|  |  |  |
| --- | --- | --- |
| **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: viktorsotirov21a@gmail.com

boyankoprinkov21a@gmail.com

g.genchev39@gmail.com

kristiyantsenov21a@gmail.com

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)

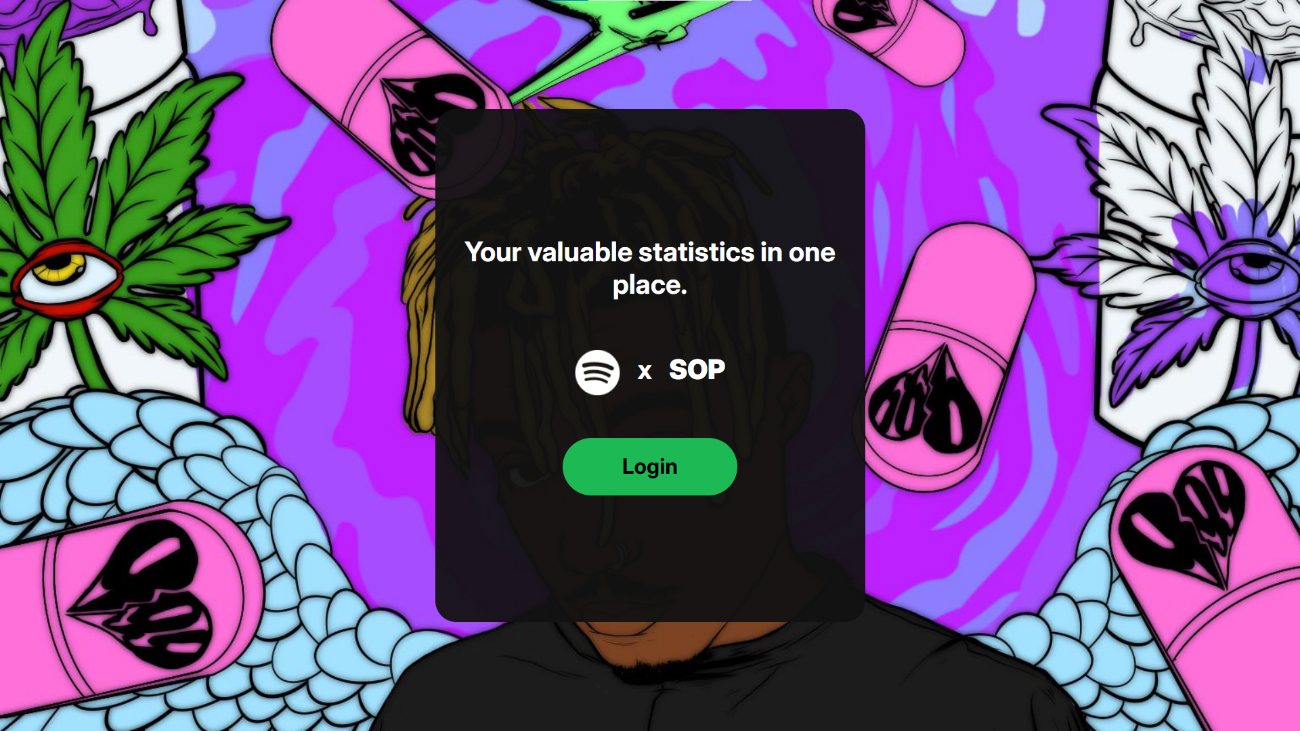
OAuth 2.0 е протокол за удостоверяване, който позволява сигурен достъп до потребителски данни без споделяне на пароли. В този проект се използва за логване чрез Spotify – потребителите влизат с профила си и предоставят достъп до информацията, необходима за функционирането на приложението.

