# Technická dokumentace – UmZor

# Stack

UmZor pro frontend používá HTML, CSS a JS. Pro backend byl využit jazyk PHP. Všechna data se ukládají do souborů JSON (kromě fotek). Fotky jsou uloženy v samostatné složce Photos.

## Formát JSON souborů

**users.json**

V users.json se nacházejí data ohledně uživatelů (jejich jméno, email, hashované heslo a role). V databázi jsou uloženi uživatelé takto:

[{"username": "Majda", "email": "lebedmag@cvut.cz", "password": "$2y$10$H54wkdeHxVbgJMAptqkMwuX\/9QiQnEp4SkPH7ScfQ2GAlYOH5eZY2", "role":"admin"},{"username": "Smrad1","email": "smrad@smrad.cz","password": "$2y$10$oiEUIOj\/dsPHbJKN3MDkM.iY1sbqvSPoEKJ4uSoHbp3ET2VFmFpv.", "role": "user"}]

**prispevky.json**

V prispevky.json jsou uložená všechna data z formuláře (kromě fotky) a náhodně vygenerované ID pro snazší párování fotky a údajů z formuláře.

[{"id": "a5ea694a6cbac902","author": "Majda","artname": "\/><script> alert","technique": "Pastel","timePost": "23:48 05.01.2025"},{"id": "a6b389c5de11223f","author": "Majda","artname": "duha","technique": "Tužka","timePost": "23:49 05.01.2025"}]

**photos:** V této složce se ukládají fotky. Fotky se zde ukládají pod stejným názvem jako je ID přispěvku, který se uloží do prispevky.json.

## CSS

V souboru style.css se upravuje: obecné nastavení stránky, navigační bar, formát příspěvků, posuvné šipky (slider), formát fotek, stránkování, formuláře, tabulka, chybové hlášky, rozložení webu na různých zařízení, tisk stránky

## JS

JS souborů je několik, všechny validují vstupní pole (kromě index.js)

## index.js

Soubor index.js je zodpovědný za správu a funkčnost obrázkových sliderů, které se zobrazují u příspěvků na hlavní stránce. Tento kód umožňuje uživatelům procházet více obrázků připojených k jednotlivým příspěvkům.

**Funkčnost kódu:**

1. **Inicializace sliderů:** Po načtení celé stránky je spuštěn skript pomocí události DOMContentLoaded. Vyhledají se všechny HTML prvky s třídou slider-container, které obsahují data o obrázcích.
2. **Načtení dat:**Pomocí atributů data-photosdir a data-photos jsou načteny informace o složce s obrázky a názvech jednotlivých souborů (data-photosdir: Definuje adresář, kde jsou obrázky uloženy (např. *photos*), data-photos: Obsahuje JSON seznam názvů souborů obrázků.)
3. **Prvky slideru:** leftArrow (tlačítko pro posun doleva), rightArrow (tlačítko pro posun doprava), imgTag (HTML element <img> pro zobrazení aktuálního obrázku)
4. **Aktualizace obrázku:** Funkce updateImage(index) nastavuje zdroj obrázku (src) na základě aktuálního indexu obrázku a složky s obrázky.

**Klíčové prvky kódu:**

* **Funkce updateImage(index):** Sestavuje URL každého obrázku na základě složky s fotkami a názvu souboru.
* **Event listener:** Šipky mají přiřazené události click, které mění aktuální index obrázku a volají funkci updateImage() pro aktualizaci obrázku v slideru.
* **Cyklické procházení:** Posun vlevo a vpravo je implementován jako cyklický, což znamená, že pokud uživatel dosáhne posledního obrázku, posun ho vrátí na první a naopak.

## addPrispevek.php

Stránka addPrispevek.php obsahuje formulář pro přidání nového příspěvku, který se odesílá metodou POST. Data se ukládají do databáze ve formě JSON souboru (prispevky.json), zatímco fotografie jsou ukládány do složky photos. Každý příspěvek získá jedinečné ID, které je generováno náhodně a používá se jak pro záznam v JSON souboru, tak jako nový název nahraných fotografií.

