

# Dokumentace pro burzu učebnic



Jméno: Alexander

Příjmení: Vršek

Datum: 27.4.2025

Škola: Gymnázium Sokolov a Krajské vzdělávací centrum, příspěvková organizace

# Obsah

Obsah .....	2
Úvod .....	6
1. Použité technologie .....	7
1.1. Frontend .....	7
1.1.1. Co to je.....	7
1.1.2. Použití Tailwind CSS .....	8
1.1.3. Ukázka jednoduchosti Tailwindu .....	9
1.1.4. Responzivní design.....	10
1.1.5. Interaktivní komponenty .....	10
1.1.6. Integrace s backendem.....	10
1.2. Backend .....	11
1.2.1. Co to je.....	11
1.2.2. Použité technologie a nástroje .....	11
1.2.3. Jak backend funguje v praxi .....	11
1.2.4. Práce s databází.....	12
1.2.5. SQL dotazy.....	12
1.2.6. Přihlašování přes Google OAuth.....	12
1.2.7. Základní bezpečnostní opatření.....	13
1.2.8. Ukázka backendového kódu.....	13
2. Architektura a návrh systému .....	14
2.1. Struktura souborů .....	14
2.1.1. Grafická reprezentace .....	14
2.2. Struktura databáze .....	16
2.2.1. Tabulky.....	16
2.2.1.1. Tabulka user .....	16
2.2.1.2. Tabulka učebnice .....	16
2.2.1.3. Tabulka typ .....	17
Tabulka kategorie.....	17
2.2.1.4. Tabulka pu .....	17
2.2.1.5. Tabulka orders .....	17

2.2.2.	Vztahy mezi tabulkami.....	18
2.2.3.	Ukázka interakce.....	19
3.	Klíčové části kódu.....	20
3.1.	Google OAuth 2.0 .....	20
3.1.1.	Úvod a OAuth údaje.....	20
3.1.2.	Přesměrování na Google autorizaci .....	20
3.1.3.	Kontrola přihlášení uživatele .....	21
3.1.4.	Příprava parametrů .....	21
3.1.4.1.	Před odesláním požadavku na Google .....	21
3.1.4.2.	Odeslání požadavku na Google API .....	22
3.1.4.3.	Zpracování odpovědi .....	22
3.1.5.	Získání dat o uživateli .....	23
3.1.5.1.	Ověření a použití přístupového tokenu .....	23
3.1.5.2.	Získání údajů o uživatelském účtu .....	23
3.1.5.3.	Zpracování odpovědi a uložení údajů o uživatelském účtu.....	24
3.1.6.	Práce s databází – kontrola a vložení uživatele .....	24
3.1.6.1.	Získání výsledku z databáze .....	24
3.1.6.2.	Pokud uživatel neexistuje .....	25
3.1.7.	Uzavření spojení s databází .....	25
3.2.	Zpracování objednávek.....	26
3.2.1.	Načítání závislostí a inicializace .....	26
3.2.2.	Funkce getBookDetail.....	26
3.2.2.1.	Inicializace a příprava SQL dotazu .....	26
3.2.2.2.	Provedení SQL dotazu .....	27
3.2.3.	Funkce getBookImages.....	27
3.2.4.	Funkce processOrder .....	28
3.2.4.1.	Kontrola platnosti uživatele .....	28
3.2.4.2.	Načtení detailů knihy.....	28
3.2.4.3.	Získání aktuálního casu a zahájení transakce .....	28
3.2.4.4.	Aktualizace stavu knihy v databázi .....	28
3.2.4.5.	Vložení objednávky do databáze.....	29
3.2.4.6.	Spuštění transakce a potvrzení úspěchu.....	29

3.2.5.    FunkceOrderByPulID .....	29
3.3.    Nahrávání fotek.....	30
3.3.1.    HTML část.....	30
3.3.1.1.    HTML formulář a zobrazení existujících fotek .....	30
3.3.1.2.    Sekce náhledu existujících obrázků.....	30
3.3.1.3.    Sekce pro výběr a náhled nových fotek .....	31
3.3.2.    PHP část.....	32
3.3.2.1.    Funkce getExistingImages .....	32
3.3.2.2.    Smazání uživatelem vybraných obrázků.....	32
3.3.2.3.    Funkce uploadNewImages .....	33
3.3.2.3.1.    Vytvoření složky pro nové obrázky.....	33
3.3.2.3.2.    Určení cesty.....	33
3.3.2.3.3.    Konverze a uložení fotky .....	34
3.3.2.3.4.    Zpracovávání chyb .....	35
3.3.2.4.    Vyvolání nahrávací funkce .....	35
3.3.2.5.    Aktualizace seznamu pro opětovné zobrazení .....	35
3.3.3.    Javascript část .....	36
3.3.3.1.    Manipulace s nahráváním nových souborů a jejich zobrazením .....	36
3.3.3.2.    Vykreslení náhledů souborů .....	37
3.3.3.3.    Odstranění souborů z náhledu .....	38
3.3.3.4.    Odstranění existujících obrázků .....	38
3.3.3.5.    Uložení odstraněných obrázků před odesláním formuláře .....	39
4.    Vývoj .....	40
4.1.    Úvod .....	40
4.2.    Vývoj.....	40
4.2.1.    Visual Studio Code.....	40
4.2.2.    GitHub .....	42
5.2.3.    XAMMP.....	43
5.2.4.    Google Cloud.....	44
4.3.    Výzvy při vývoji.....	45
5.2.1.    Přihlášení .....	45
5.2.2.    Kompatibilita se smratphony.....	46

6.	Testování a nasazení .....	47
4.4.	Testování .....	47
4.5.	Nasazení .....	48
7.	Závěr .....	49
7.1.	Shrnutí .....	49
7.2.	AI .....	50
7.3.	Možnosti rozšíření.....	51
8.	Přílohy .....	52
8.1.	Obrázky.....	52
8.2.	Tabulky.....	52
8.3.	Kódy.....	52
	Zdroje: .....	54

# Úvod

Projekt "Burza učebnic" je webová aplikace zaměřená na zjednodušení procesu prodeje a nákupu učebnic mezi studenty. Cílem je vytvořit platformu, kde si studenti mohou jednoduše nabídnout své učebnice na prodej, nebo naopak najít učebnice, které potřebují k zakoupení. Aplikace poskytuje uživatelům přehled o aktuálních nabídkách a objednávkách, umožňuje jim potvrzovat a stornovat transakce a jednoduché uživatelské rozhraní. Dále obsahuje i funkční administrátorské prostředí, skrz které je možné řídit chod celé aplikace přímo z prohlížeče.

Pro realizaci tohoto projektu byly zvoleny technologie, které umožňují efektivní správu databáze, zabezpečený přístup uživatelů a moderní a responzivní uživatelské rozhraní. Využití PHP pro serverovou část aplikace, MySQL pro správu dat a Google OAuth pro autentizaci uživatelů a Tailwind pro vzhled se ukázalo jako ideální volba pro dosažení požadované funkcionality a bezpečnosti.

Projekt zároveň reaguje na potřebu jednoduché a přístupné platformy pro sdílení a prodej učebnic mezi studenty, čímž usnadňuje nejen nákup, ale i prodej učebnic po jejich využití a oproti fyzické burze je tohle jednoduší, levnější pro všechny, jelikož nikdo nemá žádnou provizi a méně časově náročné pro všechny strany.

V této seminární práci bude popsáno, jaké technologie a nástroje byly použity pro realizaci projektu, jakým způsobem byly implementovány klíčové funkce aplikace, a proč byly zvoleny právě tyto technologie. Dále se zaměříme na vysvětlení praktické implementace, jako je autentizace pomocí Google OAuth, užívání Tailwind CSS pro design a spolupráce mezi PHP a MySQL.

# 1. Použité technologie

## 1.1. Frontend

### 1.1.1. Co to je

Frontend (nebo také **client-side**) je část webové aplikace, kterou uživatelé vidí a s kterou přímo interagují. Jedná se o veškeré vizuální a interaktivní prvky, které se zobrazují na obrazovce, jako jsou texty, obrázky, tlačítka, formuláře, odkazy a další obsah, se kterým uživatel pracuje. Frontend je zodpovědný za prezentaci a uživatelský zážitek (UX) na webových stránkách nebo v aplikacích.

Frontend se skládá ze tří hlavních technologií:

1. **HTML** (HyperText Markup Language)
  - Struktura stránky.
  - HTML definuje obsah stránky (texty, obrázky, odkazy, tabulky, seznamy, formuláře) a jeho hierarchii. HTML je kostrou každé webové stránky.
2. **CSS** (Cascading Style Sheets) – Stylování stránky.
  - CSS se používá k definování vzhledu webové stránky, jako je rozložení, barvy, písma, mezery, pozadí a další vizuální aspekty. Pomocí CSS se stránka stává atraktivní a přehledná.
3. **JavaScript** – Interaktivita stránky.
  - JavaScript je jazyk, který umožňuje webovým stránkám reagovat na akce uživatelů, jako je klikání na tlačítka, zadávání údajů do formulářů nebo animace. JavaScript dává stránkám interaktivní prvky a dynamiku.

The screenshot shows a web application for buying and selling books. At the top, there's a navigation bar with links like 'Prodávané' (Available), 'Kupované' (Bought), 'Nápojováda', 'Dokumentace - Preview', 'Administrace', and 'Odhlašit se'. Below the navigation, there are two sections: 'Čekající objednávky' (Pending orders) and 'Dokončené objednávky' (Completed orders). Each section has a table with columns for ID, Name, Price, Status, and Purchase. There are also buttons for 'Přejít' (Go) and 'Podrobnosti' (Details). Below these tables, there are three boxes: 'O projektu' (About the project), 'Kontakt' (Contact), and 'Podpora' (Support). The 'O projektu' box contains text about the project being created for internal purposes by Gymnázium Sokolov a Krajské vzdělávací centrum, with contact information for Alexander Vršek (email: roddy55yt@gmail.com). The 'Kontakt' box contains a quote from George R.R. Martin: '„Člověk, který čte, žije tisice životů, než zemře. Člověk, který nečte, žije jen jeden.“ — George R.R. Martin'. The 'Podpora' box contains a message encouraging users to buy quickly to support the project.

Obrázek 1 - Stránka s CSS

This screenshot shows the same website as above, but with a different CSS style. The layout is more compact and lacks some visual separation between sections. The tables for pending and completed orders are still present, along with the 'O projektu', 'Kontakt', and 'Podpora' boxes. The contact quote and support message are also visible.

Obrázek 2 - Stránka bez CSS

### 1.1.2. Použití Tailwind CSS

Tailwind CSS je nástroj, který zjednodušuje tvorbu uživatelského rozhraní tím, že poskytuje sadu předem definovaných tříd pro běžně používané styly, jako jsou barvy, rozměry, písmo, zarovnání a další. Hlavní výhody použití Tailwindu v tomto projektu jsou:

- **Rychlosť vývoje:** Díky množství užitečných utility tříd lze rychle implementovat design bez nutnosti psát rozsáhlé CSS.
- **Responzivita:** Tailwind umožňuje snadné vytváření responzivních designů pomocí breakpointů. To znamená, že aplikace se automaticky přizpůsobí různým velikostem obrazovek (mobil, tablet, desktop).
- **Přizpůsobitelnost:** Tailwind umožňuje snadnou změnu stylů přímo v HTML souborech, což zjednodušuje úpravy vzhledu bez nutnosti přepisovat externí CSS soubory.

Při použití Tailwindu pro "Burzu učebnic" byly využity utility třídy pro:

- **Rozvržení:** Flexbox a grid pro responzivní rozložení obsahu.
- **Typografie:** Standardizování písma, velikostí a mezer pro jednotný vzhled.
- **Barvy a pozadí:** Použití barevného schématu, které je příjemné pro oči a zajišťuje dobrou čitelnost textů.
- **Interaktivní prvky:** Stylování tlačítek, formulářů a dalších prvků pro zlepšení použitelnosti.

Proč jsem si vybral Tailwind:

- **Zkušenosti:** Je to jediná CSS knihovna, se kterou jsem na začátku vývoje měl zkušenosti.
- **Jednoduchost:** Tailwind je velice jednoduchý a intuitivní, takže po velmi krátké době si většinu příkazů zapamatujete.
- **Dokumentace:** Tailwind má velmi hezky a velmi přehledně zpracovanou [dokumentaci](#), takže si vše jednoduše můžete vyhledat.



Obrázek 3 - Ukázka dokumentace Tailwindu

### 1.1.3. Ukázka jednoduchosti Tailwindu

Na obrázku níže můžete vidět prvek, který jsem vytvořil jak pomocí Tailwindu, tak pomocí čistého CSS. Rozdíl v délce kódu můžete porovnat sami. Ale i přes to bych volil CSS pro projekty většího rozsahu s větším důrazem na vzhled.

