|  |  |  |
| --- | --- | --- |
| **МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА** | | |
| **ПРОФЕСИОНАЛНА ГИМНАЗИЯ ПО ЕЛЕКТРОТЕХНИКА И ЕЛЕКТРОНИКА**  бул. Пещерско шосе № 26  4002 гр. Пловдив, България  тел. 032 / 643-657  info-1690174@edu.mon.bg |  | **VOCATIONAL SCHOOL OF ELECTRICAL ENGINEERING AND ELECTRONICS**  26 Peshtersko chaussee blvd.,  4002 Plovdiv, Bulgaria  Phone: 032 / 643-657  info-1690174@edu.mon.bg |

**професия код 481030 „Приложен програмист“**

**специалност код 4810301 „Приложно програмиране“**

**КУРСОВ ПРОЕКТ**

**за придобиване трета степен професионална квалификация**

**Тема: spotify data tracker**

**Разработили**: Кристиян Ценов

Боян Копринков

Георги Генчев

Виктор Сотиров

Клас: 12.А

e-mail: viktor

bobo

g.genchev39@gmail.com

kris

Пловдив

2025 година

# Съдържание

[Увод 1](#_Toc184662743)

[Глава 1 – Използвани технологии 2](#_Toc184662744)

[1.1. ASP.NET MVC 2](#_Toc184662745)

[1.2. Spotify Web API 3](#_Toc184662746)

[1.3. OAuth 2.0 (логване чрез Spotify) 3](#_Toc184662747)

[Глава 2 – Основни функционалности 5](#_Toc184662749)

[2.1. Вход с профил в Spotify 5](#_Toc184662750)

[2.2. Топ 5 артисти 6](#_Toc184662751)

[2.3. Топ 5 песни 6](#_Toc184662752)

[2.4. Водещи жанрове 7](#_Toc184662753)

[2.5. Последно слушани песни 7](#_Toc184662753)

[2.6. Препоръчани песни и артисти 7](#_Toc184662753)

[Глава 3 – Структура на приложението 9](#_Toc184662754)

[3.1. Разпределение на модели, контролери и изгледи 9](#_Toc184662755)

[3.2. Служба за връзка със Spotify API (SpotifyService) 10](#_Toc184662756)

[Глава 4 -Работа с Spotify API 14](#_Toc184662759)

[4.1. Получаване на достъп (access token) 14](#_Toc184662760)

[4.2. Използвани основни endpoints 14](#_Toc184662761)

[Глава 5 -Дизайн и потребителски интерфейс 14](#_Toc184662759)

[5.1. Описание на изгледите и навигацията 14](#_Toc184662761)

[Заключение 21](#_Toc184662765)

[Използвани Източници 22](#_Toc184662766)

[Приложение 1 23](#_Toc184662767)

[Приложение 2 24](#_Toc184662768)

[Приложение 3 25](#_Toc184662769)

# Увод

Проектът Spotify Data Tracker представлява уеб приложение, разработено с помощта на ASP.NET MVC, което позволява на потребителите да се свържат със своя Spotify профил и да визуализират персонализирани статистики, базирани на тяхната музикална активност.

Основната цел на приложението е да предостави по-интуитивен и изчистен начин за разглеждане на слушателските навици на потребителя, като използва официалния Spotify Web API. Приложението дава възможност за преглед на най-слушаните артисти и песни, водещи музикални жанрове, последно слушани парчета, както и препоръки за нови песни и изпълнители.

Чрез интеграция с OAuth 2.0 логин потребителите се удостоверяват директно през Spotify, което гарантира сигурен достъп до личните им данни. Проектът демонстрира реална работа с външен API и обработка на данни в контекста на модерно уеб приложение.

# Глава 1 – Използвани технологии

В тази глава се разглеждат основните теоретични аспекти и концепции, които лежат в основата на разработката на уеб-базирани приложения, както и използваните технологии за реализацията на настоящия проект. Целта е да се даде ясно разбиране за тяхното предназначение, начин на работа и как те се интегрират в единна система.

## 1.1. ASP.NET MVC

ASP.NET MVC е уеб framework, който следва модела Model-View-Controller. Той позволява логическо разделение на кода, по-лесна поддръжка и добра организация на проекта. Използван е за изграждане на основната структура на уеб приложението – обработка на заявки, рендиране на изгледи и управление на бизнес логиката.