Formulář na stránce obsahuje několik polí:

1. **Fotografie (photo)** – Uživatel může nahrát minimálně jednu a maximálně čtyři fotografie ve formátu JPG nebo PNG. Každá fotografie musí splňovat kritéria velikosti.
2. **Autor (autorname)** – Skryté pole, které automaticky vyplňuje jméno autora na základě aktuální session přihlášeného uživatele.
3. **Název díla (artname)** – Pole (text), kde uživatel zadává název svého díla. Maximální délka názvu je 16 znaků, minimální 3.
4. **Technika (technique)** – Povinné pole typu výběr (dropdown), kde uživatel vybírá jednu z předdefinovaných uměleckých technik.
5. **Datum a čas nahrání (timePost)** – Skryté pole, které automaticky zaznamenává aktuální datum a čas včetně hodin a minut. Tento údaj se ukládá v předem definovaném formátu: hodiny:minuty, den.měsíc.rok.

Pokud je formulář správně vyplněn, příspěvek se uloží do JSON databáze a fotografie do složky photos. Po úspěšném nahrání příspěvku je uživatel přesměrován na hlavní stránku (index.php).

## changePrispevek.php

Na stránce changePrispevek.php se uživateli zobrazí formulář, jehož pole jsou předvyplněna údaji z databáze prispevky.json. Uživatel nebo admin může upravit název díla a zvolenou uměleckou techniku. Ostatní pole, jako je jméno autora (autorname) a datum přidání (timePost), jsou pevná a není možné je měnit. Pod formulářem se nachází tlačítko pro odstranění příspěvku, které smaže záznam z databáze prispevky.json a odstraní s ním spojené fotografie uložené ve složce photos.

Formulář musí validovat tyto údaje:

1. **Název příspěvku (artname)**: musí být vyplněn a mít alespoň 3 znaky a nesmí přesáhnout 16 znaků.
2. **Umělecká technika (technique)**: uživatel musí vybrat jednu z předdefinovaných hodnot.

Pokud uživatel při vyplňování formuláře udělá chybu (např. nezadá název díla nebo zvolí neplatnou techniku), objeví se chybová hláška pod formulářem a uživatel může chybu opravit. Po odeslání formuláře probíhá kontrola CSRF tokenu pro zajištění bezpečnosti. Pokud uživatel zvolí možnost „Uložit“, provedené úpravy se uloží do databáze prispevky.json a uživatel bude rovněž přesměrován na hlavní stránku.

## checkAvailability.php

Stránka checkAvailability.php je kód, který slouží k ověření dostupnosti uživatelského jména nebo e-mailu v databázi users.json. Tento skript pracuje na základě AJAX požadavků, které mu předávají údaje prostřednictvím parametrů v URL. Ověření je prováděno na serverové straně a vrací odpověď ve formátu JSON.

Soubor je navržen tak, aby jej bylo možné volat z formuláře registrace. Díky tomu uživatel dostane okamžitou zpětnou vazbu, zda je uživatelské jméno nebo e-mail již obsazeno, aniž by bylo nutné odesílat celý formulář.

**Jak checkAvailability.php funguje:**

1. **Požadavek AJAX**: Klient odesílá požadavek na checkAvailability.php s parametry field (pole, které se má zkontrolovat: username/email) a value (hodnota, která se má ověřit)
2. **Načtení databáze uživatelů**: Skript načte data z JSON souboru users.json. Pokud soubor neexistuje nebo je prázdný, vyhodnotí, že daná hodnota je volná.
3. **Kontrola dostupnosti**: Pokud je hodnota pole field nastavena na username, skript prohledá seznam uživatelů a ověří, zda uživatelské jméno již existuje nebo pokud je pole field nastavena na email, ověří se, zda je e-mail již použit.
4. **Generování odpovědi**: Pokud hodnota není nalezena, odpověď bude:

json

{ "valid": true }

* + Pokud je hodnota již obsazena, odpověď bude obsahovat:

json

{ "valid": false, "message": "Uživatelské jméno je již obsazeno." }

1. **Odpověď JSON**  
   Soubor vrátí výstup v JSON formátu (rychlé propojení s JS).