## Online Burza Učebnic

Obrázek 4 - Hlavička burzy učebnic

```
<div class="w-full max-w-5xl mx-auto">
    <!-- Záhlaví -->
    <div class="flex items-center justify-center bg-white shadow-md p-5 rounded-md">
        <!-- Nadpis -->
        <h1 class="text-2xl sm:text-3xl font-bold text-center text-gray-800 hover:underline">
            <a href="index.php">Online Burza Učebnic</a>
        </h1>
    </div>
</div>
```

Kód 1 - Ukázka kódu pro hlavičku pro Tailwind

```
<div class="header-container">
    <!-- Záhlaví -->
    <div class="header">
        <!-- Nadpis -->
        <h1 class="header-title">
            <a href="index.php">Online Burza Učebnic</a>
        </h1>
    </div>
    .header-container {
        width: 100%;
        max-width: 64rem;
        margin: 0 auto;}
.header {
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #ffffff;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    padding: 1.25rem;
    border-radius: 0.375rem;
    height: 100%;}
.header-title {
    font-size: 1.25rem;
    font-weight: 700;
    text-align: center;
    color: #2d3748;
    text-decoration: none;}
.header-title a {
    color: inherit;
    text-decoration: none;}
.header-title a:hover {
    text-decoration: underline;}
@media (min-width: 640px) {
    .header-title {font-size: 1.875rem;}}
```

Kód 2 - Ukázka kódu pro hlavičku pro CSS

Tailwind

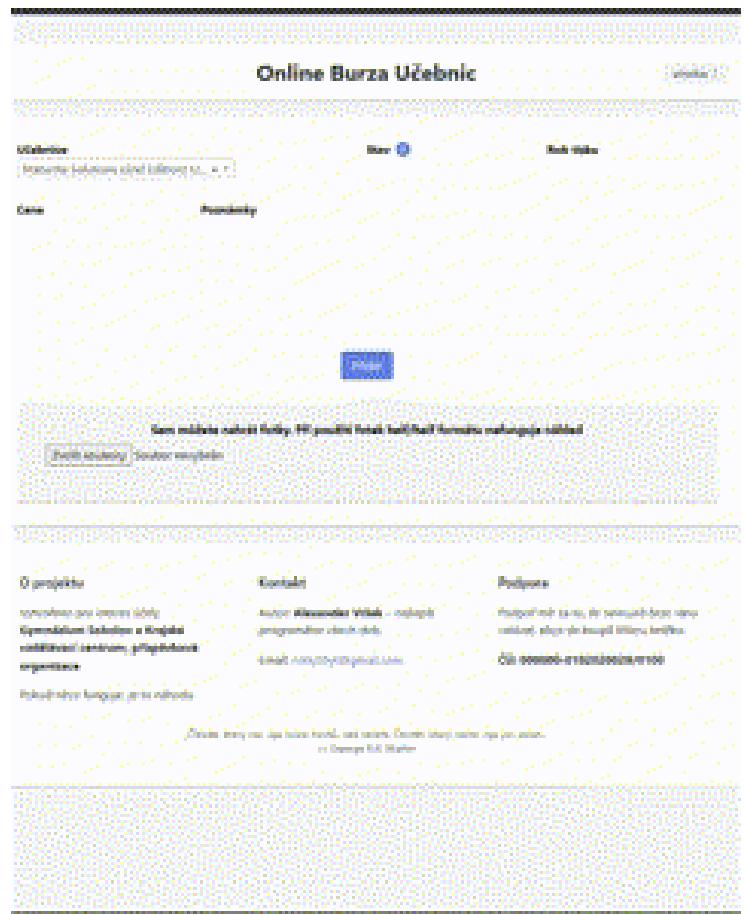
CSS

Tabulka 1 - Porovnání kódu Tailwindu a CSS

#### 1.1.4. Responzivní design

Responzivita je klíčovým prvkem uživatelského rozhraní. S použitím Tailwindu byla aplikace navržena tak, aby byla přístupná jak na mobilních telefonech, tak na tablettech a desktopových zařízeních. Každá stránka aplikace, ať už se jedná o profil uživatele, nabídky učebnic nebo objednávky, je optimalizována pro různé velikosti obrazovek. Například:

- Na malých obrazovkách (mobilní zařízení) jsou všechny prvky uspořádány vertikálně, aby usnadnily navigaci a interakci.
- Na větších obrazovkách (desktop) se aplikace automaticky přizpůsobí a zobrazuje více informací v horizontálním rozvrzení, což maximalizuje využití dostupného prostoru.



Obrázek 5 - Ukázka responzivního designu  
GIF – V PDF se nezobrazí.

#### 1.1.5. Interaktivní komponenty

Frontend aplikace zahrnuje i interaktivní komponenty, které jsou důležité pro uživatelský zážitek. Tyto komponenty zahrnují:

- **Filtry a hledání:** Uživatelé mohou vyhledávat učebnice podle názvu, ročníku nebo stavu, což zjednodušuje orientaci v nabídce.
- **Formuláře pro zadání a úpravu nabídek:** Formuláře jsou navrženy tak, aby byly jednoduché a přehledné, s validací vstupních dat, aby se předešlo chybám při zadávání informací.
- **Potvrzování a storno objednávek:** Uživatelské rozhraní umožňuje snadné potvrzování a storno objednávek pomocí tlačítek a dialogových oken, které zajišťují intuitivní ovládání.

#### 1.1.6. Integrace s backendem

Frontend je propojený s backendem aplikace, který zajišťuje správu dat a komunikaci s databází. Data o nabídkách, objednávkách a uživatelských profilech jsou dynamicky načítána z databáze pomocí PHP a zobrazována na stránkách aplikace.

## 1.2. Backend

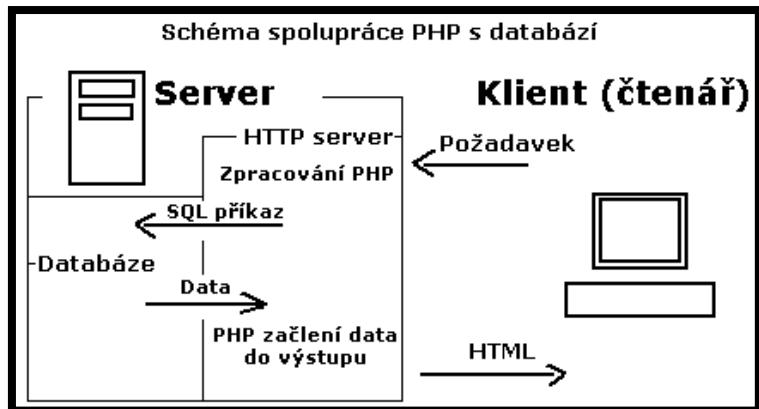
### 1.2.1. Co to je

Zatímco frontend představuje to, co uživatel přímo vidí a s čím pracuje, backend je všechno, co se odehrává „v zákulisí“ aplikace.

Je to neviditelná, ale klíčová část systému, která zajišťuje, že všechno funguje správně – od načítání učebnic až po správu objednávek a bezpečné přihlašování uživatelů.

Backend se stará o zpracování dat, komunikaci s databází, provádění logiky aplikace a odesílání odpovědí zpět na frontend.

Bez něj by byla stránka statická a nemohla by reagovat na žádné akce uživatele.



### 1.2.2. Použité technologie a nástroje

Pro vytvoření backendu Burzy učebnic jsem zvolil kombinaci **PHP** a **MySQL**.

- **PHP** je serverový skriptovací jazyk, který umožňuje dynamicky generovat webové stránky a pracovat s daty. Je jednoduchý na použití, široce podporovaný a skvěle se hodí pro menší až střední projekty, jako je tento.
- **MySQL** slouží jako databázový systém, který bezpečně ukládá všechny potřebné informace – uživatelské účty, nabídky učebnic, objednávky a další data.

Celý systém funguje tak, že uživatel provede nějakou akci (například odešle formulář), PHP tuto akci na serveru zpracuje, komunikuje s databází a podle výsledku vrátí odpověď zpět do prohlížeče.

### 1.2.3. Jak backend funguje v praxi

Když uživatel na webu například zadá novou nabídku učebnice, celý proces vypadá přibližně takto:

1. Uživatel vyplní formulář a odešle ho.
2. PHP na serveru zkонтroluje, jestli jsou všechna potřebná data správně vyplněná.
3. Pokud ano, PHP uloží údaje o nové učebnici do databáze pomocí SQL dotazu.
4. Uživatel dostane zpátky odpověď (např. přesměrování na stránku s potvrzením).

Podobně fungují i další operace jako přihlašování, úprava profilu, vytváření objednávek nebo potvrzování prodeje.

#### 1.2.4. Práce s databází

Databáze je srdcem celé aplikace – bez ní by nebylo možné spravovat nabídky ani sledovat objednávky.

Tabulky v databázi obsahují například:

- **Uživatele** – informace o registrovaných osobách (jméno, email, ročník studia).
- **Učebnice** – seznam dostupných učebnic k prodeji včetně ceny, stavu a popisu.
- **Objednávky** – propojení kupujících, prodávajících a konkrétních učebnic.

Pomocí PHP se pak k těmto datům přistupuje, aktualizují se nebo se podle nich generují seznamy a přehledy.

#### 1.2.5. SQL dotazy

Veškerou práci s databází prováděte skrz SQL dotazy a PHP se stará o jejich provádění a vyhodnocení výsledků. SQL dotazy říkají databázi co, kde, jak, s jakými daty dělat.

```
SELECT u.username, COUNT(o.id) AS order_count
Vybere sloupec 'username' z tabulky 'users' a spočítá počet objednávek (id) z
tabulky 'orders', přičemž výsledek pojmenuje jako 'order_count'.
FROM users u
Určuje tabulku 'users' jako hlavní tabulkou a přiřazuje jí alias 'u'.
LEFT JOIN orders o ON u.id = o.user_id
Provede LEFT JOIN mezi tabulkami 'users' a 'orders', kde se párují záznamy podle
'id' uživatele v tabulce 'users' a 'user_id' v tabulce 'orders'.
GROUP BY u.username
Seskupí výsledky podle uživatelského jména ('username'), aby bylo možné spočítat
počet objednávek pro každého uživatele.
ORDER BY order_count DESC
Seřadí výsledky podle počtu objednávek ('order_count') sestupně, tedy od největšího
k nejmenšímu.
```

Kód 3 - Ukázka SQL dozazu

#### 1.2.6. Přihlašování přes Google OAuth

Pro pohodlné a bezpečné přihlašování jsme integroval **Google OAuth 2.0**. Díky tomu se uživatelé nemusí registrovat klasicky přes formulář a pamatovat si další heslo – stačí se přihlásit pomocí svého Google účtu.

Celý proces funguje tak, že:

- Uživatel klikne na tlačítko „Přihlásit se pomocí Google“.
- Přesměruje se na oficiální Google přihlašovací stránku.
- Po úspěšném přihlášení Google vrátí serveru potřebné údaje (např. email a jméno).
- PHP následně ověří, jestli uživatel existuje v databázi, a pokud ne, vytvoří mu nový účet.

To celé výrazně zvyšuje bezpečnost i uživatelský komfort.

### 1.2.7. Základní bezpečnostní opatření

Aby byla aplikace chráněná před běžnými typy útoků a chyb, implementoval jsem několik jednoduchých bezpečnostních pravidel:

- **Validace vstupů** – veškerá data, která přicházejí od uživatele, jsou kontrolována (např. kontrola správného formátu emailu, omezení maximální délky textů).
- **Správa session** – přihlášení je spravováno pomocí PHP session a cookie, což brání neoprávněnému přístupu k chráněným stránkám.
- **Ochrana proti SQL Injection** – při práci s databází používám funkce, které automaticky escapují potenciálně nebezpečné znaky. Mezi hlavní ochranu patří předpřipravené (stmt) dotazy, které umožní provést jen předpřipravený dotaz.

Bezpečnost je důležitá i u malých projektů, protože každá chyba může znamenat únik dat nebo ztrátu funkčnosti systému.

### 1.2.8. Ukázka backendového kódu

Pro představu, jak jednoduše některé části backendu fungují, příkladám krátkou ukázku PHP kódu, který ukládá novou knihu do databáze:

```
function pridatKnihu($conn, $data, $userId) {
    $id_ucebnice = intval($data['id_ucebnice']);
    $stav = intval($data['stav']);
    $rok_tisku = intval($data['rok_tisku']);
    $cena = intval($data['cena']);
    $poznamky = $conn->real_escape_string($data['poznamky']);
    // Přidání knihy do tabulky `pu`
    $stmt = $conn->prepare("INSERT INTO pu (id_ucebnice, id_prodejce, rok_tisku,
    stav, cena, koupil, poznamky) VALUES (?, ?, ?, ?, ?, 0, ?)");
    if ($stmt) {
        $stmt->bind_param("iiiiis", $id_ucebnice, $userId, $rok_tisku, $stav,
        $cena, $poznamky);
        $stmt->execute();
        $puID = $stmt->insert_id; // Získání ID nově přidaného záznamu
        $stmt->close();
    } else {
        return [
            "msg" => "✗ Chyba při přípravě dotazu: " . $conn->error,
            "error" => true
        ];
    }
    return [
        "msg" => "Učebnice byla úspěšně přidána!",
        "puID" => $puID,
        "error" => false
    ];
}
```

Kód 4 - Ukázka funkce pro vložení nové knihy

Tento jednoduchý skript ukazuje, jak backend přijímá data od uživatele, pracuje s databází a podle výsledku vrací odpověď.

## 2. Architektura a návrh systému

### 2.1. Struktura souborů

Celé jsem to strukturoval velmi jednoduše, ale přehledně.

- V kořenové složce mám nejdůležitější soubory, jako je hlavně úvodní stránka, hlavička, zápatí.
- Do složky pages jsem zařadil veškeré stránky, kam se uživatel může dostat.
- Ve složce code jsou veškeré soubory zodpovědně za logiku.
- Ve složce foto jsou veškeré fotky

Tuto strukturu jsem si vybral pro svoji jednoduchost a přehlednost a v případě chyb je hned jasné, kde co hledat.

#### 2.1.1. Grafická reprezentace

```
/burza      – Kořen repozitáře
├── banned.html    – Stránka, která se zobrazí zablokovaným uživatelům
├── burza.sql     – Databázový dump pro nastavení databáze burzy.
├── footer.php    – Spodní část webu (společná pro více stránek).
├── footeradmin.php – Spodní část webu pro administrátory.
├── header.php    – Hlavička webu (společná pro více stránek).
├── index.php     – Hlavní stránka aplikace.
|
├── pages/        – Jednotlivé stránky aplikace.
|   ├── edit.php    – Stránka pro úpravu záznamů.
|   ├── edit_row.php – Stránka pro úpravu záznamu v adminském rozhraní.
|   ├── koupit.php   – Stránka pro nákup knih.
|   ├── nova.php    – Stránka pro přidání nové knihy.
|   ├── objednavka.php – Stránka pro zobrazení konkrétní objednávky.
|   ├── objednavky.php – Stránka pro zobrazení všech objednávek.
|   ├── prodat.php   – Stránka pro prodej knih.
|   ├── profil.php   – Stránka uživatelského profilu.
|   ├── superadmin.php – Stránka pro administrátory
|   ├── ucebnice.php  – Stránka pro zobrazení konkrétní učebnice.
|   └── user.php     – Stránka pro zobrazení cizího uživatele.
|
```

code/	- Serverová logika a API endpointy.
book.php	- [ucebnice.php] PHP skript pro zobrazení všech knih pro danou učebnici
buy.php	- [koupit.php] Zpracování nákupní akce.
connect.php	- Nastavení připojení k databázi.
edit_book.php	- [edit.php] Úprava existující nabídky.
index_logic.php	- [index.php] Serverová logika pro hlavní stránku.
is_admin.php	- [superadmin.php] Kontrola administrátorských práv.
is_banned.php	- Kontrola, zda je uživatel zabanovaný.
login.php	- Přihlášení uživatele přes OAuth2.
logout.php	- Odhlášení uživatele.
new.php	- [nova.php] Přidání nové učebnice.
order.php	- [objednavka.php] Zobrazení konkrétní objednávky.
orders.php	- [objednavky.php] Seznam objednávek.
profile.php	- [profil.php] Zobrazení a úprava uživatelského profilu.
row_edit.php	- [edit_row.php] PHP skript pro úpravu záznamu v adminském rozhraní.
sell.php	- [prodat.php] Zpracování prodejní akce.
super_admin.php	- [superadmin.php] Funkce pro administrátora.
user_php.php	- Načítání informací o cizím uživateli.
userinfo.php	- Detaily uživatelů (získává userID pro práci s databází).
book.js	- [ucebnice.php] JavaScript pro zobrazení vlastních knih
new.js	- [nova.php] JavaScript pro změnu náhledu fotky.
orders.js	- [objednavky.php] JavaScript pro zobrazení objednávek.
row_edit_pu.js	- [edit_row.php] JavaScript pro úpravy v adminském rozhraní pro tabulku knih.
row_edit_ucebnice.js	- [edit_row.php] JavaScript pro úpravy v adminském rozhraní pro tabulku učebnic.
sell.js	- [prodat.php] JavaScript pro zobrazení náhledu fotek.
upravit_knihu.js	- [edit.php] JavaScript pro zobrazení náhledu fotek.
foto/	- Složka pro ukládání obrázků.
ucebnice /	- Složka pro fotky obalů učebnic.
pu/	- Složka pro složky fotek jednotlivých knih.