## 1.2. Spotify Web API

## Spotify Web API предоставя достъп до различни данни от потребителски акаунти в Spotify, включително най-слушани артисти и песни, последни слушания и препоръки. API-то е използвано за извличане на статистическа информация и изграждане на персонализирано потребителско изживяване.

## 1.3. OAuth 2.0 (логване чрез Spotify)

OAuth 2.0 е протокол за удостоверяване, който позволява сигурен достъп до потребителски данни без споделяне на пароли. В този проект се използва за логване чрез Spotify – потребителите влизат с профила си и предоставят достъп до информацията, необходима за функционирането на приложението.

# Глава 2 – Основни функционалности

Spotify Data Tracker предлага поредица от персонализирани функционалности, които дават на потребителя достъп до детайлна информация за неговите слушателски навици в Spotify. След влизане в приложението чрез своя Spotify акаунт, потребителят може да използва следните функции.

## 2.1. Вход чрез Spotify акаунт

## Потребителят се удостоверява чрез OAuth 2.0 логин, като дава разрешение на приложението да използва неговите Spotify данни. След успешен вход се получава access token, който се използва за всички последващи заявки към Spotify API.

## 2.2. Топ 5 най-слушани артисти

Извлича се списък с петте най-слушани изпълнители на потребителя за избран период (напр. последните 4 седмици или последната година), както и допълнителна информация като жанр и популярност.

## 2.3. Топ 5 най-слушани песни

## Показва се списък с най-популярните песни, които потребителят е слушал, заедно с обложка на албума, име на изпълнителя и продължителност на песента.

## 2.4. Последно слушани песни

Чрез endpoint-а Get Recently Played Tracks се показва списък с последните парчета, които потребителят е слушал в Spotify.

## 2.5. Водещи жанрове

На базата на данни от най-слушаните артисти се извеждат жанровете, които потребителят слуша най-често, като се използва класификацията, предоставена от Spotify.

## 2.6. Препоръки за песни и артисти

Приложението използва Spotify recommendation API, за да предостави персонализирани предложения за нови изпълнители и песни, съобразени със слушателските навици на потребителя.

# Глава 3 – Структура на приложението

## Приложението Spotify Data Tracker е разработено по модела MVC (Model-View-Controller), който разделя логиката на три основни слоя за по-добра организация, поддръжка и разширяемост на кода:

## 3.1. Models (Модели)

Моделите представляват обектите, които носят данни между различните слоеве на приложението. В проекта са дефинирани модели като:

* ArtistModel – съдържа информация за артист (име, жанрове, изображение и популярност)
* TrackModel – информация за песен (име, изпълнител, обложка, дължина)
* GenreModel – класификация по жанрове
* RecommendationModel – структура за препоръчани артисти и песни

## 3.2. Controllers (Контролери)

Контролерите приемат потребителски заявки, обработват ги и връщат подходящ изглед. Основният контролер в приложението е:

* SpotifyController – отговаря за всички действия, свързани със Spotify API: вход, зареждане на данни, показване на статистики и препоръки.
* HomeController – Начална страница
* AuthController – използва се за валидация на потребителския профил

## 3.3. Views (Изгледи)

Модулът за управление на заявки осигурява възможност за администриране на подадените заявки – преглед, одобрение, отхвърляне и актуализация на статуса.

* Login.cshtml – начална страница за вход
* Dashboard.cshtml – основна страница със статистики
* Recommendations.cshtml – страница с персонализирани предложения
* TopArtists.cshtml – страница с петте най-слушани артисти
* Частични изгледи (например \_TopTracks.cshtml) за по-добро разделение на интерфейса