**Bezpečnostní a výkonnostní aspekty:**

* **XSS ochrana:** Skript pracuje pouze se zpracovanými hodnotami, které jsou kontrolovány a trimovány. Veškeré parametry, které neodpovídají očekávání, jsou ignorovány.
* **Výkon:** Databáze uživatelů je načítána jednorázově a procházena jednoduchou iterací, což zajišťuje rychlou odezvu.
* **Bezpečnost:** Skript vrací pouze informace o dostupnosti hodnoty. Nikdy neposkytuje citlivé údaje, jako je seznam existujících uživatelů nebo e-mailů.

Tento soubor je klíčový pro funkčnost webu: pomocí checkAvailability.php klient nemusí odesílat formulář, aby zjistil, jestli je jeho username „dostupné“ či ne.

## config.php

Soubor config.php slouží pro správu uživatelských dat. Obsahuje funkce a nastavení, které umožňují načítání a ukládání uživatelských údajů z/do souboru users.json. Tento soubor zajišťuje, aby data byla správně spravována a bezpečně uložena, a to v souladu s JSON formátem.

**Popis funkcionality:**

1. **Cesta k databázovému souboru:**
   * Soubor users.json se nachází ve stejném adresáři jako tento skript. Pokud není přítomen, skript hlásí chybu a ukončí se.
2. **Načítání uživatelských dat:**
   * Funkce load\_users slouží k načtení uživatelských dat ze souboru users.json.
   * Data jsou načtena jako text, následně dekódována pomocí funkce json\_decode.
   * Pokud dojde k chybě při dekódování JSON dat, aplikace zobrazí odpovídající chybovou zprávu.
3. **Ukládání uživatelských dat:**
   * Funkce save\_users umožňuje ukládat uživatelská data zpět do souboru users.json.
   * Data jsou nejprve kódována do formátu JSON pomocí funkce json\_encode.
   * Při kódování dat se používá příznak JSON\_PRETTY\_PRINT, který formátuje data do čitelné podoby, a JSON\_UNESCAPED\_UNICODE (podpora pro české znaky)
   * V případě chyby při ukládání dat se zobrazí odpovídající chybová zpráva.

**Jak tento skript funguje v aplikaci:**

1. **Volání funkce load\_users:**
   * Tato funkce se používá ve vypisUzivatelu.php
2. **Volání funkce save\_users:**
   * Tato funkce je využívána při úpravě uživatelů. Po úspěšné validaci vstupních dat je seznam uživatelů aktualizován a uložen zpět do souboru users.json.

## configPhotos.php

Soubor configPhotos.php je klíčovým kódem, který slouží k zpracování a uložení nahraných fotografií uživatelem. Tento soubor vykonává validaci, úpravu a ukládání obrázků, přičemž zajišťuje jejich kompatibilitu a správnou velikost pro vykreslování do index.php.