Tabulka 2 - Grafická reprezentace struktury souborů

## 2.2. Struktura databáze

Struktura databáze pro projekt „Burza učebnic“ je navržena tak, aby byla efektivní, škálovatelná a snadno rozšířitelná. Využívá relační model, kde jsou data rozdělena do několika vzájemně propojených tabulek. Každá tabulka má svůj konkrétní účel a obsahuje atributy, které odpovídají potřebám aplikace. Následuje přehled hlavních tabulek a jejich struktury.

### 2.2.1. Tabulky

#### 2.2.1.1. Tabulka user

Tato tabulka obsahuje informace o všech uživatelích, kteří se připojí k platformě. Její struktura vypadá následovně:

- **id**: Primární klíč, unikátní ID uživatele. (INT, PK, AUTO\_INCREMENT)
- **user**: Uživatelské jméno, část e-mailu před zavináčem. (VARCHAR(32))
- **email**: Email uživatele. (VARCHAR(32))
- **jmeno**: Křestní jméno uživatele. (VARCHAR(32))
- **přijmení**: Příjmení uživatele. (VARCHAR(32))
- **trida\_id**: Ročník uživatele. 1 = 6. třída ZŠ, 8 = 4. třída SŠ. (INT)
- **type**: Typ uživatele. 0 = standardní, 0 > zabanovaný, 1 = admin, 2 < superadmin (INT)

#### 2.2.1.2. Tabulka učebnice

Tato tabulka obsahuje informace o všech učebnicích, které jsou dostupné na platformě. Její struktura vypadá následovně:

- **id**: Primární klíč, unikátní ID učebnice. (INT, PK, AUTO\_INCREMENT)
- **jmeno**: Název učebnice. (VARCHAR(128))
- **kategorie\_id**: ID kategorie, do které učebnice patří (např. matematika, chemie). (INT)
- **typ\_id**: Typ učebnice (např. učebnice, pracovní sešit). (INT)
- **trida\_id**: Ročník, pro který je učebnice určena. 1 = 6. třída ZŠ, 8 = 4. třída SŠ. (INT)
- **schvaleno**: Stav schválení učebnice. Nemá žádnou funkci. (TINYINT(1))

### **2.2.1.3. Tabulka typ**

Tato tabulka obsahuje seznam typů učebnic, které lze na platformě prodávat. Její struktura vypadá následovně:

- **id**: Primární klíč, unikátní ID typu. (INT, PK, AUTO\_INCREMENT)
- **nazev**: Název typu učebnice (např. učebnice, pracovní sešit, kniha). (VARCHAR(32))

### **Tabulka kategorie**

Tato tabulka obsahuje seznam jednotlivých kategorií učebnic podle vyučovaných předmětů. Její struktura vypadá následovně:

- **id**: Primární klíč, unikátní ID kategorie. (INT, PK, AUTO\_INCREMENT)
- **nazev**: Název kategorie (např. Matematika, Anglický jazyk, Biologie, Chemie). (VARCHAR(32))

### **2.2.1.4. Tabulka pu**

Tato tabulka obsahuje informace o všech knihách, které jsou nabízeny k prodeji na platformě. Její struktura vypadá následovně:

- **id**: Primární klíč, unikátní ID nabídky knihy. (INT, PK, AUTO\_INCREMENT)
- **id\_ucebnice**: ID učebnice z tabulky ucebnice, která je nabízena. (INT)
- **id\_prodejce**: ID uživatele z tabulky users, který knihu nabízí k prodeji. (INT)
- **rok\_tisku**: Rok vydání nabízené knihy. Důležité hlavně pro učebnice ZSV (INT)
- **stav**: Stav knihy, kolik z desíti (vyšší číslo = lepší stav). (INT)
- **cena**: Cena učebnice v korunách českých. (INT)
- **koupil**: ID uživatele, který učebnici koupil (0 = dosud neprodáno). (INT)
- **poznamky**: Poznámky k nabízené knize (např. stav obálky, poznámky uvnitř). (VARCHAR(256))

### **2.2.1.5. Tabulka orders**

Tato tabulka obsahuje informace o objednávkách učebnic mezi uživateli na platformě. Její struktura vypadá následovně:

- **id**: Primární klíč, unikátní ID objednávky. (INT, PK, AUTO\_INCREMENT)
- **puID**: ID nabídky knihy z tabulky pu, která je předmětem objednávky. (INT)
- **cas**: Čas vytvoření objednávky, uložený jako Unix timestamp. Bez využití. (INT)
- **complete**: Stav dokončení objednávky. 0 = nedokončeno, 1 = dokončeno. (INT)

## 2.2.2. Vztahy mezi tabulkami

**user → pu (1:N)**

- Každý uživatel (*user*) může vytvářet nabídky (*pu*).
- Sloupec *id\_prodejce* v tabulce *pu* odkazuje na *id* v tabulce *user*.

**pu → orders (1:1)**

- Objednávky (*orders*) jsou vytvořeny na základě existujících nabídek (*pu*).
- Sloupec *puID* v tabulce *orders* odkazuje na id v tabulce *pu*.

**ucebnice → pu (N:1)**

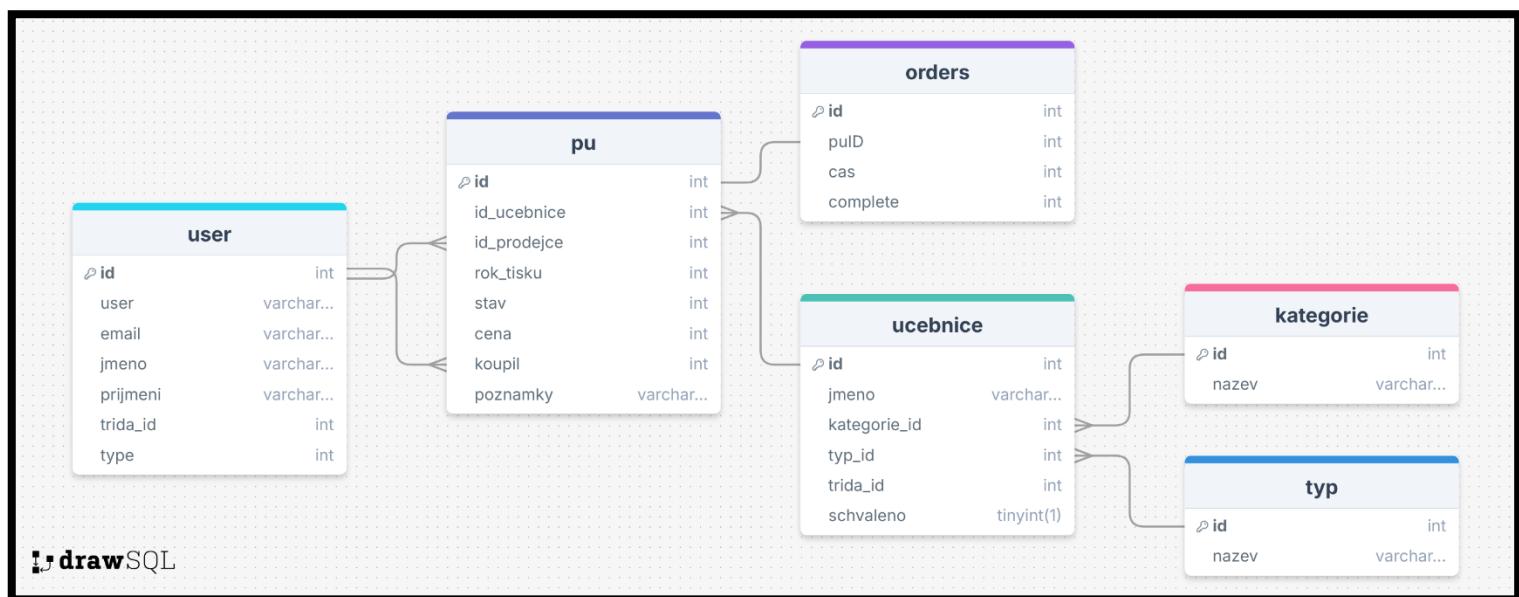
- Každá nabídka (*pu*) je vždy spojena s nějakou učebnicí (*ucebnice*).
- Sloupec *id\_ucebnice* v tabulce *pu* odkazuje na *id* v tabulce *ucebnice*.

**typ → ucebnice (N:1)**

- Každá učebnice (*ucebnice*) má přiřazený typ (*typ*).
- Sloupec *typ\_id* v tabulce *ucebnice* odkazuje na *id* v tabulce *typ*.

**kategorie → ucebnice (N:1)**

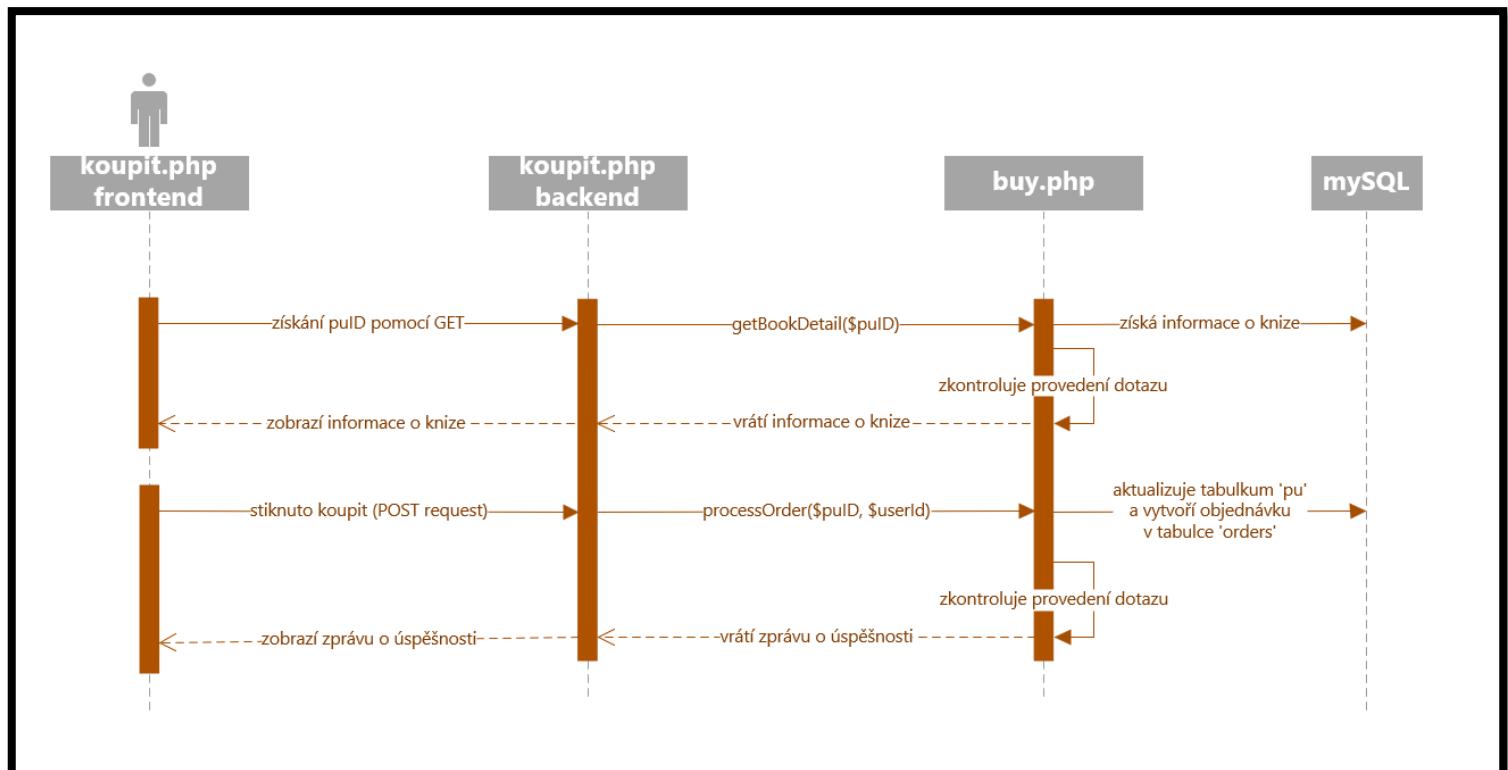
- Každá učebnice (*ucebnice*) spadá do jedné kategorie (*kategorie*).
- Sloupec *kategorie\_id* v tabulce *ucebnice* odkazuje na *id* v tabulce *kategorie*.



Obrázek 7 - Vizualizace vztahů mezi tabulkami

### 2.2.3. Ukázka interakce

Níže můžete vidět zjednodušenou vizualizaci toho, co se děje na stránce koupit.php při stisknutí tlačítka koupit a jak mezi sebou jednotlivé prvky komunikují. Vizualizace nezahrnuje načítání fotek a cesty pro případ chyb.



Obrázek 8 - Vizualizace kódu pro zakoupení knihy

K vizualizaci jsem použil **UML sekvenční diagram**. Ten slouží k znázornění, jak si objekty v systému v průběhu času posílají zprávy a jak na ně reagují. Každý objekt má svou svislou čáru (životní linii) a jeho činnost je zobrazena aktivačními bloky. Čas plyně shora dolů. Šipky mezi objekty ukazují volání metod a odpovědi. Pokud jsou aktivační bloky oddělené mezerou, znamená to, že objekt chvíli neprovádí žádnou činnost a čeká na další událost. Sekvenční diagram pomáhá lépe pochopit dynamiku systému a je důležitý při návrhu aplikací.

### 3. Klíčové části kódu

Tato kapitola se věnuje klíčovým částem kódu, bez kterých by stránka nemohla správně fungovat. Zaměřím se na vysvětlení hlavních funkcí, které zabezpečují komunikaci mezi uživatelem a databází, správu objednávek i přihlašování. Zároveň zde popíšu i ty části, které pro mě představovaly největší výzvu během programování, a uvedu, jak jsem je řešil.