## 3.4. Services (Служби)

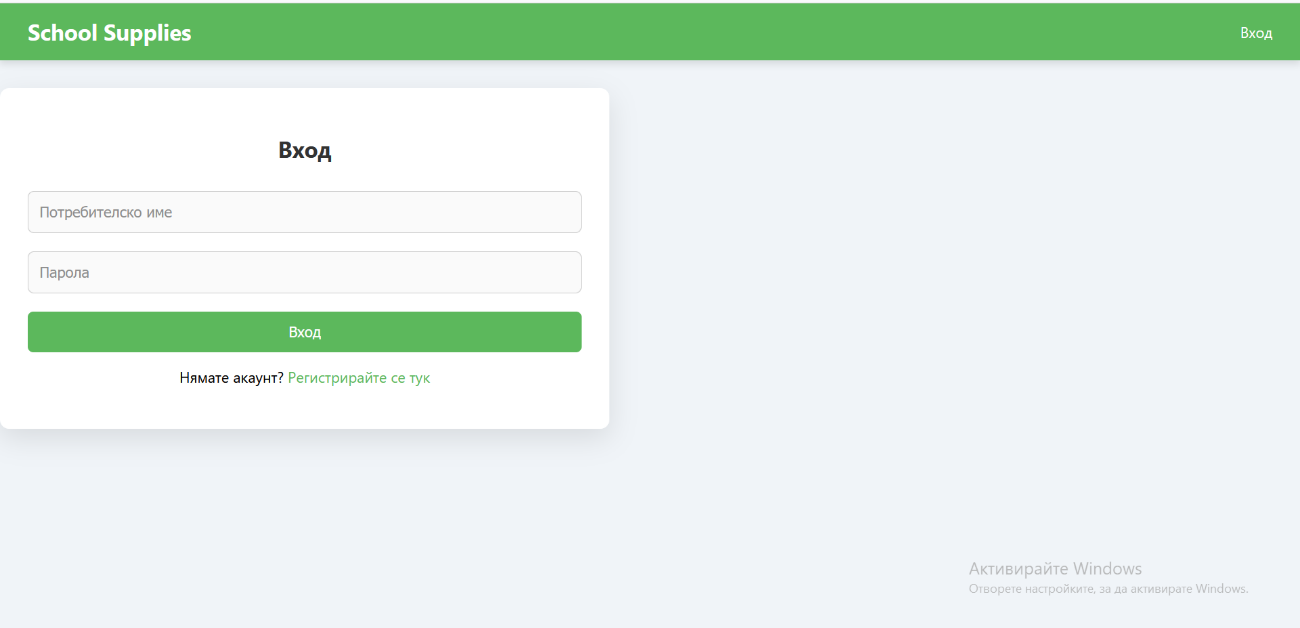
Създаден е специален клас SpotifyService, който поема комуникацията със Spotify API. Той:

* **Основни таблици и връзки:**
* **users:** Съхранява информация за потребителите (потребителско име, парола, роля).
* **requests:** Запазва заявките, направени от потребителите. Свързана е с users чрез user\_id.
* **request\_items:** Записва консумативите, включени в дадена заявка. Свързана е с requests чрез request\_id.
* **Операции:**
* Добавяне на нови данни (потребители, заявки, консумативи).
* Актуализация (напр. промяна на статус на заявка).
* Извличане на данни (списък с потребители, заявки и консумативи).
* Изтриване на записи, ако е необходимо.

Този модул е основен за правилното функциониране на приложението, тъй като осигурява съхранението и достъпа до цялата информация.

# Глава 4 - ръководство за потребителя

## 4.1. Вход в системата

Първата стъпка при използването на системата за управление на училищни заявки е влизането в акаунт. Потребителите трябва да въведат своите данни за достъп „Фигура 3“.