**Popis funkcionality:**

1. **Omezení počtu fotografií:**
   * Skript kontroluje, aby uživatel nenahrával více než 4 fotografie najednou. Pokud počet souborů překročí limit, zobrazí se chybová hláška:  
     *Nahráváte příliš mnoho fotek (max. 4).*
2. **Validace každého nahraného souboru:**
   * Pro každou fotografii se provádí následující kontroly:
     + **Chyba nahrávání:** Pokud došlo k chybě při nahrávání souboru, zobrazí se chybová hláška pro konkrétní soubor.
     + **Velikost souboru:** Maximální povolená velikost je 4 MB. Pokud je soubor větší, zobrazí se zpráva:  
       Fotky, které se snažíte nahrát přesahují 4M. Zkuste nahrát prosím menší fotku.
     + **Formát souboru:** Jsou podporovány pouze formáty JPEG a PNG. Pokud soubor neodpovídá těmto formátům, zobrazí se zpráva:  
       Soubor není JPG ani PNG.
3. **Zpracování a úprava fotografií:**
   * Soubor otevírá každý obrázek jako grafický zdroj (JPEG nebo PNG) pomocí funkcí imagecreatefromjpeg nebo imagecreatefrompng.
   * Fotografie jsou upravovány na velikost 600x600 pixelů (zachován poměr stran a přidá se bílá výplň).
4. **Uložení fotografií:**
   * Upravené fotografie jsou ukládány do zadané složky ($photosDir) pod názvy, které obsahují unikátní id a index fotografie (např. id-0.jpg).
5. **Výstupní seznam uložených fotografií:**
   * Funkce vrací pole obsahující názvy všech úspěšně uložených fotografií. Pokud nastane chyba během nahrávání některé fotografie, tato fotka se přeskočí a vznikne chybová zpráva ($errors).

**Průběh zpracování obrázků:**

1. **Převzít nahrané soubory:**
   * Data nahraných souborů ($\_FILES['photo']) se předají funkci processPhotos spolu s odkazem na pole chyb $errors, cestou ke složce pro ukládání obrázků a id názvu souboru.
2. **Validace:**
   * Každý soubor se ověřuje.
3. **Úprava obrázků:**
   * Obrázky jsou přizpůsobeny na požadovanou velikost a zarovnány do čtverce s bílým pozadím (600x600 px).
4. **Uložení:**
   * Upravené obrázky jsou ukládány do složky Photos s generovanými názvy.

## footer.php

Soubor, který v sobě obsahují krátký text. Nachází se v každé viditelné stránce uživatelem.

## headerHtml.php

Soubor, který kontroluje role uživatele a pomocí toho jim ukazuje odkazy na ostatní stránky.

**Funkčnost kódu:**

1. **Startování session**:
   * Pokud session ještě nebyla inicializována, kód ji automaticky spustí pomocí session\_start().
2. **Získání role a uživatelského jména**:
   * Role uživatele je získána ze session pomocí $\_SESSION['role']. Pokud role není nastavena, výchozí hodnota je 'visitor'.
   * Uživatelovo jméno je získáno přes $\_SESSION['username']. Pokud není nastaveno, zobrazí se 'Host'.
3. **Generování navigace**:
   * Navigační menu je strukturováno pomocí HTML elementu <nav> a <ul>.
   * Každá role má přizpůsobené odkazy:
     + **Visitor (nepřihlášený)**:
       - Odkaz na hlavní stránku (index.php).
       - Odkaz na přihlašovací stránku (login.php) místo možnosti Nahrát.
       - Odkaz na přihlašovací stránku (login.php) místo profilu.
     + **User (přihlášený)**:
       - Odkaz na hlavní stránku (index.php).
       - Odkaz na přidání příspěvku (addPrispevek.php).
       - Odkaz na profil (profil.php).
     + **Admin**:
       - Kromě všech možností usera se přidává odkaz na stránku vypisUzivatelu.php, kde administrátor může vidět seznam všech uživatelů.

## Index.php

Díky navigačnímu menu v header.php se uživatelé mohou snadno dostat na index.php, hlavní stránku aplikace UmZor. Na této stránce jsou zobrazovány všechny příspěvky uložené v databázi prispevky.json, které mohou být řazeny, stránkovány a zobrazeny se spárovanými obrázky.