#### 3.1. Google OAuth 2.0

Tento skript se stará o přihlašování uživatele pomocí Google účtu přes protokol OAuth 2.0. Níže vám popíšu, jak jednotlivé části kódu fungují.

##### 3.1.1. Úvod a OAuth údaje

```
session_start();
require_once "connect.php";
//OAUTH 2.0 hodnoty
$clientID = '...';
$clientSecret = '...';
$redirectUri = '...';
$tokenRevocationUrl = "https://oauth2.googleapis.com/revoke";
```

Kód 5 - Google OAuth 2.0 A

- `session_start()` spustí session pro správu uživatelských dat, která jsou uchovávána mezi stránkami.
- `require_once "connect.php"` zahrnuje soubor pro připojení k databázi.
- Nastavení parametrů pro připojení k OAuth 2.0 od Google: `clientID`, `clientSecret` a `redirectUri` jsou údaje získané při registraci aplikace v Google API Console.

##### 3.1.2. Přesměrování na Google autorizaci

```
if (!isset($_GET['code'])) {
    $authorizationUrl = 'https://accounts.google.com/o/oauth2/v2/auth';
    $params = array(
        'response_type' => 'code',
        'client_id' => $clientID,
        'redirect_uri' => $redirectUri,
        'scope' => 'https://www.googleapis.com/auth/userinfo.email
https://www.googleapis.com/auth/userinfo.profile',
    );
    header('Location: ' . $authorizationUrl . '?' . http_build_query($params));
    exit;
}
```

Kód 6 - Google OAuth 2.0 B

- Pokud není přítomný parametr `code` v URL (což znamená, že uživatel se ještě nepřihlásil), přesměruje to uživatele na Google stránku pro autorizaci.

- Parametry pro přesměrování obsahují: `response_type` (kód), `client_id`, `redirect_uri` (kam se Google vrátí po autorizaci), a `scope` (co vše bude aplikace moci na účtu uživatele číst, např. email a profil).

### 3.1.3. Kontrola přihlášení uživatele

```
if (!isset($_SESSION['user_id'])) {
    header("Location: ./index.php");
}
```

Kód 7 - Google OAuth 2.0 C

- Zkontroluje, zda je uživatel přihlášen. Pokud není (`$_SESSION['user_id']` neexistuje), přesměruje ho na hlavní stránku (`index.php`).
- Tento krok zajistí, že pouze přihlášení uživatelské budou pokračovat v procesu.

### 3.1.4. Příprava parametrů

#### 3.1.4.1. Před odesláním požadavku na Google

```
$accessTokenUrl = "https://oauth2.googleapis.com/token";
$params = array(
    'code' => $_GET['code'],
    'client_id' => $clientID,
    'client_secret' => $clientSecret,
    'redirect_uri' => $redirectUri,
    'grant_type' => 'authorization_code',
);
```

Kód 8 - Google OAuth 2.0 D

- `accessTokenUrl` obsahuje URL pro získání přístupového tokenu.
- `params` jsou parametry, které budou odeslány Google API. Tyto parametry obsahují:
  - `code`: Kód, který Google vrátil po úspěšném přihlášení.
  - `client_id` a `client_secret`: Údaje o aplikaci, které identifikují naši aplikaci v systému Google.
  - `redirect_uri`: URL, kam bude uživatel přesměrován po úspěšném přihlášení.
  - `grant_type`: Typ grantového typu (v tomto případě `authorization_code`), což znamená, že používáme autorizační kód pro získání přístupového tokenu.

### 3.1.4.2. Odeslání požadavku na Google API

```
$ch = curl_init();
curl_setopt($ch,CURLOPT_URL,$accessTokenUrl);
curl_setopt($ch,CURLOPT_POST,true);
curl_setopt($ch,CURLOPT_POSTFIELDS,http_build_query($params));
curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
$response = curl_exec($ch);
curl_close($ch);
$accessTokenData = json_decode($response,true);
```

Kód 9 - Google OAuth 2.0 E

- `curl_init()` inicializuje nový CURL požadavek.
- `curl_setopt()` nastaví různé možnosti pro CURL:
  - `CURLOPT_URL`: URL pro přístupový token.
  - `CURLOPT_POST`: Nastaví požadavek na POST (chceme poslat data).
  - `CURLOPT_POSTFIELDS`: Odesíláme parametry jako URL-encoded string.
  - `CURLOPT_RETURNTRANSFER`: Nastaví, že odpověď od serveru bude vrácena jako řetězec, nikoli přímo vytisknuta.
- `curl_exec($ch)` provede požadavek a uloží odpověď do proměnné `$response`.
- `curl_close($ch)` uzavře CURL požadavek.

### 3.1.4.3. Zpracování odpovědi

```
$accessTokenData = json_decode($response,true);
```

Kód 10 - Google OAuth 2.0 F

- `json_decode($response,true)` dekóduje JSON odpověď z Google API do asociativního pole, které obsahuje přístupový token a případně další údaje.

### 3.1.5. Získání dat o uživateli

#### 3.1.5.1. Ověření a použití přístupového tokenu

```
if (isset($accessTokenData['access_token'])) {  
    $_SESSION['access_token'] = $accessTokenData['access_token'];  
}
```

Kód 11 - Google OAuth 2.0 G

- `isset($accessTokenData['access_token'])`: Kontroluje, jestli byla vrácena platná hodnota přístupového tokenu z předchozího požadavku. Pokud token existuje, pokračujeme.
- `$_SESSION['access_token'] = $accessTokenData['access_token']`: Uloží přístupový token do session, aby byl dostupný během celé relace uživatele. To je důležité, protože přístupový token bude použit pro získání údajů o uživatelském účtu.

#### 3.1.5.2. Získání údajů o uživatelském účtu

```
$userInfoUrl = "https://www.googleapis.com/oauth2/v1/userinfo?access_token=" .  
    $_SESSION['access_token'];  
$ch = curl_init();  
curl_setopt($ch, CURLOPT_URL, $userInfoUrl);  
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);  
$userInfo = curl_exec($ch);  
curl_close($ch);
```

Kód 12 - Google OAuth 2.0 H

- `$userInfoUrl`: URL, které používáme k získání údajů o uživatelském účtu. Do URL přidáváme přístupový token, který bude autentifikovat naši žádost o údaje.
  - API URL pro získání informací o uživatelském účtu:  
[https://www.googleapis.com/oauth2/v1/userinfo?access\\_token=...](https://www.googleapis.com/oauth2/v1/userinfo?access_token=...)
- `curl_init()`: Inicializuje nový CURL požadavek. CURL je nástroj pro odesílání HTTP požadavků.
- `curl_setopt($ch, CURLOPT_URL, $userInfoUrl)`: Nastaví URL, na kterou bude CURL požadavek odeslán.
- `curl_setopt($ch, CURLOPT_RETURNTRANSFER, true)`: Tento příznak znamená, že odpověď nebude automaticky vytiskována, ale bude uložena jako řetězec. To nám umožňuje následně s odpovědí pracovat.
- `curl_exec($ch)`: Spustí CURL požadavek a uloží odpověď (ve formátu JSON) do proměnné `$userInfo`.
- `curl_close($ch)`: Uzavře CURL požadavek, aby se uvolnily systémové prostředky.

### 3.1.5.3. Zpracování odpovědi a uložení údajů o uživatelském účtu

```
$userInfoData = json_decode($userInfo, true);
$_SESSION['userinfodata'] = $userInfoData;
setcookie('user_info', json_encode($userInfo), time() + (86400 * 30), "/");
}
```

Kód 13 - Google OAuth 2.0 I

- `json_decode($userInfo, true)`: Dekóduje JSON odpověď (obdrženou z Google API) na asociativní pole `$userInfoData`. Tato data obsahují informace o uživatelském účtu, jako jsou jméno, email, profilová fotografie atd.
- `$_SESSION['userinfodata'] = $userInfoData;`: Uložíme získaná data o uživatelském účtu do session, což nám umožní použít je v dalších částech aplikace během celé uživatelské relace.
- `setcookie('user_info', json_encode($userInfo), time() + (86400 * 30), "/");`: Uložíme informace o uživatelském účtu také do cookies. Cookie bude trvat 30 dní, což znamená, že uživatel zůstane přihlášený i po opuštění stránky a znovaotevření prohlížeče. Data jsou uložena ve formátu JSON, což je efektivní pro uchovávání komplexních dat v cookies.

### 3.1.6. Práce s databází – kontrola a vložení uživatele

#### 3.1.6.1. Získání výsledku z databáze

```
$email = $conn->real_escape_string($userInfoData['email']);
$query = "SELECT * FROM user WHERE email='$email'";
$result = $conn->query($query);
```

Kód 14 - Google OAuth 2.0 J

- `real_escape_string()` se používá k ochraně proti SQL injection. Ujistíme se, že email uživatele je správně escapován pro bezpečné použití v SQL dotazu.
- SQL dotaz `SELECT * FROM user WHERE email='$email'` hledá v tabulce `user` uživatele s daným emailem.
- `$result = $conn->query($query)` vykoná dotaz na databázi a uloží výsledek do proměnné `$result`.

### 3.1.6.2. Pokud uživatel neexistuje

```
if ($result->num_rows === 0) {
    $username = $conn->real_escape_string
(explode('@',htmlspecialchars($userInfoData['email']))[0]);
$firstName = $conn->real_escape_string($userInfoData['given_name']);
$lastName = $conn->real_escape_string($userInfoData['family_name']);
$insertQuery = "INSERT INTO user (user, email, jmeno, prijmeni)
VALUES ('$username', '$email', '$firstName', '$lastName')";
if ($conn->query($insertQuery) === TRUE) {
    // Uživatel úspěšně vložen
} else {
    echo "Error: " . $insertQuery . "<br>" . $conn->error;
}
}
```

Kód 15 - Google OAuth 2.0 K

- `if ($result->num_rows === 0)` kontroluje, jestli uživatel s tímto emailem již existuje. Pokud neexistuje (`num_rows === 0`), pokračujeme ve vkládání nového uživatele.
- `explode('@',htmlspecialchars($userInfoData['email']))[0]` získá část emailu před zavináčem pro vytvoření uživatelského jména (převádí HTML speciální znaky na bezpečný formát).
- `real_escape_string()` opět chrání před SQL injection při vkládání uživatelských údajů.
- SQL dotaz `INSERT INTO user (user, email, jmeno, prijmeni) VALUES (...)` vloží nového uživatele do databáze s těmito údaji:
  - `user`: Uživatelské jméno (část emailu před zavináčem).
  - `email`: Email uživatele.
  - `jmeno`: Křestní jméno.
  - `prijmeni`: Příjmení.
- Pokud vložení proběhne úspěšně, pokračuje bez zprávy, pokud dojde k chybě, zobrazí se chybová hláška.

### 3.1.7. Uzavření spojení s databází

```
$conn->close();
```

Kód 16 - Google OAuth 2.0 L

Na konci se zavře spojení s databází, aby se uvolnily systémové prostředky.

## 3.2. Zpracování objednávek

Níže vám vysvětlím, jak funguje backend, který je zodpovědný za proces nákupu učebnic. Proces zahrnuje několik kroků, od načítání informací o knize, přes ověření platnosti objednávky, až po samotnou transakci a záznam objednávky v databázi.

### 3.2.1. Načítání závislostí a inicializace

```
session_start();
require_once "connect.php";
require_once "userinfo.php";
```

Kód 17 - buy.php A

- `session_start()`: Inicializuje session, která umožňuje uchovávat data o uživatelské relaci během celého používání aplikace.
- `require_once "connect.php";`: Načte soubor pro připojení k databázi, kde se pravděpodobně nachází nastavení pro připojení a proměnná `$conn`, která slouží pro přístup k databázi.
- `require_once "userinfo.php";`: Načte soubor, který pravděpodobně obsahuje funkce pro práci s informacemi o uživatelském účtu.

### 3.2.2. Funkce getBookDetail

#### 3.2.2.1. Inicializace a příprava SQL dotazu

```
function getBookDetail($conn, $puID) {
    $query = "SELECT ucebnice.id, ucebnice.jmeno AS ucebnice_nazev, kategorie.nazev
              AS kategorie_nazev, ucebnice.trida_id, typ.nazev AS typ_nazev,
              pu.rok_tisku, pu.stav, pu.cena, pu.poznamky, user.user AS
              prodejce, user.id AS prodejce_id, pu.koupil as koupil
        FROM pu
        INNER JOIN ucebnice ON pu.id_ucebnice = ucebnice.id
        INNER JOIN kategorie ON ucebnice.kategorie_id = kategorie.id
        INNER JOIN typ ON ucebnice.typ_id = typ.id
        INNER JOIN user ON pu.id_prodejce = user.id
        WHERE pu.id = ?";
```

Kód 18 - buy.php B

- Funkce `getBookDetail` získává detailní informace o knize z databáze.
- SQL dotaz: Tento dotaz vybírá několik sloupců z tabulek:
  - `ucebnice`: Informace o knize.
  - `kategorie`: Kategorii knihy.
  - `typ`: Typ knihy (např. učebnice, pracovní sešit atd.).
  - `user`: Informace o uživateli, který knihu prodává.
  - `pu`: Tabulka, která zřejmě obsahuje informace o knize v nabídce.
- Parametr `puID` slouží k určení konkrétní knihy podle jejího ID.

### 3.2.2.2. provedení SQL dotazu

```
$stmt = $conn->prepare($query);
if ($stmt) {
    $stmt->bind_param("i", $puID);
    $stmt->execute();
    $result = $stmt->get_result();
    $book = $result->fetch_assoc();
    $stmt->close();
    return $book;
} else {
    die("Chyba při přípravě dotazu: " . $conn->error);
}
}
```

Kód 19 - buy.php C

- `$stmt = $conn->prepare($query);`: Příprava SQL dotazu pro bezpečné provedení s využitím připravených příkazů.
- `$stmt->bind_param("i", $puID);`: Bind parametrů (v tomto případě i znamená integer) pro SQL dotaz, což znamená, že parametr puID je bezpečně vložen do dotazu.
- `$stmt->execute();`: Spustí SQL dotaz na databázi.
- `$result = $stmt->get_result();`: Získá výsledek dotazu.
- `$book = $result->fetch_assoc();`: Převádí výsledek na asociativní pole.
- Pokud se dotaz nezdaří, vyvolá se chyba.

### 3.2.3. Funkce getBookImages

```
function getBookImages($puID) {
    $cesta = "../foto/pu/$puID/";
    $files = glob($cesta . "*.webp"); // Vráti všechny soubory s příponou .webp
    return $files;
}
```

Kód 20 - buy.php D

- Funkce získává všechny obrázky související s knihou. Obrázky jsou později zobrazené na stránce `koupit.php`.
- Cesta k obrázkům je generována na základě `puID`, což znamená, že obrázky jsou uloženy ve složce, která odpovídá ID knihy.
- `glob($cesta . "*.webp");`: Funkce `glob` vrací seznam souborů, které odpovídají zadanému vzoru (v tomto případě soubory s příponou `.webp`).
- Funkce vrací pole souborů s obrázky, které mohou být použity pro zobrazení knihy.