# Глава 2 – Основни функционалности

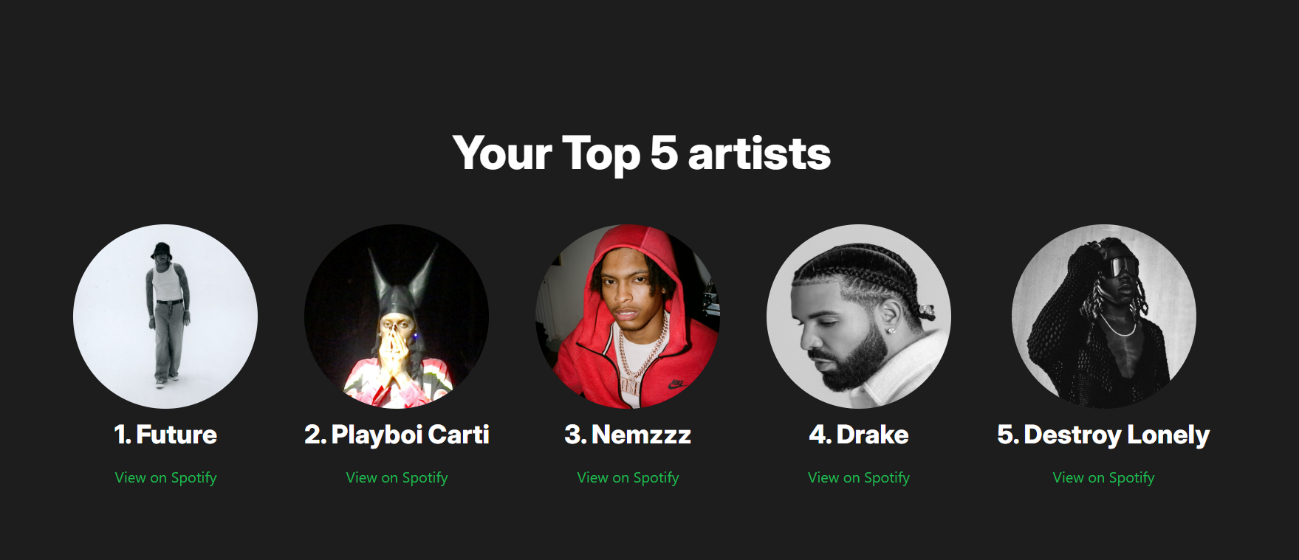
Spotify Data Tracker предлага поредица от персонализирани функционалности, които дават на потребителя достъп до детайлна информация за неговите слушателски навици в Spotify. След влизане в приложението чрез своя Spotify акаунт, потребителят може да използва следните функции.

## 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:

* 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

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

public class AuthController : Controller

{

public IActionResult Login()

{

return Challenge(new AuthenticationProperties { RedirectUri = "/" }, "Spotify");

}

public IActionResult Logout()

{

return SignOut(new AuthenticationProperties { RedirectUri = "/" },

CookieAuthenticationDefaults.AuthenticationScheme);

}

}

public class SpotifyController : Controller

{

private readonly SpotifyService \_spotifyService;

public SpotifyController(SpotifyService spotifyService)

{

\_spotifyService = spotifyService;

}

public async Task<IActionResult> TopArtists()

{

var accessToken = await HttpContext.GetTokenAsync("access\_token");

if (accessToken == null)

{

return RedirectToAction("Login", "Auth");

}

var artists = await \_spotifyService.GetTopArtistsAsync(accessToken);

return View(artists);

}

public IActionResult Dashboard()

{

return View();

}

}

## 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

<div class="login">

<div class="login-hero">

<h1 class="title">Your valuable statistics in one place.</h1>

<div class="hero-icons">

<**img** class="spotify-logo" src="~/media/spotify.png"/>

<h1 style="font-family: Inter, sans-serif; font-size: 32px; font-weight: 700; color: white;">

x

</h1>

<h1 class="sop-logo">SOP</h1>

</div>

@if (User.Identity.IsAuthenticated)

