuje návštěvníkům webu snadno nalézt fyzickou polohu spojenou s kontaktními informacemi. Mapa je implementována jako interaktivní *iframe*, což dovoluje uživatelům prohlížet mapu přímo na stránce bez nutnosti opustit web.

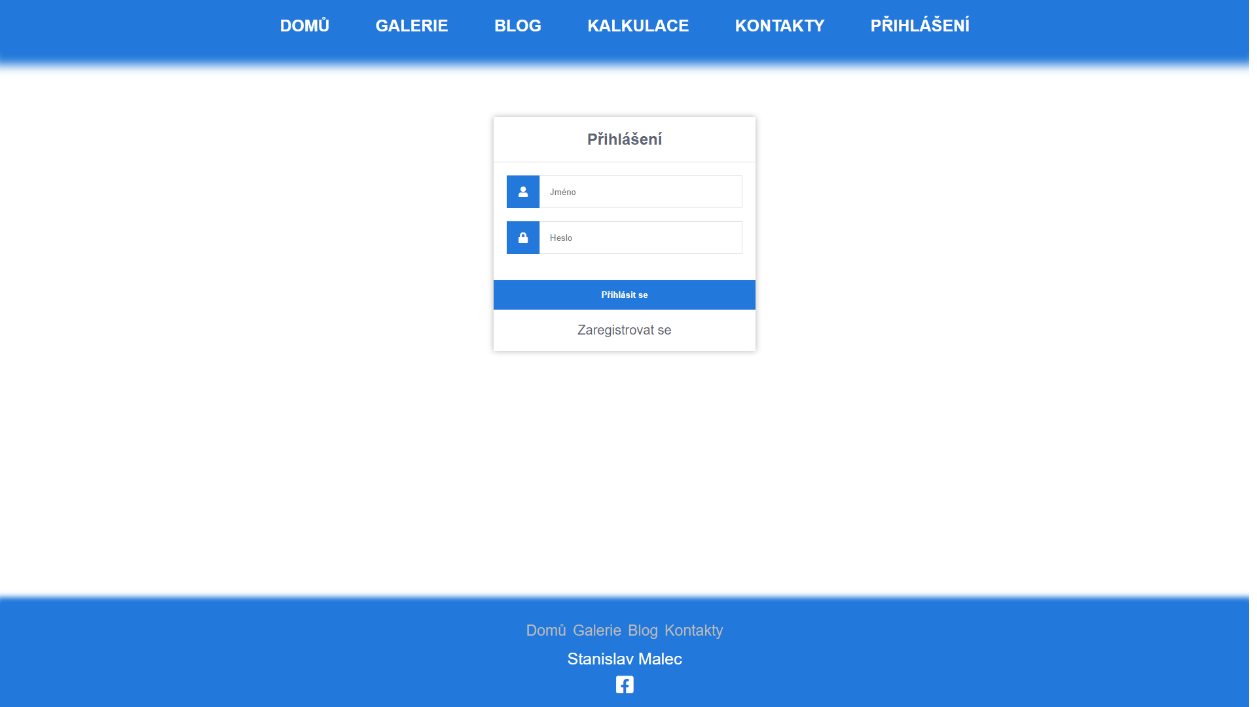
Dalším segmentem na této stránce je formulář pro kontaktování.

Obr. . Kontaktní formulář

V tomto formuláři uživatelé vyplní své jméno, emailovou adresu, předmět zprávy a samotnou zprávu či otázku. Tento formulář je následně poslán správci webu. Tento formulář funguje za pomocí použití Javascriptu kdy za pomocí objektu FormData skript získává data z formuláře což umožňuje snadný přístup k hodnotám jednotlivých polí formuláře bez nutnosti manuálně je extrahovat. Dále skript postupně přidává do proměnné emailBody údaje jako jméno, e-mail, předmět a text zprávy, přičemž každou položku odděluje znakem nového řádku (\n). Poté je email poslán pomocí příkazu mailto.

## Přihlášovací formulář (login.php)

Tato stránka slouží k přihlášení uživatelů. Je převážně určena pro přihlášení uživatele s administrátorským přístupem.