**Funkčnost kódu:**

1. **Session management**:
   * Kód začíná inicializací **session**, která umožňuje správu stavu uživatele, jako je přihlášení nebo určení role (např. admin, uživatel, návštěvník).
2. **Definice pomocné funkce escape**:
   * Tato funkce zajišťuje, že veškeré uživatelské vstupy nebo data z databáze budou „očištěny“ a nebudou obsahovat škodlivý kód (XSS útok).
3. **Načítání příspěvků**:
   * Příspěvky jsou načteny z JSON souboru **prispevky.json** pomocí funkce file\_get\_contents a dekódovány do pole.
   * Příspěvky jsou seřazeny podle času přidání (timePost) od nejnovějších po nejstarší pomocí funkce usort.
4. **Stránkování příspěvků**:
   * Stránka zobrazuje maximálně 4 příspěvky na jedné stránce.
   * Pomocí proměnných currentPage a totalPages je zajištěno stránkování s odkazy na předchozí a následující stránky.
5. **Zobrazení příspěvků**:

* Každý příspěvek obsahuje název díla (pole artname), jméno autora (author), techniku (technique), datum a čas přidání (timePost) a fotky (složka photos).
* Příspěvky jsou zobrazovány pomocí foreach loopů

1. **Podpora více obrázků**:
   * Pokud příspěvek obsahuje více než jeden obrázek, je automaticky vytvořen slider, který uživatelům umožňuje procházet jednotlivé obrázky pomocí šipek.
2. **Upravit příspěvek**:
   * Pokud je uživatel přihlášen a je buď admini, nebo autor daného příspěvku, zobrazí se odkaz „Upravit“, který přesměruje uživatele na stránku **changePrispevek.php** pro úpravu příspěvku.
3. **Přehled příspěvků**:
   * Pokud nejsou v databázi žádné příspěvky, zobrazí se zpráva: „Žádné příspěvky k zobrazení.“
4. **Stránkování**:
   * Odkazy na první, aktuální, předchozí a následující stránku jsou dynamicky generovány na základě aktuálního čísla stránky a celkového počtu stránek.

## login.php

Pomocí přihlašovacího formuláře v login.php se uživatelé mohou přihlásit na web a získat přístup k funkcím, které vyžadují přihlášení. Tento soubor obsahuje jak PHP skript, který zajišťuje ověření uživatelských údajů, tak HTML strukturu samotného přihlašovacího formuláře.

**Funkčnost kódu:**

1. **Session Management**
   * Skript začíná inicializací session (uchovává info o uživateli během procházení stránky).
   * Do session se při úspěšném přihlášení ukládají hodnoty jako username (uživatelské jméno) a role (admin nebo user).
2. **Funkce pro bezpečnost: escape, csrf\_token, check\_csrf\_token**
   * **escape**: Ošetřuje výstup do HTML proti XSS útokům.
   * **csrf\_token**: Generuje CSRF token, který zajišťuje, že formulář může být odeslán pouze z aplikace samotné (ne cizí zdroje).
   * **check\_csrf\_token**: Kontroluje, zda CSRF token odeslaný formulářem odpovídá tomu, který je uložen v session.
3. **Ověření přihlašovacích údajů:**
   * Po odeslání formuláře je pomocí metody POST přijato uživatelské jméno a heslo.
   * Pokud je jedno z polí prázdné, zobrazí se chyba: „Uživatelské jméno a heslo jsou povinné.“
   * Pokud jsou pole vyplněná, načte se soubor users.json, který obsahuje informace o registrovaných uživatelích.
4. **Ověření uživatele:**
   * Login.php prochází seznam uživatelů v souboru users.json.
   * Ověří, zda uživatelské jméno odpovídá a zda se heslo shoduje pomocí funkce password\_verify.
   * Pokud ověření proběhne úspěšně, uživatel je přesměrován na index.php.
5. **Zobrazení chybové hlášky:**
   * Pokud je zadáno nesprávné uživatelské jméno nebo heslo, zobrazí se chybová hláška: „Špatně zadané uživatelské jméno nebo heslo.“

**HTML část:**

1. **Formulář pro přihlášení:**
   * Obsahuje pole pro uživatelské jméno a heslo.
   * Pole pro **CSRF** token je skryté a automaticky generované.
   * Zobrazuje chybové zprávy v případě, že přihlašovací proces selže.
2. **Odkaz na registraci:**
   * Pokud uživatel nemá účet, může se zaregistrovat pomocí odkazu na register.php.