### 3.2.4. Funkce processOrder

#### 3.2.4.1. Kontrola platnosti uživatele

```
function processOrder($conn, $puID, $userId) {  
    if (!isset($userId) || $userId <= 0) {  
        return "Chyba: Neplatný uživatel.";  
}
```

Kód 21 - buy.php E

- Funkce začíná kontrolou platnosti `userId`. Pokud není zadán platný uživatel, vrátí chybu.

#### 3.2.4.2. Načtení detailů knihy

```
$book = getBookDetail($conn, $puID);  
if ($book['prodejce_id'] == $userId) {  
    return "Chyba: Nemůžete koupit vlastní knihu.";  
}
```

Kód 22 - buy.php F

- Funkce získává detaily knihy pomocí funkce `getBookDetail`.
- Pokud uživatel je prodejcem knihy (tj. pokusí se koupit vlastní knihu), funkce vrátí chybu.

#### 3.2.4.3. Získání aktuálního casu a zahájení transakce

```
$cas = time();  
$conn->begin_transaction();
```

Kód 23 - buy.php G

- `time()` získá aktuální čas ve formátu UNIX timestamp. Ten udává kolik sekund uběhlo od začátku roku 1970. Výhodou oproti ostatním formátům zápisu času je menší velikost dat.
- `begin_transaction()` zahajuje transakci, což znamená, že změny v databázi budou provedeny atomicky (buď všechny změny budou úspěšné, nebo se všechno vrátí zpět, pokud něco selže).

#### 3.2.4.4. Aktualizace stavu knihy v databázi

```
$sql = "UPDATE pu SET koupil = ? WHERE id = ?";  
if ($stmt = $conn->prepare($sql)) {  
    $stmt->bind_param("ii", $userId, $puID);  
    if (!$stmt->execute()) {  
        $conn->rollback();  
        return "Chyba při aktualizaci objednávky: " . $stmt->error;  
    }  
    $stmt->close();  
} else {  
    $conn->rollback();  
    return "Chyba při přípravě prvního dotazu: " . $conn->error;
```

```
}
```

Kód 24 - buy.php H

- Tento SQL dotaz aktualizuje stav knihy v tabulce `pu` na základě ID knihy (`puID`), že byla koupena konkrétním uživatelem (`userId`).
- Pokud něco selže, transakce je vrácena zpět pomocí `$conn->rollback()`, aby se zajistila konzistence databáze.

### 3.2.4.5. Vložení objednávky do databáze

```
$sql_orders = "INSERT INTO orders (puID, cas) VALUES (?, ?)";
if ($stmt = $conn->prepare($sql_orders)) {
    $stmt->bind_param("ii", $puID, $cas);
    if (!$stmt->execute()) {
        $conn->rollback();
        return "Chyba při vkládání do orders: " . $stmt->error;
    }
    $stmt->close();
} else {
    $conn->rollback();
    return "Chyba při přípravě druhého dotazu: " . $conn->error;
}
```

Kód 25 - buy.php I

- Tento dotaz vkládá novou objednávku do tabulky `orders`. Uchovává `puID` knihy a čas, kdy byla objednávka vytvořena.

### 3.2.4.6. Spuštění transakce a potvrzení úspěchu

```
$conn->commit();
return "Objednávka byla úspěšně zpracována!";
}
```

Kód 26 - buy.php J

- Pokud všechny SQL dotazy probíhnou úspěšně, transakce je potvrzena pomocí `$conn->commit()`, což znamená, že všechny změny v databázi jsou trvalé.
- Funkce vrátí potvrzení o úspěchu.

### 3.2.5. FunkceOrderByPuID

```
function getOrderByPuID($orders, $puID) {
    foreach ($orders as $order) {
        if ($order['puID'] == $puID) {
            return $order;
        }
    }
    return null;
}
```

Kód 27 - buy.php K

- Tato funkce prohledává pole `$orders` a hledá objednávku s odpovídajícím `puID`. Pokud takovou objednávku najde, vrátí ji.
- Pokud žádná objednávka s daným `puID` neexistuje, funkce vrátí `null`.

- Potřebná pro možnost přesměrovat uživatele na objednávku po koupi.

### 3.3. Nahrávání fotek

Níže vám podrobně popíšu, jak funguje nahrávání fotek na server. Tento proces byl pro mě jedním z nejnáročnějších, protože vyžaduje bezchybné propojení několika technologií – JavaScript pro práci na straně klienta, HTML pro správné odeslání souborů, PHP pro serverové zpracování a knihovny Imagick pro optimalizaci a konverzi obrázků. Každá část musela správně komunikovat s ostatními, aby bylo možné bezpečně a efektivně nahrát fotografie ke knihám.

#### 3.3.1. HTML část

##### 3.3.1.1. HTML formulář a zobrazení existujících fotek

```
<form method="post" enctype="multipart/form-data">
```

Kód 28 - HTML pro fotky A

- `method="post"`: data (včetně binárních souborů) se odesílají v těle požadavku.
- `enctype="multipart/form-data"`: klíčové pro odeslání souborů; bez něj by `$_FILES` zůstal prázdný.

##### 3.3.1.2. Sekce náhledu existujících obrázků

```
<div id="preview" class="flex flex-wrap gap-2 mt-2 justify-center">
    <?php foreach ($existingImages as $image): ?>
        " 
    class="preview-img h-[24vh] object-cover border cursor-pointer hover:opacity-70
    rounded-md">
    <?php endforeach; ?>
</div>
```

Kód 29 - HTML pro fotky B

- `<div id="preview">`: kontejner pro již nahrané fotky.
  - `flex flex-wrap gap-2 justify-center`: Tailwind CSS pro flexibilní řádkové zobrazení, zalamování a odsazení.
  - `mt-2`: malý prostor nad kontejnerem.
- Smyčka `foreach`: pro každý prvek v poli `$existingImages` vykreslí `<img>`.
- `<img>` atributy:
  - `src="<?= $image ?>"`: absolutní či relativní URL k fotce.
  - `data-file="<?= basename($image) ?>"`: ukládá název souboru pro pozdější JS mazání.
  - `class="preview-img ..."`:
    - `h-[24vh]`: výška obrázku 24 % výšky viewportu.

- `object-cover`: zachování poměru stran a oříznutí.
- `border + rounded-md`: viditelný rámeček a zaoblené rohy.
- `cursor-pointer + hover:opacity-70`: vizuální zpětná vazba na hover.

### 3.3.1.3. Sekce pro výběr a náhled nových fotek

```
<div class="bg-gray-100 shadow-md rounded-md p-8 mx-auto">
  <div class="text-center font-bold">Přidat nové fotky</div>
  <hr>
  <input
    type="file"
    name="newFiles[]"
    id="fileInput"
    accept="image/*"
    multiple
    class="p-2 w-full"
  >
  <div id="newPreview" class="flex flex-wrap gap-2 mt-2 justify-center"></div>
</div>
</form>
```

Kód 30 - HTML pro fotky C

- Rámec kontejneru:
  - `bg-gray-100 shadow-md rounded-md p-8 mx-auto`: světle šedé pozadí, stín, zaoblené rohy, vnitřní odsazení, vycentrování.
- Nadpis `Přidat nové fotky`: tučný a centrovaný.
- `<hr>`: vodorovná čára oddělující sekce.
- `<input type="file">`:
  - `name="newFiles[]"`: pole pro více souborů.
  - `accept="image/*"`: omezení výběru jen na obrázky.
  - `multiple`: umožňuje vybrat více souborů najednou.
  - `id="fileInput"`: vazba pro JS skript.
  - `class="p-2 w-full"`: vycpávka a plná šířka.
- `<div id="newPreview">`: prázdný kontejner, kam JS přidá náhledy nově vybraných fotografií pomocí `<code>FileReader</code>`.

### 3.3.2. PHP část

#### 3.3.2.1. Funkce getExistingImages

```
function getExistingImages ($puID) {  
    $dir = "../foto/pu/$puID";  
    $existingImages = glob("$dir/*.{jpg,jpeg,png,webp,gif}", GLOB_BRACE);  
    return $existingImages;  
}
```

Kód 31 - PHP kód pro fotky A

- `getExistingImages ($puID)`: Vrátí pole cest ke všem obrázkům v adresáři konkrétní nabídky (`puID`).
- `glob(..., GLOB_BRACE)`: Podporuje vícenásobné přípony v jediném vzoru.
- Vrácené hodnoty: Pole řetězců, které PHP použije ve formuláři pro vykreslení náhledu.

#### 3.3.2.2. Smazání uživatelem vybraných obrázků

```
if (!empty($_POST['removedImages'])) {  
    $removedImages = json_decode($_POST['removedImages'], true);  
    if (is_array($removedImages)) {  
        foreach ($removedImages as $file) {  
            $filePath = "../foto/pu/$puID/" . basename($file);  
            if (file_exists($filePath)) {  
                unlink($filePath);  
            }  
        }  
    }  
}
```

Kód 32 - PHP kód pro fotky B

- Čtení vstupu: Skrytý pole `removedImages` obsahuje JSON seznam názvů odebraných souborů.
- `basename ($file)`: Zajistí, že se nepracuje s celou cestou, ale pouze s názvem souboru.
- `unlink()`: Fyzicky smaže soubor ze serveru, pokud existuje.

### 3.3.2.3. Funkce uploadNewImages

#### 3.3.2.3.1. Vytvoření složky pro nové obrázky

```
function uploadNewImages($puID, $files) {
    $dir = realpath("../foto/pu/$puID");
    if (!$dir) {
        if (!mkdir("../foto/pu/$puID", 0777, true)) {
            die("☒ Nelze vytvořit složku: ../foto/pu/$puID");
        }
        $dir = realpath("../foto/pu/$puID");
    }
    echo "☑ Složka existuje nebo byla vytvořena: $dir<br>";
```

Kód 33 - PHP kód pro fotky C

- `realpath()`: Kontrola, zda složka existuje (vrací `false`, pokud ne).
- `mkdir(..., 0777, true)`: Rekurzivně vytvoří chybějící adresáře s plnými právy.
- Chybový exit: Pokud se vytvoření nezdaří, skript skončí s chybovou hláškou.

#### 3.3.2.3.2. Určení cesty

```
foreach ($files['tmp_name'] as $key => $tmpName) {
    if (!empty($tmpName) && is_uploaded_file($tmpName)) {
        $fileName = pathinfo($files['name'][$key], PATHINFO_FILENAME);
        $targetFilePath = "$dir/$fileName.webp";
```

Kód 34 - PHP kód pro fotky D

- `foreach`: prochází všechna dočasná jména nahraných souborů (`tmp_name`).
- Kontrola: `!empty($tmpName)` zaručí, že proměnná není prázdná, a `is_uploaded_file()` ověří, že soubor skutečně přišel přes HTTP POST.
- `pathinfo(..., PATHINFO_FILENAME)`: oddělí jméno souboru bez přípony (slouží jako základ nového názvu).
- `$targetFilePath`: složí cílovou cestu a příponu `.webp`, kam se bude konvertovaný obrázek zapisovat.

### 3.3.2.3.3. Konverze a uložení fotky

```
try {
    $image = new Imagick($tmpName);
    echo "Obrázek načten: " . $files['name'][$key] . "<br>";
    // Automatická orientace podle EXIF
    if ($image->getImageOrientation() != Imagick::ORIENTATION_UNDEFINED) {
        $image->autoOrient();
    }
    $image->setImageFormat('webp');
    $image->setImageCompressionQuality(2);
    $image->writeImage($targetFilePath);
    echo "[] Obrázek uložen: $targetFilePath<br>";
    $image->clear();
    $image->destroy();
```

Kód 35 - PHP kód pro fotky E

- `new Imagick($tmpName)`: načte dočasný soubor do objektu Imagick.
- Debug výpis: potvrdí, který původní soubor se načetl.
- `autoOrient()`: opraví rotaci dle EXIF značek
- `setImageFormat('webp')` a `setImageCompressionQuality(2)`: převede obrázek do formátu WebP s vysokou kompresí (pro co nejmenší velikost).
- `writeImage()`: uloží výsledek na disk do `$targetFilePath`.
- `clear() + destroy()`: uvolní paměť a interní zdroje Imagick objektu po zpracování.

### 3.3.2.3.4. Zpracovávání chyb

```
        } catch (Exception $e) {
            echo "X Chyba při zpracování obrázku: " . $e->getMessage() .
            "<br>";
            echo "Cílová cesta: $targetFilePath<br>";
        }
    }
}
```

Kód 36 - PHP kód pro fotky F

- `catch (Exception $e)`: zachytí jakoukoli výjimku (`Exception`), která může vzniknout při práci s Imagick (např. neplatný formát souboru, chyba při zápisu na disk apod.).
- `$e->getMessage()`: vrátí popis chyby, který se vypíše, aby bylo snadné diagnostikovat problém (např. „Unable to read image“).
- Díky tomuto bloku se skript nezhavaruje při chybě, ale uživatel i vývojář získají zpětnou vazbu o příčině selhání a kontextu.

### 3.3.2.4. Vyvolání nahrávací funkce

```
if (!empty($_FILES['newFiles']['name'][0])) {
    uploadNewImages($puID, $_FILES['newFiles']);
}
```

Kód 37 - PHP kód pro fotky G

- Ověří, že byl vybrán alespoň jeden soubor.
- Volá `uploadNewImages()`, která provede veškeré kroky od validace po uložení.

### 3.3.2.5. Aktualizace seznamu pro opětovné zobrazení

```
$existingImages = getExistingImages($puID);
```

Kód 38 - PHP kód pro fotky H

- Na konci skriptu se znova načtou všechny zbývající/a nově přidané obrázky.
- Výsledek se předá zpět do HTML sekce pro vykreslení náhledu.

### 3.3.3. Javascript část

#### 3.3.3.1. Manipulace s nahráváním nových souborů a jejich zobrazením

```
document.addEventListener("DOMContentLoaded", function () {
    let removedImages = [];
    const previewDiv = document.querySelector("#preview");
    let dt = new DataTransfer();
    const fileInput = document.querySelector("#fileInput");

    fileInput.addEventListener("change", function () {
        for (let i = 0; i < fileInput.files.length; i++) {
            dt.items.add(fileInput.files[i]);
        }
        fileInput.files = dt.files;
        renderNewPreviews();
    });
});
```

Kód 39 - JS kód pro fotky A

- `document.addEventListener("DOMContentLoaded", ...)`: čeká, až se celý HTML dokument načte, aby mohl začít provádět akce (po načtení DOM).
- `removedImages`: pole pro uchování názvů odstraněných obrázků.
- `previewDiv`: proměnná pro element, kde se zobrazují existující obrázky.
- `dt`: objekt `DataTransfer`, který ukládá soubory pro nahrávání.
- `fileInput.addEventListener("change", ...)`: sleduje změnu ve vstupním poli pro soubory a přidává nové soubory do `DataTransfer`.
- `fileInput.files = dt.files`: přepíše soubory inputu těmi, které držíme v `DataTransfer`.
- `renderNewPreviews()`: po každé změně inputu se vykreslí náhledy nových souborů.