Na této stránce se nachází formulář, který obsahuje přihlašovací údaje uživatele (jméno a heslo).

Obr. . Přihlašovací stránka

Pokud přihlášení proběhne úspěšně je uživatel přesměrován na úvodní stránku. V případě špatně zadaných údajů se vypíše chybová zpráva. Pokud uživatel nemá účet je zde umístěn text „Zaregistrovat se“, který uživatele přesměruje na registrační formulář, viz. Kapitola 2.8.

### Přihlášení (authenticate.php)

Tento kód slouží k přihlášení uživatele.

Inicializace a Připojení k Databázi: Skript začíná načtením třídy Database a inicializací objektu databáze, což umožňuje následné dotazování do databáze.

Session Start: Volání session\_start() zajišťuje, že je možné pracovat se session proměnnými, což je zásadní pro uchovávání stavu mezi různými požadavky HTTP.

Zpracování POST Požadavku: Skript nejprve ověří, zda byl formulář odeslán metodou POST. Pokud ne, uživatel je přesměrován zpět na přihlašovací formulář.

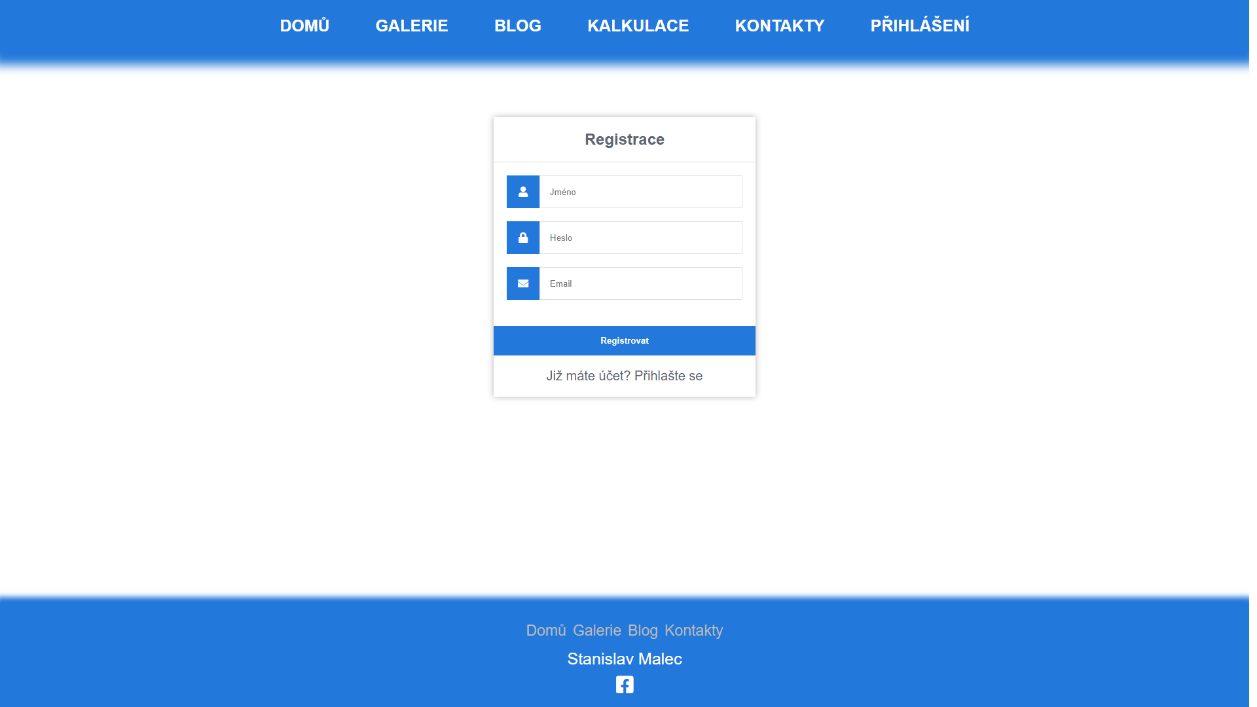
Validace Vstupů: Skript zkontroluje, zda byla vyplněna obě pole formuláře (uživatelské jméno a heslo). Pokud některé z polí zůstane prázdné, uživatel je upozorněn na chybu a je přesměrován zpět na přihlašovací stránku.