## Logout.php

Stránka logout.php zajišťuje odhlášení uživatele z aplikace. Tento proces zahrnuje ukončení session, což odstraní uložené informace o přihlášení uživatele, a následné přesměrování na hlavní stránku. Tento jednoduchý kód obsahuje také základní ochranu proti XSS útokům pomocí funkce escape.

**Funkčnost kódu:**

1. **Session management**
   * Na začátku skriptu se inicializuje session pomocí funkce **session\_start()**, což je nezbytné pro přístup k uloženým údajům, jako jsou **username** nebo **role**.
2. **Funkce escape**
   * Pokud ještě není definovaná funkce **escape**, skript ji vytvoří. Tato funkce ošetřuje speciální znaky v řetězci pro bezpečné použití v HTML a chrání před XSS útoky.
3. **Zpracování požadavku POST:**
   * Když je metoda HTTP požadavku POST, session se ukončí pomocí:
     + session\_unset(): Odstraní všechny proměnné uložené v session.
     + session\_destroy(): Zničí celou session.
   * Po ukončení session je uživatel přesměrován na hlavní stránku (index.php) pomocí funkce header("Location: index.php").

## Profil.php

Stránka **profil.php** slouží k zobrazení profilu přihlášeného uživatele. Na této stránce si uživatel může prohlédnout své údaje (uživatelské jméno, email a role). Součástí stránky je možnost se odhlásit.

**Funkčnost kódu:**

1. **Správa session:**
   * Na začátku skriptu se inicializuje session pomocí session\_start() (info o username).
   * Pokud session neobsahuje proměnnou username, uživatel je přesměrován na stránku login.php pomocí funkce header().
2. **Funkce escape:**
   * Skript obsahuje funkci escape() (ochrana proti XSS útokům).
3. **Načtení a ověření dat uživatelů:**
   * Data o uživatelích jsou uložena v souboru users.json. Profil.php načítá obsah tohoto souboru a dekóduje jej do podoby asociativního pole.
   * Pokud soubor neexistuje, nebo se data nepodaří dekódovat, zobrazí se chybová zpráva.
   * Uživatel, který je aktuálně přihlášen, je identifikován podle jeho uživatelského jména (username) uloženého v session.
4. **Zobrazení profilu:**
   * Pokud je přihlášený uživatel nalezen v databázi, zobrazí se jeho uživatelské jméno, email a role.
5. **Možnost odhlášení:**
   * Na stránce je formulář s tlačítkem Odhlásit se, který odešle POST požadavek na stránku logout.php, kde je zajištěno odhlášení uživatele.

**Struktura HTML:**

* + Hlavní nadpis stránky Profil uživatele, zobrazení údajů o uživateli (uživatelské jméno, email, role) uvnitř formuláře, tlačítko pro odhlášení.

## Register.php

Stránka register.php umožňuje uživatelům vytvořit si nový účet. Tato stránka obsahuje formulář, který validuje uživatelské údaje a ukládá je do souboru users.json. Uživatelé, kteří dokončí registraci, jsou automaticky přihlášeni a přesměrováni na hlavní stránku.