### 3.3.3.2. Vykreslení náhledů souborů

```
function renderNewPreviews() {
    const previewDiv = document.querySelector("#newPreview");
    previewDiv.innerHTML = "";

    Array.from(dt.files).forEach((file, index) => {
        const reader = new FileReader();
        reader.onload = function (e) {
            const img = document.createElement("img");
            img.src = e.target.result;
            img.classList.add("preview-img", "h-[24vh]", "object-cover",
                "border", "cursor-pointer", "hover:opacity-70");
            img.dataset.index = index;
            img.addEventListener("click", function () {
                removeNewFile(index);
            });
            previewDiv.appendChild(img);
        };
        reader.readAsDataURL(file);
    });
}
```

Kód 40 - JS kód pro fotky B

- `renderNewPreviews()`: funkce, která vykreslí náhledy souborů přidaných do `DataTransfer`.
- `previewDiv.innerHTML = ""`: vyčistí aktuální náhledy, aby mohly být vykresleny nové.
- `Array.from(dt.files).forEach(...)`: projde všechny soubory a pro každý vytvoří náhled.
- `FileReader`: umožňuje načíst soubor jako Data URL (obrázek pro zobrazení v prohlížeči).
- `img.src = e.target.result`: nastaví zdroj obrázku na načtený soubor.
- `img.addEventListener("click", ...)`: přidá událost pro odstranění obrázku po kliknutí.
- `previewDiv.appendChild(img)`: přidá obrázek do DOM pro zobrazení.

### 3.3.3.3. Odstranění souborů z náhledu

```
function removeNewFile(removeIndex) {
    let newDt = new DataTransfer();
    Array.from(dt.files).forEach((file, index) => {
        if (index != removeIndex) {
            newDt.items.add(file);
        }
    });
    dt = newDt;
    fileInput.files = dt.files;
    renderNewPreviews();
}
```

Kód 41 - JS kód pro fotky C

- `removeNewFile(removeIndex)`: funkce pro odstranění souboru z náhledu a z `DataTransfer` objektu.
- `let newDt = new DataTransfer()`: vytvoří nový objekt `DataTransfer`, do kterého přidáme pouze ty soubory, které jsme neodstranili.
- `Array.from(dt.files).forEach(...)`: projde všechny soubory a přidá je zpět, kromě toho, který chceme odstranit.
- `fileInput.files = dt.files`: přepíše soubory v inputu s novým seznamem, bez odstraněného souboru.
- `renderNewPreviews()`: znovu vykreslí náhledy po odstranění souboru.

### 3.3.3.4. Odstranění existujících obrázků

```
previewDiv.addEventListener("click", function (event) {
    if (event.target.classList.contains("preview-img")) {
        let fileName = event.target.getAttribute("data-file");
        removedImages.push(fileName);
        event.target.remove(); // Odstranění obrázku z DOM
    }
});
```

Kód 42 - JS kód pro fotky D

- `previewDiv.addEventListener("click", ...)`: naslouchá kliknutí na náhledy existujících obrázků.
- `event.target.classList.contains("preview-img")`: kontroluje, jestli byl kliknut obrázek.
- `let fileName = event.target.getAttribute("data-file")`: získá název souboru z data atributu obrázku.
- `removedImages.push(fileName)`: přidá název souboru do pole pro odstraněné obrázky.
- `event.target.remove()`: odstraní obrázek z DOM.

### 3.3.3.5. Uložení odstraněných obrázků před odesláním formuláře

```
document.querySelector("form").addEventListener("submit", function () {
    let input = document.createElement("input");
    input.type = "hidden";
    input.name = "removedImages";
    input.value = JSON.stringify(removedImages);
    this.appendChild(input);
});
```

Kód 43 - JS kód pro fotky E

- `document.querySelector("form").addEventListener("submit", ...)`: při odeslání formuláře přidá skrytý input do formuláře.
- `let input = document.createElement("input")`: vytvoří nový skrytý input element.
- `input.type = "hidden"`: nastaví input jako skrytý (uživatel ho nevidí, ale je odeslán).
- `input.name = "removedImages"`: nastaví název skrytého inputu na `removedImages`, což je název, který server použije k přijetí odstraněných obrázků.
- `input.value = JSON.stringify(removedImages)`: převede seznam odstraněných obrázků do JSON formátu a uloží ho jako hodnotu inputu.
- `this.appendChild(input)`: přidá skrytý input do formuláře, aby byl odeslán spolu s ostatními daty při submitu.

## 4. Vývoj

### 4.1. Úvod

Vývoj aplikace "Burza učebnic" přinesl řadu výzev, které bylo nutné postupně řešit. Některé problémy byly technického rázu, jiné se týkaly návrhu logiky systému nebo bezpečnosti. Největšími výzvami byly především integrace přihlašování přes Google OAuth, správná správa a bezpečnost práce s databází, efektivní nahrávání a zpracování obrázků, zajištění responzivního designu a správné oddělení jednotlivých vrstev aplikace. V této kapitole popíšu nejdůležitější překážky, se kterými jsem se během vývoje setkal, a jak jsem je překonal. Taky se zde dozvítíte, jak jsem postupoval při vývoji, které aplikace jsem používal a jak mi to pomohlo.

### 4.2. Vývoj

Vývoj této aplikace byl dost intenzivní. Pracoval jsem hlavně doma na svém počítači a používal jsem nástroje jako Visual Studio Code pro psaní kódu, XAMPP jako lokální server pro běh PHP a MySQL databáze, a GitHub pro verzování a zálohování projektu.

Ze začátku mi dalo hodně práce nastavit všechny věci tak, aby správně spolupracovaly – například propojení databáze s PHP, správné spuštění serveru přes XAMPP nebo propojení projektu s Google OAuth. Musel jsem si dohledávat spoustu informací na internetu, protože některé chyby nebyly na první pohled jasné.

Vývoj trval několik měsíců. Nešlo to vždy hladce – někdy jsem musel kód několikrát předělávat, protože jsem narazil na chyby nebo lepší způsob, jak něco udělat. Velkou pomocí byl také Tailwind CSS, díky kterému šla tvorba vzhledu rychleji.

Celkově mi tento projekt zabral hodně času, ale díky tomu jsem se naučil spoustu nových věcí, hlavně jak lépe pracovat s databází, jak zajistit bezpečnost webu, jak strukturovat větší projekt a jak využívat API třetích stran, jako je Google OAuth.

#### 4.2.1. Visual Studio Code

Pro psaní kódu jsem používal program Visual Studio Code (zkráceně VS Code). Je to lehký, ale výkonný editor kódu od společnosti Microsoft, který je zdarma. VS Code podporuje spoustu programovacích jazyků, včetně PHP, HTML, CSS i JavaScriptu, což jsem využil při vývoji této aplikace.

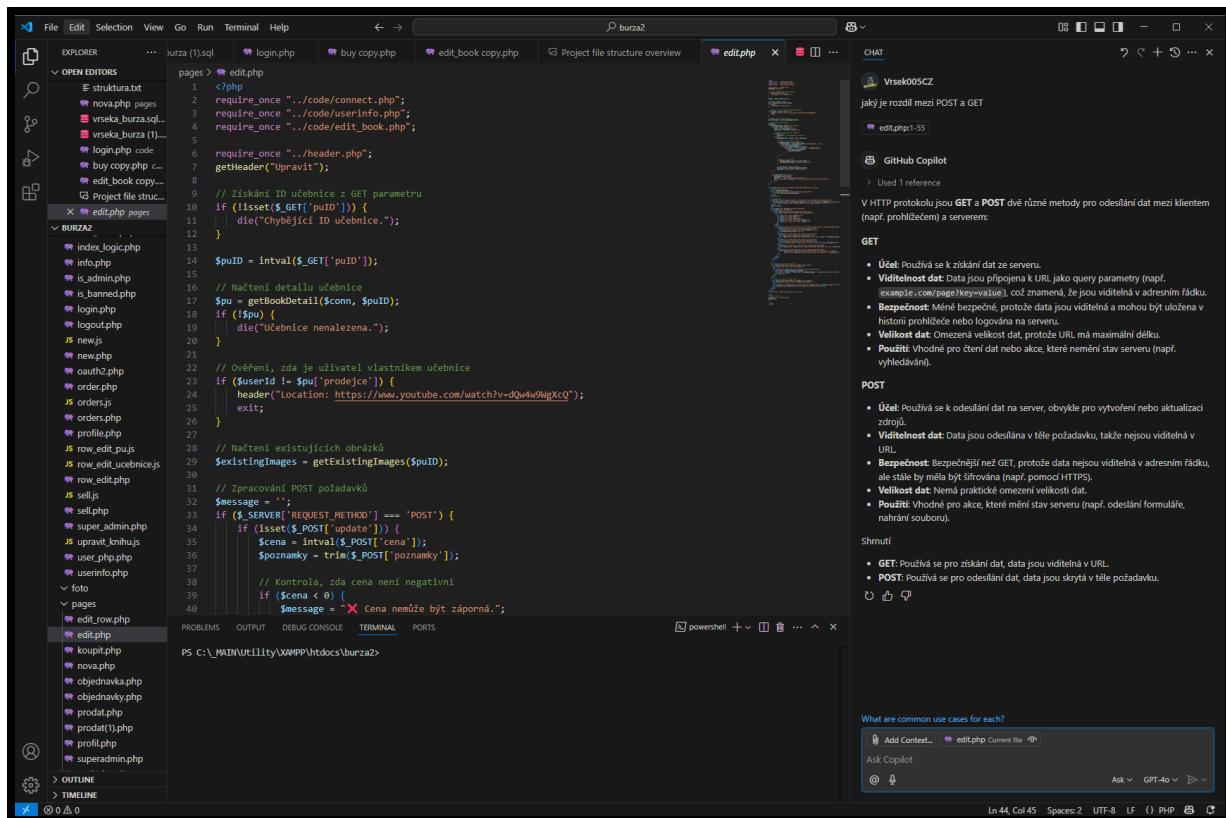
Velkou výhodou VS Code je možnost instalace rozšíření. Přidal jsem si pluginy jako např. PHP Intelephense pro lepší nápovědu při psaní PHP kódu, Format HTML in PHP

pro návodů HTML při psaní v php souboru. Díky těmto rozšířením jsem mohl rychleji programovat a snadněji hledat chyby.

Visual Studio Code má také dobrou integraci s GitHubem, takže jsem přímo z aplikace mohla nahrávat a stahovat verze mého projektu bez nutnosti manuálně soubory kamkoliv nahrávat

Ke konci vývoje jsem si aktivoval studentskou verzi GitHubu a s tím jsem i získal přístup k GitHub Copilot AI přímo ve VS, což mi extrémně ulehčilo vývoj

Celkově mi VS Code výrazně ulehčil práci a zrychlil celý vývoj aplikace.



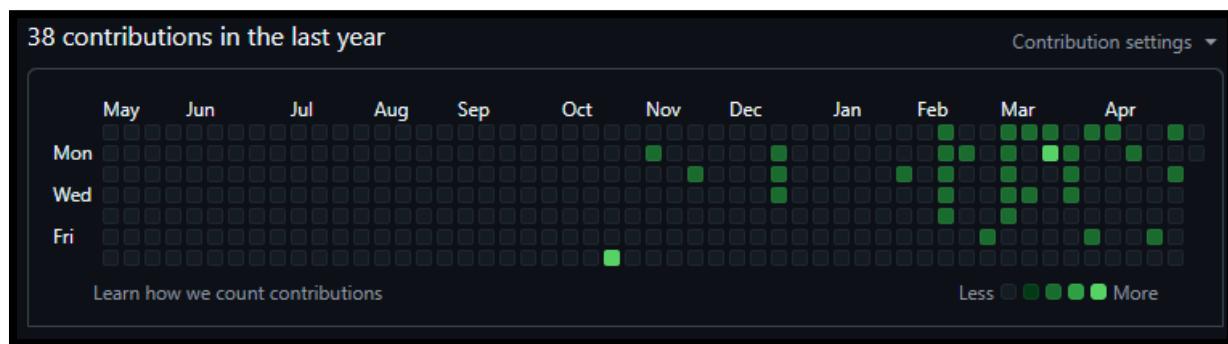
Obrázek 9 - Ukázka Visual Studio Code

## 4.2.2. GitHub

Využíval jsem integrovaný Source Control ve Visual Studio Code, což mi umožnilo jednoduše spravovat změny bez nutnosti pracovat s terminálem. Přímo z VS Code jsem mohl provádět commitování, pushování změn do repozitáře a také snadno přistupovat k historii verzí.

Díky tomu, že jsem měl vše přímo ve VS Code, mohl jsem mít neustálou zálohu a možnost vrátit se k předchozím verzím kódu, pokud by došlo k nějakým problémům nebo změnám, které bych později potřeboval upravit.

GitHub jsem také používal pro zveřejnění projektu, což mi umožnilo sdílet kód s ostatními, i když jsem na projektu pracoval sám. Tento přístup byl užitečný pro uchování verze kódu, který byl připravený na nasazení nebo sdílení s potenciálními uživateli.



Obrázek 10 - Ukázka aktivity na GitHubu

A screenshot of a GitHub repository page for "burza". The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The repository name "burza" is shown as public. The main content area shows a list of files: code (0.9), foto (0.9), pages (0.9), banned.html (0.97), burzasql (0.97), header.php (0.97), index.php (0.97), poznamky.txt (0.99), readme.txt (0.61), template.php (0.4), test.php (0.97), and user.php (0.95 MOBILE). On the right side, there are sections for About (No description, website, or topics provided), Activity (0 stars, 1 watching, 0 forks), Releases (No releases published, Create a new release), Packages (No packages published, Publish your first package), Languages (PHP 89.9%, JavaScript 9.6%, HTML 0.5%), and Suggested workflows (SLSA Generic generator, Laravel).