Ověření Uživatelského Jména a Hesla: Skript dále přistoupí k databázi a vyhledá záznam odpovídající zadanému uživatelskému jménu. Pokud je uživatel nalezen   
a heslo ověřeno pomocí funkce password\_verify(), jsou do session proměnných uloženy údaje o úspěšném přihlášení a uživatelském jménu. Uživatel je poté přesměrován na hlavní dashboard.

Ošetření Chybné Autentikace: Pokud ověření hesla nebo uživatelského jména selže, je uživatel upozorněn na chybu prostřednictvím session proměnné error a je přesměrován zpět na přihlašovací stránku.

Bezpečnostní aspekty - kód důkladně zajišťuje bezpečnost procesu autentikace. Použití password\_verify() pro ověření hesla znamená, že hesla jsou v databázi uložena v zašifrované formě, což zvyšuje bezpečnost uložených údajů. Kontroly vstupů   
a zpracování chyb také přispívají k ochraně proti bezpečnostním hrozbám, jako je SQL injection či brute force útoky.

## Registrační formulář (reg.php)

Tato stránka slouží k registraci uživatelů.

Obr. . Registrační stránka

Funguje na stejné bázi jako přihlašovací formulář s tou změnou, že k údajům pro vyplnění je přidáno pole pro vyplnění emailové adresy. Pokud již uživatel má vytvořený účet, nachází se pod formulářem text „Již máte účet? Přihlašte se“, který po kliknutí přesměruje uživatele na stránku s přihlašovacím formulářem, viz. Kapitola 2.7.

### Registrace (register.php)

Sanitizace a Validace Vstupů: - Uživatelské vstupy jsou nejprve sanitizovány   
a validovány. Funkce *filter\_var()* se používá k odstranění nebezpečných znaků   
a ověření správnosti e-mailové adresy.

Hashování Hesla - heslo zadané uživatelem je zahashováno pomocí funkce *password\_hash()*, což zajišťuje bezpečné uložení hesla v databázi.

Vložení Záznamu do Databáze - poté, co jsou vstupy ověřeny a zpracovány, je nový uživatel zaregistrován v databázi. SQL příkaz *INSERT INTO* vloží nový záznam   
s uživatelským jménem, zahashovaným heslem a e-mailovou adresou do tabulky users.

uložení přihlášení do session proměnných - po úspěšné registraci je uživatel automaticky přihlášen tím, že jsou do session proměnných uloženy údaje o úspěšném přihlášení (loggedin a username).

Uživatel je nakonec přesměrován na dashboard.

Bezpečnostní Aspekty:

Sanetizace a validace vstupů – fukcefiler\_var()   
a FILTER\_SANITIZE\_FULL\_SPECIAL\_CHARS slouží k odstranění nebezpečných znaků a filtrování neplatných e-mailových adres.

Hashování hesla - heslo je zahashováno pomocí funkce password\_hash(), což zabraňuje odhalení hesel v případě, že databáze je kompromitována.

Použití prepared statements - SQL dotazy jsou vytvářeny pomocí připravených příkazů (prepared statements), což chrání před SQL injection útoky.

## Ověření přihlášení (dashboard.php)

Tento kód slouží pro ověření, zda uživatel byl přihlášen.

Obr. . Kód pro ověření přihlášení

Kód nejprve ověří, za uživatel je přihlášen pomocí podmínky   
*if (!isset($\_SESSION['loggedin']))* , pokud není přihlášen, je kód ukončen pomocí příkazu *exit*. V případě, že uživatel je přihlášen je přesměrován na úvodní stránku.

## Propojení s databází (Database.php)

Tento kód louží pro propojení s databází kdy definuje třídu Database, která zjednodušuje proces připojení k databázi. Třída obsahuje proměnné pro údaje potřebné k připojení k databázi a metodu \_\_construct(), která se automaticky vyvolá při vytváření instance této třídy.

Obr. . Kód pro propojení s databází

Proměnné pro Připojení:

$host - obsahuje adresu serveru databáze, ve většině případů se jedná o 'localhost', což značí, že databázový server běží na stejném serveru jako webová aplikace.