**Funkčnost kódu:**

1. **Správa session:**
   * Na začátku skriptu je inicializována session pomocí session\_start()
2. **CSRF ochrana:**
   * Každá stránka obsahuje jedinečný CSRF token uložený v session (použit v poli jako hidden).
   * Funkce check\_csrf\_token() kontroluje, zda token v POST požadavku odpovídá tokenu uloženému v session.
3. **Funkce escape:**
   * Funkce escape() zabezpečuje uživatelské vstupy proti XSS útokům
4. **Validace uživatelských dat:**
   * Skript kontroluje:
     + Uživatelské jméno (délka mezi 3–30 znaky, unikátnost), email (validní formát a unikátnost), heslo (shoda s potvrzením hesla).
   * V případě chyby se zobrazí odpovídající chybová hláška vedle pole, kde k chybě došlo.
5. **Uložení dat:**
   * Pokud jsou všechna data validní, tak je heslo zabezpečeno pomocí funkce password\_hash(), nový uživatel je přidán do pole (uloženo do users.json).
6. **Automatické přihlášení:**
   * Po úspěšné registraci je uživatelské jméno a role uloženo do session.
   * Uživatel je přesměrován na hlavní stránku index.php.

**Struktura HTML:**

* + Formulář obsahuje uživatelské jméno, email, heslo, potvrzení hesla
  + Pole jsou validována na straně serveru i klienta.

1. **Odkaz na přihlášení:**
   * Pokud uživatel již má účet, je mu nabídnut odkaz na přihlášení na stránku login.php.

## Vypisuzivatelu.php

Stránka **vypisUzivatelu.php** je určena pro administrátory a umožňuje správu uživatelských účtů v aplikaci. Administrátor má možnost zobrazit všechny uživatele, měnit jejich role (uživatel/admin) a spravovat jejich základní informace.

**Funkčnost kódu:**

1. **Správa session:**
   * Na začátku kódu se spustí session pomocí session\_start() (info o useru).
2. **Ověření role:**
   * Pokud uživatel nemá roli admin, je automaticky přesměrován na index.php.
3. **Zpracování formuláře:**
   * Při odeslání formuláře pro změnu role:
     + Skript kontroluje platnost CSRF tokenu pomocí funkce check\_csrf\_token().
     + Upraví roli vybraného uživatele na základě ID a nové hodnoty role.
     + Aktualizuje soubor users.json pomocí funkcí z konfiguračního souboru config.php.
     + Zobrazuje úspěšné nebo chybové hlášky podle výsledku operace.
4. **Ochrana proti XSS:**
   * Všechny uživatelské vstupy jsou zabezpečeny funkcí escape()

**Struktura HTML:**

* + Nadpis Správa uživatelů pro přehlednost stránky.
  + Zobrazení tabulky všech uživatelů se sloupci ID uživatele, uživatelské jméno, email, role (uživatel/admin), akce (form na změnu role).

1. **Formulář pro úpravu role:**
   * Každý řádek obsahuje formulář, který umožňuje administrátorovi vybrat novou roli uživatele.
   * Formulář obsahuje CSRF token pro ochranu, skryté pole s ID uživatele, dropdown menu pro výběr role (user/admin), tlačítko pro odeslání změny.

**Technická dokumentace – UmZor**

**Stack**

UmZor využívá pro frontend HTML, CSS a JavaScript. Backend je implementován v PHP. Data se ukládají do JSON souborů, fotografie jsou uloženy ve složce photos.

**Formát JSON souborů**

**users.json**: Obsahuje informace o uživatelích, včetně jmen, emailů, hashovaných hesel a rolí (např. admin nebo user).

[

{"username": "Majda", "email": "lebedmag@cvut.cz", "password": "<hash>", "role": "admin"},

{"username": "Smrad1", "email": "smrad@smrad.cz", "password": "<hash>", "role": "user"}

]

**prispevky.json**: Obsahuje příspěvky s náhodně generovanými ID, autory, názvy děl, technikami a časem přidání.

[

{"id": "a5ea694a6cbac902", "author": "Majda", "artname": "duha", "technique": "Pastel", "timePost": "23:48 05.01.2025"}

]

**photos**: Fotografie jsou uloženy pod stejnými ID jako příspěvky v JSON souboru.

**CSS**

Soubor style.css definuje:

* Obecné nastavení vzhledu stránky,
* Styl navigačního baru, formulářů a tabulek,
* Responsivitu pro různá zařízení,
* Slider na fotky a tiskovou úplavu.