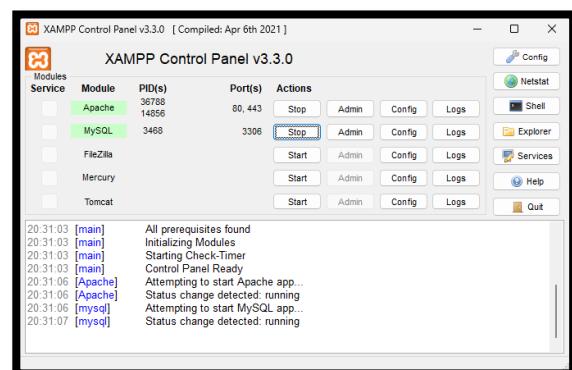
Obrázek 11 - Ukázka stránky burzy na GitHub

### 5.2.3. XAMPP

XAMPP je populární nástroj pro vývoj webových aplikací, který poskytuje kompletní vývojové prostředí s Apache serverem, MySQL databází, PHP a dalšími užitečnými nástroji. Pro mě byl XAMPP ideálním řešením pro lokální vývoj webové aplikace.

XAMPP je snadno nastavitelný a umožňuje rychle spustit webovou aplikaci na místním serveru. Díky jeho jednoduchému rozhraní jsem mohl spustit Apache server a MySQL databázi jedním kliknutím, což usnadnilo testování aplikace a interakci s databází během vývoje.

Pro správu databáze jsem využíval phpMyAdmin, který je součástí XAMPP. Tento nástroj mi poskytl grafické uživatelské rozhraní pro vytváření a správu databázových tabulek, což mi ušetřilo čas oproti práci s příkazovou řádkou. Díky phpMyAdminu jsem mohl jednoduše importovat a exportovat data, což bylo užitečné při testování a migraci databáze.



Obrázek 12 - Ukázka ovládacího panelu XAMPP

XAMPP mi také umožnil pracovat s více projekty najednou, protože jsem mohl mít různé projekty spuštěné na různých portech Apache serveru. Tato flexibilita mi pomohla při testování a správě více verzí aplikace současně.

The screenshot shows the phpMyAdmin interface for a database named 'pu'. The left sidebar shows tables like 'Nová', 'burza', 'Nová', 'kategorie', 'orders', 'pu', 'typ', 'ucebnice', 'Index', 'Sloupcy', 'information\_schema', 'mysql', 'performance\_schema', 'phpmyadmin', and 'test'. The main area displays a table with columns: id, id\_ucebnice, id\_prodejce, rok\_tisku, stav, cena, kupuj, poznamky. The table contains 92 rows of data, each with edit and delete icons. At the bottom, there are buttons for 'Základní' and 'Avancované' search, and a toolbar with 'Projekt', 'Struktura', 'SQL', 'Vyhledávání', 'Vložit', 'Import', 'Oprávění', 'Úpravy', 'Sledování', and 'Trigger'.

Obrázek 13 - Ukázka phpMyAdmin

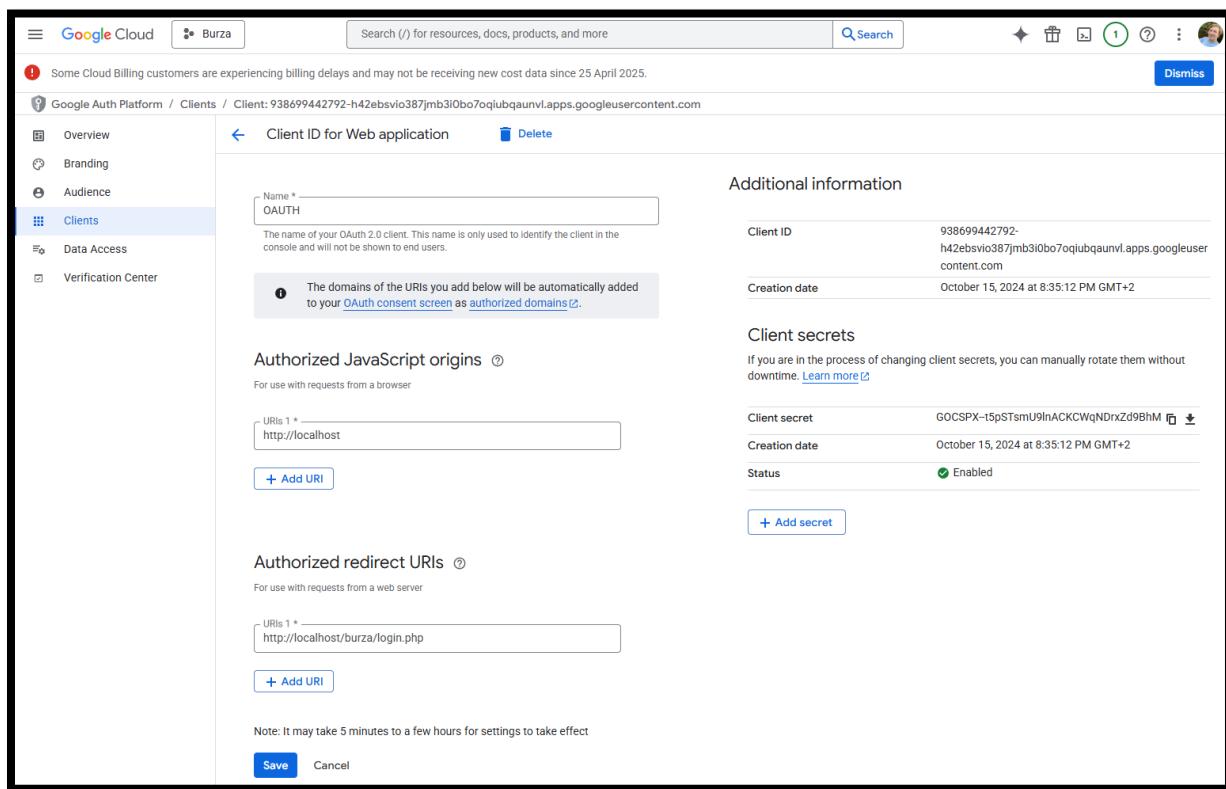
Celkově XAMPP představoval velmi efektivní a uživatelsky přívětivý nástroj pro vývoj aplikace, který mi poskytl všechny potřebné nástroje pro běh a správu aplikace v lokálním prostředí.

## 5.2.4. Google Cloud

Pro svůj projekt jsem využil služby Google Cloud Platform, především nástroj Google OAuth 2.0 pro přihlašování uživatelů. Google OAuth umožňuje bezpečné ověření identity uživatele přes jeho Google účet, aniž by bylo nutné vytvářet nový účet přímo v aplikaci. Tento způsob přihlášení je velmi bezpečný a zároveň pohodlný pro uživatele, protože využívá důvěryhodnou platformu a eliminuje nutnost zadávání hesla.

Implementace probíhala přes vytvoření projektu v Google Cloud Console, kde jsem nakonfiguroval OAuth 2.0 klienta. Nastavil jsem povolené přesměrovací adresy a údaje o aplikaci, které se zobrazují při přihlašování. Po úspěšném ověření jsem získal přístup k základním údajům o uživateli, jako je jeho jméno, e-mail a Google ID, které jsem ukládal do vlastní databáze a využíval k řízení přístupu v rámci aplikace.

Využití Google Cloud v této části vývoje mi přineslo hlavně jednoduchost při správě přihlašování, vyšší bezpečnost a profesionální způsob ověřování uživatelů, který odpovídá dnešním standardům pro moderní webové aplikace.



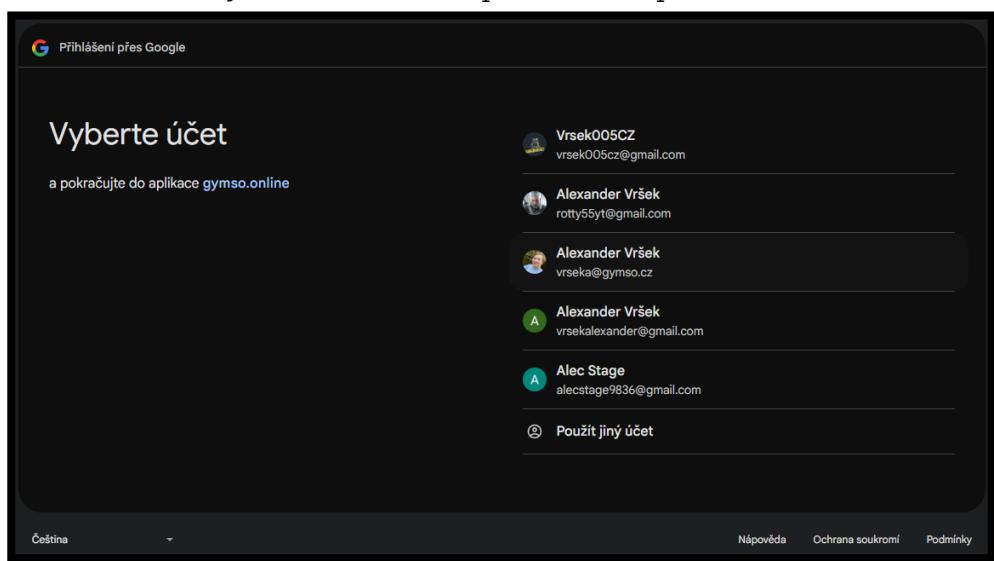
Obrázek 14 - Ukázka Google Cloudu

## 4.3. Výzvy při vývoji

### 5.2.1. Přihlášení

Jednou z klíčových částí při vývoji aplikace bylo řešení přihlašování uživatelů. Cílem bylo navrhnut systém, který bude bezpečný, jednoduchý pro uživatele a zároveň co nejméně náročný na správu.

Na začátku jsem zvažoval vlastní implementaci přihlašování, která by spočívala v registraci přes e-mail a heslo, ukládání údajů do databáze a správu zabezpečení hesel pomocí šifrování. Tento přístup by však vyžadoval dodatečné řešení zabezpečení (například hashování hesel, ochranu proti útokům typu brute force nebo resetování hesla), což by výrazně zvýšilo náročnost vývoje a zároveň i odpovědnost za správu citlivých údajů.



Obrázek 15 - Ukázka Google login

Z tohoto důvodu jsem se rozhodl využít Google OAuth 2.0, který poskytuje ověřený a bezpečný způsob přihlášení přes již existující Google účet. Tento systém přináší několik hlavních výhod:

- **Bezpečnost** – Ověření identity a správa hesel je plně v režii Googlu, čímž se výrazně snižuje riziko bezpečnostních problémů.
- **Pohodlí pro uživatele** – Uživatel se může přihlásit během několika vteřin bez nutnosti zakládat nový účet nebo si pamatovat další přihlašovací údaje.
- **Zjednodušení vývoje** – Díky využití OAuth se vývoj soustředil na integraci služby a práci s přijatými daty (např. e-mail, jméno), nikoli na budování vlastního systému ověřování a správy hesel.

Jak to celé funguje je popsáno v [kapitole 4.1](#).

Výběr Google OAuth se tak ukázal jako efektivní řešení, které umožnilo rychle nasadit přihlašovací systém na vysoké bezpečnostní úrovni a současně nabídnout uživatelům moderní a pohodlnou formu přihlášení.

## 5.2.2. Kompatibilita se smartphony

Při vývoji aplikace bylo důležité zaměřit se i na kompatibilitu s mobilními zařízeními. Smartphony mají oproti počítačům několik specifik, která je potřeba brát v úvahu – především menší rozlišení obrazovky, dotykové ovládání a jiné způsoby práce s daty a soubory.

Jedním z hlavních problémů bylo přizpůsobení rozvržení stránky, protože moje aplikace pracuje s množstvím tabulek obsahujících mnoho sloupců. Na menších displejích to způsobovalo nepřehlednost a nutnost horizontálního posouvání. Řešením bylo přizpůsobení šířky tabulek a úprava některých částí tak, aby byly klíčové informace viditelné i na malých obrazovkách bez potřeby neustálého rolování.

Další specifikum mobilních zařízení je samotné zobrazení detailů – kvůli omezenému prostoru je potřeba počítat s tím, že méně důležité informace mohou být schované nebo zobrazené až na vyžádání, aby byl design přehlednější a použitelnější.

Stav	Rok tisku	Cena	Poznámky	Koupit
1/10	1900	1 Kč	rotty55yt test 2	<button>Koupit</button>

[Zobrazit vlastní knihu:](#) ◊

Obrázek 16 - Stránka na smartphonu

V oblasti práce se soubory jsem narazil na problém specifický pro zařízení Apple – iPhone ukládají fotografie ve formátu HEIC místo běžného JPEG. Tento formát není

standardně podporován ve všech prohlížečích a serverových aplikacích. Aby bylo možné správně zpracovávat a zobrazovat obrázky od všech uživatelů, integroval jsem do aplikace nástroj ImageMagick, který umožňuje převádět HEIC soubory do běžnějších formátů, jako je webp.  
O fungování kódu pro nahrávání fotek si můžete přečíst v [kapitole 4.3](#).

Celkově bylo nutné věnovat kompatibilitě se smartphony zvláštní pozornost, aby aplikace zůstala snadno použitelná a funkční napříč různými typy zařízení.

Obrázek 17 - Stránka na počítači

## **6. Testování a nasazení**

### **4.4. Testování**

Během vývoje jsem průběžně testoval veškerou funkčnost aplikace, aby bylo možné co nejdříve odhalit chyby a nedostatky. Testování jsem prováděl především na počítači, na kterém běžel lokální server přes XAMPP. Aplikaci jsem tedy mohl pohodlně zkoušet v různých prohlížečích, kontrolovat správnost zobrazení, funkčnost formulářů, přihlašování přes Google OAuth a další klíčové části systému.

Testování na mobilních zařízeních bylo zpočátku omezené, protože aplikace nebyla umístěná na veřejně dostupném serveru. Kvůli tomu nebylo možné snadno otevřít aplikaci přímo na telefonu a ověřit mobilní chování v reálných podmínkách. Přesto jsem si byl vědom nutnosti přizpůsobení vzhledu i funkčnosti pro menší obrazovky a testoval jsem alespoň pomocí nástrojů pro vývojáře v prohlížeči (simulace zařízení).

Po zpřístupnění aplikace na veřejné adrese jsem provedl testování i na mobilních telefonech a narazil jsem na několik problémů. U zařízení iPhone jsem zjistil, že některé ikony se nezobrazují správně. Bylo tedy potřeba hledat alternativní řešení, například pomocí běžnějších ikon nebo úpravou způsobu načítání.

Dalším problémem bylo nahrávání fotografií na server, které na mobilních zařízeních trvalo podstatně déle než při nahrávání z počítače. Dalo by se to lehce opravit a veškerou kompresi fotek provádět na straně uživatele, ale vzhledem k omezené časové dotaci (za 2 a půl hodiny to mám odevzdat) to už nestíhám opravit.

Testování tak ukázalo, že různé platformy (počítač vs. mobil) mohou přinést odlišné chování aplikace, a bylo důležité těmto rozdílům přizpůsobit i samotnou implementaci.