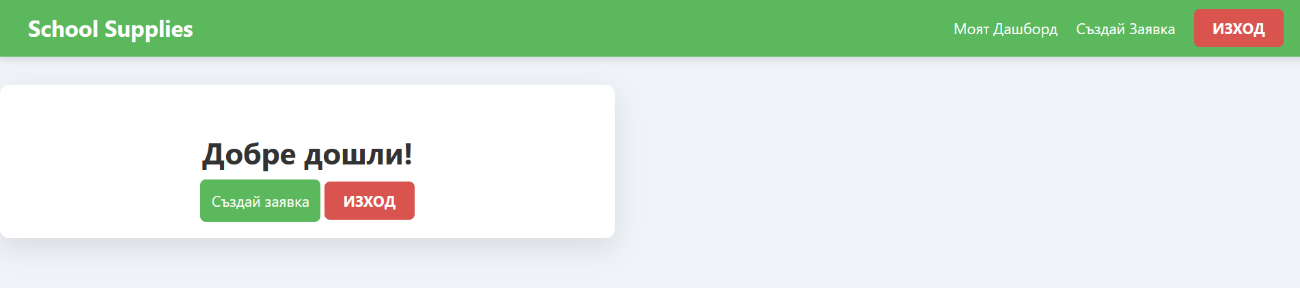
Фигура 3: Форма за вход

**Интерфейс за вход от „Фигура 3“:**

* **Поле за потребителско име:** Тук се въвежда потребителското име.
* **Поле за парола:** Тук се въвежда паролата.
* **Бутон "Вход":** След въвеждане на данните натиснете този бутон за достъп до системата.
* **Линк за регистрация:** Ако нямате акаунт, можете да се регистрирате, като кликнете върху „Регистрирайте се тук“ и попълните необходимите полета.

## 4.2. Успешно влизане в системата

След успешно влизане в системата, потребителят се приветства с основното меню за управление на училищни заявки. Тук той може да създава и управлява заявки.



Фигура 4: Успешен вход в системата

**Интерфейс за успешно влизане:**

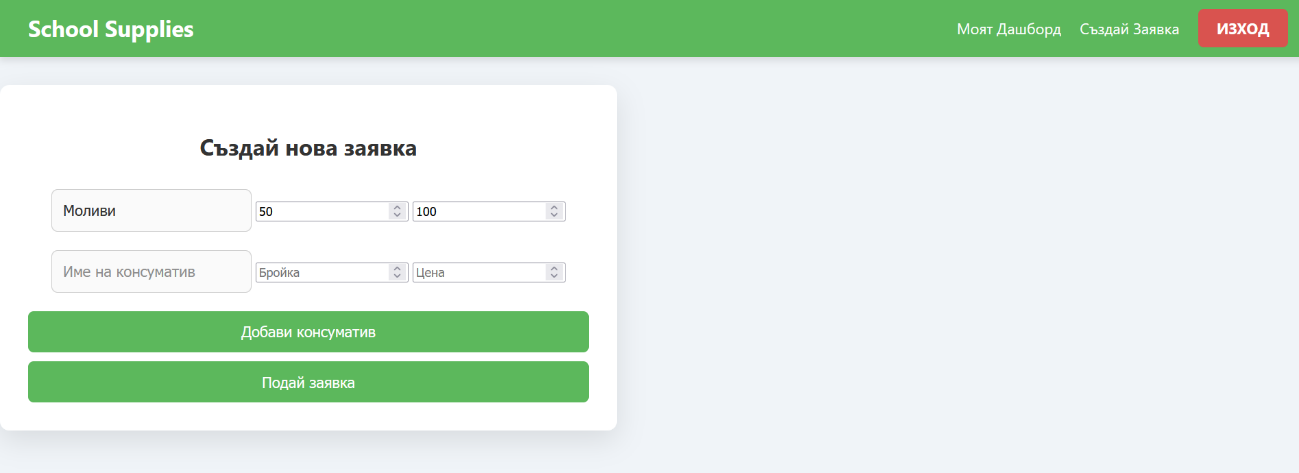
На „Фигура 4“ се вижда основната страница за навигация, която включва:

* **Добре дошли съобщение:** Поздравява потребителя след успешен вход.
* **Бутон „Създай заявка“:** Позволява на потребителя да създаде нова заявка за училищни консумативи.
* **Бутон „Изход“:** Прекратява текущата сесия и връща потребителя към страницата за вход.
* **Горна навигационна лента:** Съдържа допълнителни връзки като „Моят Дашборд“ и „Създай Заявка“ за бърз достъп до основните функции на системата.

Тази страница предоставя лесен достъп до всички основни действия, които потребителят може да извърши в системата.

## 4.3. Създаване на нова заявка за консумативи

Този код предоставя възможност на потребителите да създадат нова заявка за училищни консумативи. Основната функционалност включва добавяне на множество артикули, като всеки артикул има наименование, бройка и цена „Фигура 5“.