{

<**a** **asp-controller**="Auth" **asp-action**="Logout">

<div class="hero-button">

<h1 class="button-title">Logout</h1>

</div>

</**a**>

}

else

{

<**a** **asp-controller**="Auth" **asp-action**="Login">

<div class="hero-button">

<h1 class="button-title">Login</h1>

</div>

</**a**>

}

@if (User.Identity.IsAuthenticated)

{

<**a** **asp-controller**="Spotify" **asp-action**="TopArtists">

<div class="hero-button">

<h1 class="button-title">Dashboard</h1>

</div>

</**a**>

}

</div>

</div>

* Dashboard.cshtml – основна страница със статистики
* Recommendations.cshtml – страница с персонализирани предложения
* TopArtists.cshtml – страница с петте най-слушани артисти

<div class="artists">

<h1 class="artists-heading">Your Top 5 artists</h1>

<div class="artists-hero">

@for (int i = 0; i < Model.Count; i++)

{

var artist = Model[i];

<div class="artists-cont">

@if (artist.Images != null && artist.Images.Any())

{

<img src="@artist.Images[0].Url" alt="@artist.Name" class="artists-image" />

}

<h2 class="artists-name">@(i + 1). @artist.Name</h2>

<a href="@artist.ExternalUrls?.Spotify" target="\_blank" class="artists-redirect">View on Spotify</a>

</div>

}

</div>

</div>

* 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:

* Изпраща заявки към нужните endpoint-и
* Обработва и парсва JSON отговорите
* Преобразува данните в обекти, използвани от контролерите и изгледите

Тази абстракция улеснява работата със Spotify и прави кода по-четим и модулен.

public class SpotifyService

{

private readonly IHttpClientFactory \_httpClientFactory;

public SpotifyService(IHttpClientFactory httpClientFactory)

{

\_httpClientFactory = httpClientFactory;

}

public async Task<List<Artist>?> GetTopArtistsAsync(string accessToken)

{

var client = \_httpClientFactory.CreateClient();

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", accessToken);

var response = await client.GetAsync("https://api.spotify.com/v1/me/top/artists?limit=5");

if (!response.IsSuccessStatusCode)

{

return null;

}

var json = await response.Content.ReadAsStringAsync();

var data = JsonSerializer.Deserialize<TopArtistsResponse>(json);

return data?.Items;

}

}

public class TopArtistsResponse

{

[JsonPropertyName("items")]

public List<Artist>? Items { get; set; }

}

public class Artist

{

[JsonPropertyName("name")]

public string? Name { get; set; }

[JsonPropertyName("images")]

public List<Image>? Images { get; set; }

[JsonPropertyName("external\_urls")]

public ExternalUrls? ExternalUrls { get; set; }

}

public class Image

{

[JsonPropertyName("url")]

public string? Url { get; set; }

}

public class ExternalUrls

{

[JsonPropertyName("spotify")]

public string? Spotify { get; set; }

}



# Глава 4 - Работа с Spotify API

Приложението използва официалния Spotify Web API, който предоставя достъп до различни данни на потребителя чрез HTTPS заявки. За достъп до тези ресурси е необходимо удостоверяване чрез OAuth 2.0.

## 4.1. Получаване на достъп (access token)

За да може приложението да извлича лични данни от Spotify, потребителят трябва да се логне със своя акаунт. Процесът включва следните стъпки:

## Пренасочване към Spotify Login URL, където потребителят одобрява достъпа.

1. След одобрение, Spotify връща authorization code към нашето приложение.
2. Приложението разменя този код за access token и refresh token, чрез POST заявка към Spotify token endpoint:

POST https://accounts.spotify.com/api/token

1. С получения access token, приложението може да извършва заявки от името на потребителя за ограничено време. Refresh token-ът се използва за подновяване на достъпа без нов логин.