**JavaScript**

**index.js**

Zajišťuje funkčnost sliderů pro fotky u příspěvku.

1. **Inicializace**: Načítá data o fotkách z atributů HTML (např. data-photosdir).
2. **Cyklický posun**: Kliknutím na tlačítka šipek se fotky posouvají dokola.
3. **Aktualizace obrazu**: Funkce updateImage mění zdroj zobrazeného obrázku.

**Ostatní JS**

Validační skripty pro vstupy uživatelských formulářů.

**Hlavní PHP soubory**

**addPrispevek.php**

Stránka umožňuje přidání nových příspěvku.

1. **Formulář**: Obsahuje pole pro fotky (minimálně 1, maximálně 4, formáty JPG/PNG), název díla (3–16 znaků) a techniku (povinný výběr).
2. **Validace**: Zajišťuje kontrolu správného formátu vstupů.
3. **Uložení dat**: Příspěvek je uložen do prispevky.json, fotografie do photos.
4. **Přesměrování**: Po úspěchu je uživatel přesměrován na hlavní stránku.

**changePrispevek.php**

Formulář umožňuje změnu názvu díla a techniky u existujícího příspěvku.

* Validace: Kontroluje se správný formát vstupů.
* Odstranění: Obsahuje tlačítko pro smazání příspěvku včetně fotografií.
* Bezpečnost: Chráněno CSRF tokenem.

**checkAvailability.php**

Skript kontroluje dostupnost uživatelských jmén a emailů v reálném čase pomocí AJAX.

* Odpověď JSON: Vrácí, zda hodnota je/není dostupná.
* Bezpečnost: Ošetření vstupů proti XSS.

**config.php**

Spravuje datové soubory.

* load\_users: Načte uživatelská data z users.json.
* save\_users: Ukládá změny do souboru a podporuje formátování JSON.

**configPhotos.php**

Validuje a ukládá fotografie.

1. **Validace**: Kontroluje formát, počet (max. 4) a velikost (max. 4 MB).
2. **Zpracování**: Fotky jsou upraveny na 600x600 px, zachovávají poměr stran.
3. **Uložení**: Fotky jsou uloženy se specifickým ID.

**headerHtml.php**

Generuje navigační menu na základě role uživatele.

* Role visitor: Odkazy na přihlášení a hlavní stránku.
* Role user/admin: Možnost přidávat příspěvky a spravovat profil.

**footer.php**

Obsahuje společný text zobrazený ve spodní části každé stránky.

**Další soubory**

**index.php**

Hlavní stránka zobrazující příspěvky.

1. **Načtení dat**: Příspěvky se načtou z prispevky.json, jsou seřazeny podle data.
2. **Stránkování**: Zobrazuje se max. 4 příspěvky na stránku.
3. **Slider**: Umožňuje prohlížení více fotek.

**login.php**

Stránka pro přihlášení. Ošetřuje vstupy a kontroluje uživatele v users.json.

* Přihlášení: Při úspěšné autentizaci se uživatel přesměrovává na hlavní stránku.
* Chyby: Zobrazují se při nesprávných datech.

**logout.php**

Odhlášení uživatele ukončí session a přesměruje na hlavní stránku.

**register.php**

Registrace uživatelů. Po validaci a úspěšném uložení dat do users.json se uživatel přihlásí automaticky.

**vypisUzivatelu.php**

Administrátorská stránka pro správu uživatelů.

* Změna rolí: Umožňuje změnu role (user/admin).
* Bezpečnost: Chráněno CSRF tokenem.