**Интерфейс за създаване на заявка:**

Фигура 5: Форма за създаване на заявка

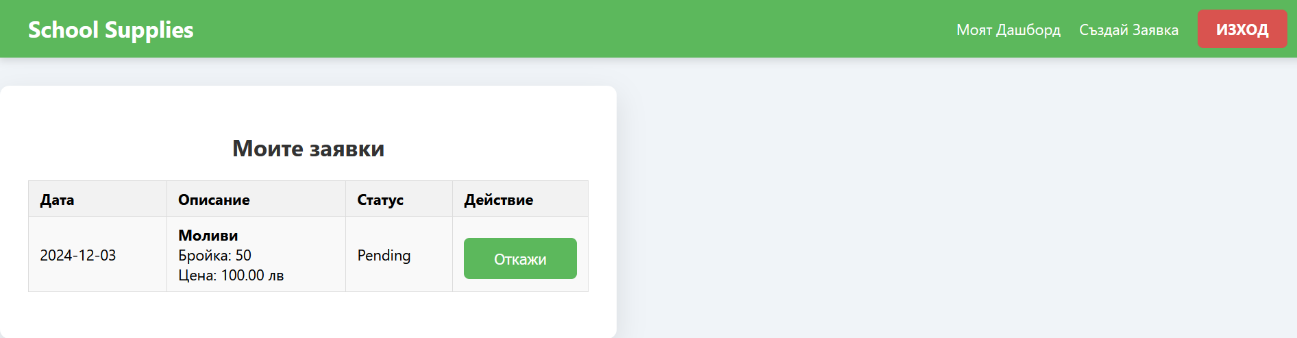
Форма за добавяне на консумативи „Фигура 5“:

Съдържа полета, в които потребителят въвежда информация за заявката:

* **Поле „Име на консуматив“:** Потребителят въвежда наименованието на консуматива.
* **Поле „Бройка“:** Потребителят въвежда количеството на съответния консуматив. Полето приема само числови стойности.
* **Поле „Цена“:** Потребителят въвежда цената на единичен брой от консуматива. Полето поддържа десетични стойности.
* **Бутон „Добави консуматив“:** Позволява на потребителя да добави нови редове за въвеждане на допълнителни консумативи. При всяко натискане се създава нов набор от полета за въвеждане.
* **Бутон „Подай заявка“:** Изпраща въведената информация за обработка. След успешното създаване на заявката потребителят се пренасочва към страницата за управление на заявките.
* **Навигационна лента:** В горната част на страницата се намира лента с опции за бърз достъп:
* **„Моят Дашборд“:** Води до основния панел на потребителя.
* **„Създай Заявка“:** Текущата страница за създаване на нова заявка.
* **„Изход“:** Излизане от акаунта.

Този интерфейс осигурява лесен и интуитивен начин за потребителите да подават заявки за консумативи, като им дава възможност да добавят няколко артикула наведнъж.

## 4.4. Преглед и управление на заявки (от страна на потребителя)

На предоставения интерфейс „Фигура 6“ е показана страницата за управление на подадените заявки на потребителя. Тя включва следните елементи:

Фигура 6: Преглед и управление на заявки (user)

На предоставения интерфейс е показана страницата за управление на подадените заявки на потребителя. Тя включва следните елементи:

Таблица с подробности за заявките: Основният елемент на страницата е таблица, която представя информация за всички заявки на потребителя. Включва следните колони:

* **„Дата“:** Показва датата на създаване на заявката.
* **„Описание“:** Подробности за консумативите в заявката, включително:
* **Име на консуматив**
* **Бройка на всеки консуматив**
* **Цена на консуматива в лева**
* **„Статус“:** Показва текущото състояние на заявката (например „В очакване“, „Одобрена“, „Отхвърлена“).
* **„Действие“:** Предоставя опция за отказване на заявка, ако статусът ѝ е „В очакване“. Включва:
* **Бутон „Откажи“:** Потребителят може да отмени заявка, като кликне този бутон. Появява се потвърдително съобщение за сигурност.

Ако заявката вече е обработена (например със статус „Одобрена“ или „Отхвърлена“), вместо бутон се показва текстът „Не може да бъде отказана“.