**Използвани scopes** (разрешения):

* user-top-read – достъп до топ песни и артисти
* user-read-recently-played – последно слушани песни
* user-read-email, user-read-private – основна потребителска информация
* user-read-playback-position, user-library-read – при нужда от разширени препоръки

## 4.2. Използвани основни endpoints

| **Endpoint** | **Описание** |
| --- | --- |
| GET https://api.spotify.com/v1/me | Взема основна информация за потребителя |
| GET https://api.spotify.com/v1/me/top/artists | Връща най-слушаните артисти на потребителя |
| GET https://api.spotify.com/v1/me/top/tracks | Връща най-слушаните песни на потребителя |
| GET https://api.spotify.com/v1/me/player/recently-played | Връща списък с последно слушани песни |
| GET https://api.spotify.com/v1/recommendations | Генерира персонализирани препоръки на база артист/песен/жанр |
| GET https://api.spotify.com/v1/artists/{id} | Подробна информация за артист, включително жанрове |
| GET https://api.spotify.com/v1/tracks/{id} | Подробна информация за песен |

Заявките се изпращат чрез HTTP с включен Authorization header:

* Authorization: Bearer {access\_token}

Всички отговори са в JSON формат и се обработват в SpotifyService за конвертиране към модели.

# Глава 5 - Дизайн и потребителски интерфейс

Интерфейсът на приложението Spotify Data Tracker е изграден с цел да бъде интуитивен, визуално чист и фокусиран върху статистиките на потребителя. Използвани са Razor изгледи, съчетани със CSS и JavaScript за динамични елементи.

## 5.1. Основна навигация

След успешен вход потребителят се насочва към Dashboard, който служи като централен хъб. Навигацията е проста и се осъществява чрез горно меню или бутони, които водят до следните изгледи:

* Dashboard (Главна страница):  
  Съдържа кратка информация за потребителя, топ 5 песни, топ 5 артисти, както и кратко обобщение на жанровете и последно слушаните парчета. Този изглед се зарежда автоматично след логване.
* Top Artists (Топ изпълнители):

Специален изглед, където се показват детайли за петте най-слушани артисти, включително жанрове, изображение и линкове към Spotify.

* **Top Tracks (Топ песни):**  
  Показва най-слушаните песни с обложка, име, изпълнител и продължителност. Налични са бутони за прослушване в Spotify.
* **Recently Played (Последно слушани):**  
  Времева линия с най-новите слушания на потребителя. Изгледът използва информация от recently-played endpoint-а.
* **Recommendations (Препоръки):**  
  Визуално представяне на персонализирани предложения за нови песни и артисти въз основа на предпочитанията на потребителя.

## 5.2. Визуални елементи

* Използван е **респонсив дизайн** за добра работа на различни устройства.
* Основните цветове и фон са съобразени със стилистиката на Spotify – тъмна тема, зелени и бели акценти.
* Карти с артисти и песни съдържат изображения и бутони за действия (напр. „Play in Spotify“).
* Има индикатори за зареждане и съобщения при грешки или липса на данни.

# Заключение

Проектът **Spotify Data Tracker** демонстрира ефективна интеграция на външен API в ASP.NET MVC приложение и предоставя реална стойност на потребителя чрез визуализиране на неговите музикални навици. Чрез сигурна автентикация, обработка на данни в реално време и изчистен интерфейс, приложението показва как модерни технологии могат да бъдат използвани за създаване на персонализирано дигитално изживяване.

**Основните цели бяха постигнати успешно:**

* Успешна връзка със Spotify чрез OAuth 2.0
* Извличане и представяне на ключови статистики
* Изграждане на адаптивен и лесен за използване потребителски интерфейс

### **Бъдещи подобрения**

Възможни посоки за надграждане на приложението включват:

* Визуализации чрез графики (напр. жанрове по процент)
* Историческо проследяване на слушанията през различни периоди
* Добавяне на възможност за създаване на автоматични плейлисти
* Интеграция с други музикални платформи (напр. Apple Music)

С този проект е направена важна стъпка към изграждане на реално уеб приложение с външни API услуги, обединяващо back-end логика, front-end изживяване и сигурна потребителска автентикация.

# 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']

    ];

}