$db - jméno databáze, ke které se aplikace snaží připojit.

$user - uživatelské jméno pro přístup k databázi.

$pass - heslo pro přístup k databázi.

$conn - veřejná proměnná, která bude obsahovat objekt reprezentující připojení   
k databázi po úspěšném navázání spojení.

Konstruktor

Metoda \_\_construct() slouží k inicializaci objektu. V rámci této metody se sestaví připojovací řetězec a vytvoří se nový objekt PDO, který naváže spojení s databází. Při vytváření objektu PDO jsou nastaveny i jeho atributy pro chybový režim a výchozí režim pro výsledky dotazů:

PDO::ATTR\_ERRMODE - Tento atribut je nastaven na PDO::ERRMODE\_EXCEPTION, což znamená, že PDO vyhodí výjimku PDOException v případě chyby. Toto nastavení umožňuje lepší zpracování chyb a jejich ladění.

PDO::ATTR\_DEFAULT\_FETCH\_MODE - Nastavení tohoto atributu na PDO::FETCH\_ASSOC znamená, že výsledky dotazů budou vráceny jako asociativní pole.

Závěr

Webová stránka dokáže přidávání příspěvků do blogu, jejich úpravu a jejich smazání. Dále nabízí fotogalerii s možností přidávání fotografií a jejich smazání. Také nabízí vzorové kalkulace položek, jejich nahrávání, úpravu a mazání. Pro informace o firmě je zde sekce pro všechny důležité informace jako jsou telefonní číslo, emailová adresa a adresa s přidanou možností pro kontaktování firmy za pomoci kontaktního formuláře. Webová stránka rovněž obsahuje log-in systém, který zde převážně slouží pro administrátora webu pro možnosti úprav a přidávání.

V této práci jsem dosáhl svých veškerých stanovených cílů.

Díky tomuto projektu jsem si více osvojil práci v jazyce PHP a rovněž práci s databázemi.

Z mého pohledu bude webová stránka používaná, jelikož původní webové stránky byli pouze statické bez možnosti přidávání příspěvků pro správce.

Co bych jiného přidal je web hosting, jelikož v momentální fázi webová stránka funguje pouze lokálně. Na web hostingu se v tuto chvíli pracuje a v nejbližší době bude implementován. To přinese nejen certifikovanou ochranu webu, ale i možnost upravení kontaktního formuláře, aby nebyla používána aplikace Microsoft Outlook.