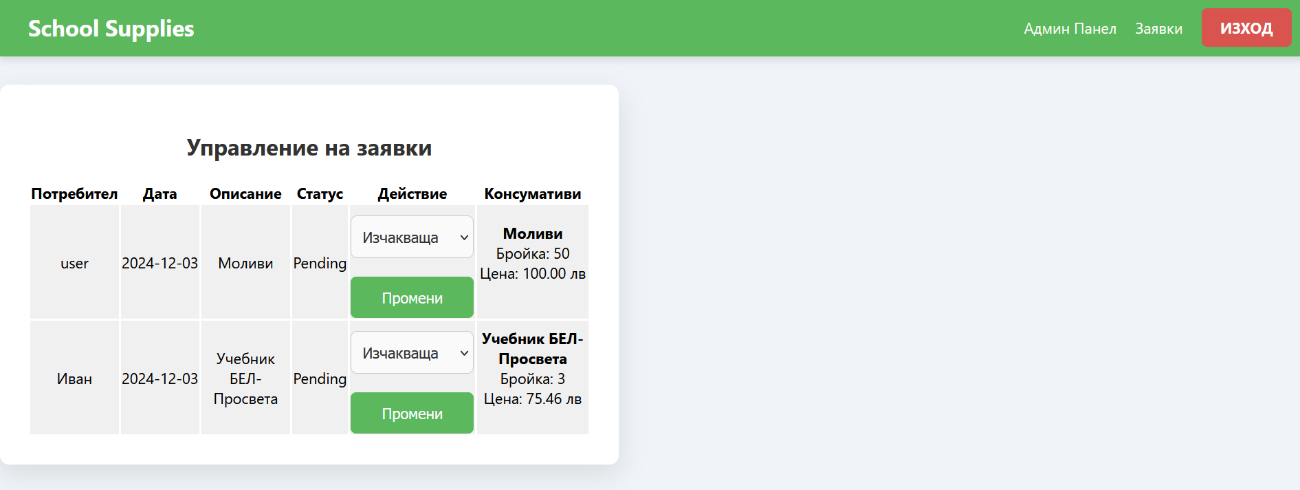
**Навигационна лента:**

В горната част на страницата се намира лента с опции за навигация:

* **„Моят Дашборд“:** Връща потребителя към основния му панел.
* **„Създай Заявка“:** Позволява на потребителя да създаде нова заявка.
* **„Изход“:** Излизане от акаунта.

Този интерфейс осигурява удобен преглед на подадените заявки и позволява на потребителите да отменят заявки, които все още са в процес на разглеждане.

## 4.5. Администраторско управление на заявки

На предоставения скрийншот „Фигура 7“ е показан интерфейс за администраторско управление на заявки. Страницата позволява на администратора да преглежда и управлява подадените заявки за консумативи от потребителите. Тя включва следните елементи:

Фигура 7: Управление на заявки (admin)

**Основни компоненти във „Фигура 7“:**

1. **Таблица „Управление на заявки“:** Основната таблица показва всички подадени заявки и включва следните колони:

* **„Потребител“:** Показва името на потребителя, който е подал заявката.
* **„Дата“:** Датата, на която е създадена заявката.
* **„Описание“:** Кратко описание на заявката, което обобщава консумативите.
* **„Статус“:** Текущото състояние на заявката. В този пример статусът е „Pending“ (в очакване).
* **„Действие“:** Администраторът може да промени статуса на заявката чрез падащо меню.
  + **Падащо меню „Изчакваща“:** Позволява промяна на статуса (например на **„Одобрена“** или **„Отхвърлена“**).
  + **Бутон „Промени“:** Запазва промените, направени в статуса на заявката.
* **„Консумативи“:**

Детайлна информация за заявените консумативи, включително:

* **Име на консуматива:** Например „Моливи“ или „Учебник БЕЛ-Просвета“.
* **Бройка:** Показва количеството на заявените консумативи.
* **Цена**: Показва цената на консумативите в лева (лв).

Този интерфейс предоставя удобен и функционален начин за администраторите да наблюдават заявките на потребителите, да преглеждат техните детайли и да променят статуса на всяка заявка при необходимост.

# Заключение

Настоящият проект представлява значителен принос към модернизацията на управлението на учебните ресурси в училищна среда. Чрез разработването на приложението за управление на заявки за материали, проектът демонстрира ефективно съчетаване на съвременни уеб технологии (HTML, CSS, JavaScript) с цел оптимизиране на административните процеси. Той осигурява интуитивен интерфейс за подаване и проследяване на заявки, подобрява комуникацията между различните звена в училището и създава прозрачна система за управление на наличностите.

Проектът не само улеснява ежедневната работа на училищната администрация, но също така допринася за по-добро планиране и изразходване на ресурсите. Това решение повишава ефективността и отчетността, като в крайна сметка подкрепя учебния процес и създава условия за по-успешно управление на материалните потребности.