## 4.5. Nasazení

Po dokončení vývoje aplikace na lokálním počítači přišel čas na její nasazení na veřejný server, aby byla dostupná i ostatním uživatelům. Samotné nasazení ale přineslo řadu technických problémů, které bylo nutné postupně vyřešit.

Prvním krokem bylo překonfigurování Google OAuth 2.0. Při vývoji jsem používal přesměrování na lokální adresu (např. localhost), ale po přesunu na ostrý server bylo nutné změnit nastavení přesměrovacích URL adres v Google Cloud Console na novou veřejnou doménu. Bez této úpravy by autentizace přes Google nefungovala, protože OAuth přísně ověřuje správnost přesměrování.

Dalším problémem byly cesty v aplikaci. Mnoho odkazů a odkazování na soubory bylo původně nastaveno jako absolutní nebo přímo přizpůsobené lokálnímu prostředí. Po přesunu na server bylo potřeba cesty upravit, aby fungovaly správně v novém adresářovém uspořádání a odpovídaly struktuře serveru.

Při samotném nasazení se objevilo i několik problémů s právy na soubory a složky. Například upload obrázků selhal kvůli chybějícím právům pro zápis do určitých adresářů. Bylo tedy nutné nastavit správná oprávnění (např. 755 nebo 775 na složky) a ověřit, že server umožňuje nahrávání souborů.

Kromě toho se objevily i drobné rozdíly v chování mezi lokálním a ostrým serverem, například odlišnosti v konfiguraci PHP nebo v přístupových cestách k souborům. I tyto problémy jsem postupně odhalil a upravil tak, aby aplikace fungovala správně.

Celý proces nasazení byl náročnější, než jsem původně očekával, ale všechny komplikace se mi podařilo vyřešit a aplikace byla úspěšně uvedena do provozu.

## 7. Závěr

### 7.1. Shrnutí

Práce na tomto projektu pro mě představovala velký krok kupředu v oblasti webového vývoje. Před jeho zahájením jsem měl za sebou pouze zkušenosť s tvorbou jednoduché chatovací aplikace v PHP a SQL. V porovnání s tím byl tento projekt výrazně složitější – jak po stránce objemu kódu, tak po stránce správy databáze, uživatelského přihlášení a nasazení aplikace do produkčního prostředí.

I přes tu náročnost se mi podařilo projekt úspěšně dokončit. Během testování jsem ověřil všechny klíčové funkce a postupně opravil problémy, které se objevily, například kompatibilitu s mobilními zařízeními, správné zpracování souborů nebo nasazení Google OAuth na veřejný server. Aplikace na konci testování fungovala tak, jak jsem si na začátku stanovil.

Jedním z největších přínosů tohoto projektu pro mě bylo pochopení, jak velké projekty skutečně vznikají – jak je důležité správně si rozvrhnout strukturu aplikace, dobře navrhnout databázi, psát udržitelný kód a postupně řešit chyby, které se objeví až během provozu. Uvědomil jsem si také, že velká část práce vývojáře není jen v samotném psaní kódu, ale i v testování, opravách, přizpůsobení prostředí a hledání nových řešení na vzniklé problémy.

Vývoj této aplikace mi zabral přibližně 50 až 120 hodin čistého času. Tento čas zahrnoval nejen samotné programování, ale i studium dokumentací, hledání chyb, ladění aplikace a její nasazení. Přestože byl projekt časově náročnější, než jsem původně očekával, přinesl mi spoustu nových dovedností, a především sebevědomí v práci na rozsáhlějších projektech.

Celkově hodnotím tuto zkušenosť velmi pozitivně. Projekt mi umožnil si v praxi vyzkoušet mnoho technologií, které jsem znal jen teoreticky, a ukázal mi, jak vypadá skutečný vývoj funkční a veřejně dostupné webové aplikace.

## 7.2. AI

Během vývoje projektu jsem intenzivně využíval umělou inteligenci, která mi výrazně usnadnila práci a urychlila celý proces. Bez podpory AI by vývoj trval minimálně dvakrát déle a některé části kódu by bylo složitější a časově náročnější správně navrhnut či optimalizovat.

AI nástroje jsem využíval hlavně při tvorbě frontendu, návrhu struktury HTML, CSS a při optimalizaci kódu. Umělá inteligence mi pomáhala hledat efektivnější řešení, rychleji opravovat chyby, zlepšovat čitelnost a udržovatelnost kódu a navrhovat alternativní postupy tam, kde jsem si nebyl jistý.

Významnou roli sehrála AI také při tvorbě základních kostr – například u funkcí, API volání, nebo návrhu databázových struktur. AI dokázala rychle vytvořit základní návrh, na kterém jsem mohl dál stavět a přizpůsobit ho konkrétním potřebám projektu. Díky tomu jsem se mohl soustředit spíše na složitější části vývoje a logiku aplikace.

Přestože mi AI ušetřila obrovské množství času a práce, ukázalo se, že v některých situacích není její použití ideální. Byly případy, kdy bylo lepší spolehnout se na vlastní úsudek, ručně upravit návrhy, které AI vytvořila, nebo si některé věci promyslet sám, protože lidský přístup byl přesnější, praktičtější a více odpovídal konkrétním požadavkům projektu.

V jednom případě jsem při řešení problému, kdy metoda GET posílala špatné parametry, využil AI k hledání chyby. AI však problém nenašla. Skutečná příčina byla skrytá v tom, že se požadavek GET odesílal dvakrát za sebou, což vedlo ke zmateným výsledkům. Tento problém jsem musel nakonec odhalit a vyřešit vlastní analýzou a testováním.

Celkově se ukázalo, že AI je vynikajícím pomocníkem pro rutinní a opakující se úkoly, tvorbu základních návrhů a optimalizaci, ale v komplikovanějších nebo specifických případech je stále nezastupitelný lidský přístup a vlastní logické myšlení.

Bez využití AI by celý projekt trval podstatně déle a byl by mnohem náročnější. Dnes si už bez kombinace vlastní práce a chytré podpory AI podobný vývoj prakticky neumím představit.

### 7.3. Možnosti rozšíření

Ačkoli je aplikace plně funkční a pokrývá všechny hlavní požadavky, existuje mnoho oblastí, kde by bylo možné systém dále rozvíjet, zlepšovat a přizpůsobovat rostoucím potřebám uživatelů.

Jedním z hlavních směrů by bylo **vylepšení administračního prostředí**. V současné době je správa uživatelů, knih a objednávek řešená poměrně jednoduše. V budoucnu by bylo možné vytvořit plnohodnotný administrační dashboard, který by obsahoval například:

- Statistiky o aktivitě uživatelů a prodeji knih,
- Možnost hromadných akcí (mazání, úpravy),
- Správu reportů a řešení nahlášených problémů.

Další důležitou oblastí rozšíření by bylo **zavedení interního chatu** mezi uživateli. Přímá komunikace mezi kupujícím a prodávajícím v rámci platformy by výrazně zjednodušila domlouvání detailů, odstranila potřebu externí komunikace (například přes e-mail) a zároveň by umožnila lepší kontrolu nad procesem vyřizování objednávek. Chat by mohl být řešen pomocí systému jednoduchých zpráv s notifikacemi.

Velký potenciál se nabízí také v oblasti **optimalizace UI (uživatelského rozhraní)** a **UX (uživatelské zkušenosti)**. Především:

- Lepší responzivita a přizpůsobení vzhledu pro mobilní zařízení,
- Jednodušší, přehlednější navigace,
- Atraktivnější a modernější design, například použití tmavého režimu,
- Lepší vizuální zpětná vazba při akcích uživatele (potvrzení, chyby apod.).

**Moje vlastní návrhy na rozšíření** zahrnují i další vylepšení:

- **Systém hodnocení a recenzí uživatelů:** umožnit kupujícím i prodávajícím hodnotit protistranu a přispět tak k větší důvěře v platformu.
- **Implementace platebních bran:** například napojení na služby jako Stripe nebo PayPal, aby bylo možné provádět platby přímo přes web.
- **Notifikační systém:** e-mailové nebo push notifikace při změnách stavu objednávky, nových zprávách apod.
- **Zabezpečení a auditní logy:** sledování důležitých událostí v systému (přihlášení, změny v databázi) pro lepší bezpečnost a dohledatelnost.

Celkově je aplikace navržená tak, aby byla rozšířitelná a další funkce bylo možné přidávat postupně, bez nutnosti zásadních přepisů stávajícího systému.

## **8. Přílohy**

### **8.1. Obrázky**

Obrázek 1 - Stránka s CSS.....	7
Obrázek 2 - Stránka bez CSS.....	7
Obrázek 3 - Ukázka dokumentace Tailwindu .....	8
Obrázek 4 - Hlavička burzy učebnic.....	9
Obrázek 5 - Ukázka responzivního designu .....	10
Obrázek 6 - Schéma spolupráce PHP a mySQL .....	11
Obrázek 7 - Vizualizace vztahů mezi tabulkami .....	18
Obrázek 8 - Vizualizace kódu pro zakoupení knihy.....	19
Obrázek 9 - Ukázka Visual Studio Code .....	41
Obrázek 10 - Ukázka aktivity na GitHubu.....	42
Obrázek 11 - Ukázka stránky burzy na GitHub .....	42
Obrázek 12 - Ukázka ovládacího panelu XAMPP .....	43
Obrázek 13 - Ukázka phpMyAdmin .....	43
Obrázek 14 - Ukázka Google Cloudu .....	44
Obrázek 15 - Ukázka Google login .....	45
Obrázek 16 - Stránka na smartphonu.....	46
Obrázek 17 - Stránka na počítači.....	46

### **8.2. Tabulky**

Tabulka 1 - Porovnání kódu Tailwindu a CSS.....	9
Tabulka 2 - Grafická reprezentace struktury souborů.....	15

### **8.3. Kódy**

Kód 1 - Ukázka kódu pro hlavičku pro Tailwind .....	9
Kód 2 - Ukázka kódu pro hlavičku pro CSS.....	9
Kód 3 - Ukázka SQL dozazu.....	12
Kód 4 - Ukázka funkce pro vložení nové knihy .....	13
Kód 5 - Google OAuth 2.0 A .....	20
Kód 6 - Google OAuth 2.0 B.....	20
Kód 7 - Google OAuth 2.0 C .....	21
Kód 8 - Google OAuth 2.0 D .....	21
Kód 9 - Google OAuth 2.0 E.....	22
Kód 10 - Google OAuth 2.0 F.....	22
Kód 11 - Google OAuth 2.0 G .....	23
Kód 12 - Google OAuth 2.0 H .....	23
Kód 13 - Google OAuth 2.0 I.....	24
Kód 14 - Google OAuth 2.0 J.....	24

Kód 15 - Google OAuth 2.0 K.....	25
Kód 16 - Google OAuth 2.0 L.....	25
Kód 17 - buy.php A .....	26
Kód 18 - buy.php B .....	26
Kód 19 - buy.php C.....	27
Kód 20 - buy.php D .....	27
Kód 21 - buy.php E .....	28
Kód 22 - buy.php F.....	28
Kód 23 - buy.php G.....	28
Kód 24 - buy.php H.....	29
Kód 25 - buy.php I.....	29
Kód 26 - buy.php J.....	29
Kód 27 - buy.php K .....	29
Kód 28 - HTML pro fotky A.....	30
Kód 29 - HTML pro fotky B.....	30
Kód 30 - HTML pro fotky C .....	31
Kód 31 - PHP kód pro fotky A .....	32
Kód 32 - PHP kód pro fotky B.....	32
Kód 33 - PHP kód pro fotky C .....	33
Kód 34 - PHP kód pro fotky D .....	33
Kód 35 - PHP kód pro fotky E .....	34
Kód 36 - PHP kód pro fotky F .....	35
Kód 37 - PHP kód pro fotky G .....	35
Kód 38 - PHP kód pro fotky H .....	35
Kód 39 - JS kód pro fotky A .....	36
Kód 40 - JS kód pro fotky B .....	37
Kód 41 - JS kód pro fotky C .....	38
Kód 42 - JS kód pro fotky D .....	38
Kód 43 - JS kód pro fotky E .....	39

## Zdroje:

Obrázek 3: TAILWIND CSS. *Tailwind CSS*. Dostupné z: <https://tailwindcss.com/> [cit. 2025-04-28]

Obrázek 6: JAKPSATWEB.CZ. *Možnosti PHP*. Dostupné z: <https://www.jakpsatweb.cz/php/moznosti-php.html> [cit. 2025-04-28]

Obrázek 7: DRAWSQL. *DrawSQL*. Dostupné z: <https://drawsql.app/> [cit. 2025-04-28]

Obrázek 8: MICROSOFT. *Microsoft Visio: Flowchart software*. Dostupné z: <https://www.microsoft.com/cs-cz/microsoft-365/visio/flowchart-software> [cit. 2025-04-28]

Obrázek 9: MICROSOFT. *Visual Studio Code*. Dostupné z: <https://code.visualstudio.com/> [cit. 2025-04-28]

Obrázek 10, 11: GITHUB. *GitHub*. Dostupné z: <https://github.com/> [cit. 2025-04-28]

Obrázek 12: APACHE FRIENDS. *XAMPP: Apache + MariaDB + PHP + Perl*. Dostupné z: <https://www.apachefriends.org/> [cit. 2025-04-28]

Obrázek 13: PHPMYADMIN. *phpMyAdmin*. Dostupné z: <https://www.phpmyadmin.net/> [cit. 2025-04-28]

Obrázek 14, 15: GOOGLE CLOUD. *Google Cloud Console*. Dostupné z: <https://console.cloud.google.com/> [cit. 2025-04-28]

Kód 1: TAILWIND CSS. *Tailwind CSS*. Dostupné z: <https://tailwindcss.com/> [cit. 2025-04-28]

Kód 5-16: YOUTUBE. *PHP 7 cURL Google OAuth2 Login & Logout Script to Store Profile & Revoke Access Token in Sessions* [video]. Dostupné z: <https://www.youtube.com/watch?v=T-wmTu7-yu8> [cit. 2025-04-28]

Kód 29, 30: TAILWIND CSS. *Tailwind CSS*. Dostupné z: <https://tailwindcss.com/> [cit. 2025-04-28]

YOUTUBE. *UML Sequence Diagram Tutorial / Easy to Understand with Examples* [video]. Dostupné z: <https://www.youtube.com/watch?v=gzKe7yt8qEo> [cit. 2025-04-28]

W3SCHOOLS. *W3Schools*. Dostupné z: <https://www.w3schools.com/> [cit. 2025-04-28]

STACK OVERFLOW. *Stack Overflow*. Dostupné z: <https://stackoverflow.com/> [cit. 2025-04-28]

QUORA. *Quora*. Dostupné z: <https://www.quora.com/> [cit. 2025-04-28]

Neocitované fotky, tabulky, diagramy a kódy jsou mé tvorby.