Seznam použitých zdrojů

1. PÍSEK, Slavoj. *HTML: začínáme programovat*. Online. 4., aktualiz. vyd. Průvodce (Grada). Praha: Grada, 2014. ISBN ﻿ISBN978-80-247-5059-0. Dostupné z: [https://books.google.cz/books?id=tLamAgAAQBAJ&printsec=frontcover&key=AIzaSyDlPfI89JdFhWBVsMVsavVo6aNh057xlTc#v=onepage&q&f=false](https://books.google.cz/books?id=tLamAgAAQBAJ&printsec=frontcover&key=AIzaSyDlPfI89JdFhWBVsMVsavVo6aNh057xlTc%23v=onepage&q&f=false). [cit. 2023-12-29].
2. KOĎOUSKOVÁ, Barbora. *KASKÁDOVÉ STYLY PRO ZAČÁTEČNÍKY ANEB JAK ZAČÍT S CSS*. Online. Rascasone. Rascasone, 2021. Dostupné z: <https://www.rascasone.com/cs/blog/kaskadove-styly-css-jak-zacit>. [cit. 2023-12-29].
3. *Úvod do PHP*. Online. Web Tvorba. Web Tvorba, © 2004. Dostupné z: http://www.webtvorba.cz/php/uvod-do-php.html. [cit. 2023-12-30].
4. *PhpMyAdmin*. Online. 2003. Dostupné z: [https://www.phpmyadmin.net](https://www.phpmyadmin.net/). [cit. 2023-12-30].
5. YAQUB, Mohhamad. *Data Management: Databases - MySQL Introduction for Researchers*. Online. University of Oxford. 2020. Dostupné z: <https://skills.it.ox.ac.uk/sites/default/files/skills/documents/media/dm006w_mysql_introduction_online_oct20.pdf>. [cit. 2023-12-30].
6. *PhpMyAdmin.* Online. In: Installatron. Installatron, © 2004 - 2023. Dostupné z: https://installatron.com/phpmyadmin?locale=sv. [cit. 2024-01-06].
7. *SQL (Structured query language) Injection.* Online. Imperva. Imperva, © 2024. Dostupné z: https://www.imperva.com/learn/application-security/sql-injection-sqli/. [cit. 2024-01-07].
8. *SQL Injection*. Online. W3Schools. W3Schools, © 2024. Dostupné z: <https://www.w3schools.com/sql/sql_injection.asp>. [cit. 2024-01-10].
9. *What is SQL Injection (SQLi) and How to Prevent It*. Online. Acunetix. Acunetix, © 2024. Dostupné z: <https://www.acunetix.com/websitesecurity/sql-injection/>. [cit. 2024-01-07].
10. Kirsten*S. Cross Site Scripting (XSS).* Online. OWASP. OWASP, © 2024. Dostupné z: https://owasp.org/www-community/attacks/xss/. [cit. 2024-01-09].
11. *Cross-site scripting.* Online. PortSwigger. PortSwigger, © 2024. Dostupné z: https://portswigger.net/web-security/cross-site-scripting. [cit. 2024-01-09].
12. *Brute Force Attack*. Online. Imperva. Imperva, © 2024. Dostupné z: [https://www.imperva.com/learn/application-security/brute-force-attack](https://www.imperva.com/learn/application-security/brute-force-attack/)/. [cit. 2024-01-10].
13. *Understanding Brute Force Attacks and How to Stop Them in Their Tracks.* Online. Threat Intelligence. Threat Intelligence, © 2024. Dostupné z: <https://www.threatintelligence.com/blog/brute-force>. [cit. 2024-01-18].
14. *What is a denial of service attack (DoS) ?* Online. Palo Alto Networks. Palo Alto Networks, © 2024. Dostupné z: <https://www.paloaltonetworks.com/cyberpedia/what-is-a-denial-of-service-attack-dos>. [cit. 2024-01-20].

Seznam použitých symbolů a zkratek

HTML HyperText Markup Language

CSS Cascading Style Sheets

PHP Hypertext Preprocessor

SQL The Structured Query Language

RDBMS Relational Database Management Systém

GUI Grafické uživatelské rozhraní

SQLi SQL Injection

XSS Cross-Site Scripting

Seznam obrázků

[Obr. 1.1.1 phpMyAdmin prostředí [6] 11](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466370)

[Obr. 2.1.1 Domovská stránka 20](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466371)

[Obr. 2.3.1 Stránka blogu 22](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466372)

[Obr. 2.3.2 Stránka pro úpravu příspěvku 23](#_Toc162466373)

[Obr. 2.4.1 Stránka galerie 24](#_Toc162466374)

[Obr. 2.4.2 Kontrola za je soubor obrázek 25](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466375)

[Obr. 2.4.3 Kontrola za soubor již existuje 26](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466376)

[Obr. 2.4.4 Kontrola velikosti souboru 26](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466377)

[Obr. 2.4.5 Kontrola typu souboru 27](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466378)

[Obr. 2.4.6 Nahrání obrázku 27](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466379)

[Obr. 2.5.1 Stránka pro vzorové kalkulace 29](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466380)

[Obr. 2.5.2 Stránka pro kalkulace s administrátorskými právy 30](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466381)

[Obr. 2.6.1 Panel s informacemi 31](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466382)

[Obr. 2.6.2 Kontaktní formulář 32](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466383)

[Obr. 2.7.1 Přihlašovací stránka 33](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466384)

[Obr. 2.8.1 Registrační stránka 35](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466385)

[Obr. 2.9.1 Kód pro ověření přihlášení 36](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466386)

[Obr. 2.10.1 Kód pro propojení s databází 37](file:///C:\Users\Vojtěch%20Mastný\Desktop\Mastný-%20maturitní%20prácePos.docx#_Toc162466387)