В заключение, приложението е ключова стъпка към дигитализацията и модернизацията на училищната администрация, което отговаря на съвременните изисквания за ефективно управление на ресурси в образователните институции.

# Използвани Източници

PHP.net. PHP Manual – Documentation. Достъпно към 07.12.2024г. от <https://www.php.net/manual/en/>

MySQL.com. MySQL – Documentation. Достъпно към 07.12.2024г. от <https://dev.mysql.com/doc/>

MDN Web Docs. MDN Documenting Web Technologies. Достъпно към 07.12.2024г. от <https://developer.mozilla.org/en-US/>

W3Schools.com. PHP MySQL Database. Достъпно към 07.12.2024г. от <https://www.w3schools.com/php/php_mysql_intro.asp>

# Приложение 1

**Логика за регистрация в PHP:**

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

    $username = trim($\_POST['username']);

    $password = trim($\_POST['password']);

    $role = 'user'; // Default role for new users

    if (empty($username) || empty($password)) {

        $error = "Всички полета са задължителни!";

    } else {

        // Check if the username already exists

        $stmt = $conn->prepare("SELECT id FROM users WHERE username = ?");

        $stmt->bind\_param("s", $username);

        $stmt->execute();

        $stmt->store\_result();

        if ($stmt->num\_rows > 0) {

            $error = "Потребителското име вече съществува!";

        } else {

            // Hash the password

            $hashedPassword = password\_hash($password, PASSWORD\_DEFAULT);

            // Insert the new user

            $stmt = $conn->prepare("INSERT INTO users (username, password, role) VALUES (?, ?, ?)");

            $stmt->bind\_param("sss", $username, $hashedPassword, $role);

            $stmt->execute();

            header("Location: index.php");

            exit();

        }

}

}

# Приложение 2

**PHP код за актуализация на статуса:**

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

    $user\_id = $\_SESSION['user\_id'];

    $items = $\_POST['items'];

    // Create a dynamic description based on the first item name

    $description = "";

    if (!empty($items[0]['name'])) {

        $description = $items[0]['name'];  // Using the name of the first item for the description

    }

    // Insert the main request into the 'requests' table

    $stmt = $conn->prepare("INSERT INTO requests (user\_id, description) VALUES (?, ?)");

    $stmt->bind\_param("is", $user\_id, $description);

    $stmt->execute();

    $request\_id = $conn->insert\_id; // Get the ID of the inserted request

    // Insert each item into the 'request\_items' table

    $stmtItem = $conn->prepare("INSERT INTO request\_items (request\_id, item\_name, quantity, price) VALUES (?, ?, ?, ?)");

    foreach ($items as $item) {

        $stmtItem->bind\_param("isid", $request\_id, $item['name'], $item['quantity'], $item['price']);

        $stmtItem->execute();

    }

    // Redirect the user to the user management page

    header("Location: user\_management.php");

    exit();

}

# Приложение 3

**PHP код за актуализация на статуса:**

if ($\_SESSION['role'] != 'admin') {

    header('Location: dashboard.php');

    exit();

}

// Handle status updates

if (isset($\_POST['update\_status'])) {

    $status = $\_POST['status'];

    $request\_id = $\_POST['request\_id'];

    $stmt = $conn->prepare("UPDATE requests SET status = ? WHERE id = ?");

    $stmt->bind\_param("si", $status, $request\_id);

    $stmt->execute();

}

// Fetch requests with their associated items

$result = $conn->query("

    SELECT r.id AS request\_id, u.username, r.request\_date, r.description, r.status,

           ri.item\_name, ri.quantity, ri.price

    FROM requests r

    JOIN users u ON r.user\_id = u.id

    JOIN request\_items ri ON r.id = ri.request\_id

    ORDER BY r.id, ri.id

");

// Group items by request\_id

$requests = [];

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

    $requests[$row['request\_id']]['details'] = [

        'username' => $row['username'],

        'request\_date' => $row['request\_date'],

        'description' => $row['description'],

        'status' => $row['status']

    ];

    $requests[$row['request\_id']]['items'][] = [

        'item\_name' => $row['item\_name'],

        'quantity' => $row['quantity'],

        'price' => $row['price']

    ];

}