

create rest api for android app using php and mysql

Published: September 27, 2026 | Index ID: #-

How to Create a REST API for an Android App Using PHP and MySQL

When you build an Android application that needs to store, retrieve, or manipulate data, you almost always need a backend server. A common, proven stack for this purpose is PHP on the server side paired with MySQL as the database. Together they provide a lightweight, cost effective solution that can be deployed on almost any hosting provider. In this guide we will walk you through every step of creating a full featured REST API that your Android app can consume. From setting up the development environment to securing your endpoints and deploying to production, this article covers all the essentials.

Why Use PHP and MySQL for Android Backends?

While there are many modern backend frameworks—Node.js, Python/Django, Ruby on Rails, Go, and more—PHP remains a popular choice for small to medium sized projects. Here are a few reasons why:

- **Widespread Hosting Support:** Almost every shared hosting plan includes PHP and MySQL out of the box.
- **Large Community & Resources:** Thousands of tutorials, libraries, and forums help you solve almost any problem.
- **Fast Development:** PHP's syntax is straightforward, allowing rapid prototyping.
- **Integrated Database Layer:** MySQL is the most common relational database, and PHP's PDO extension offers a secure, flexible API.
- **Cost Effective:** You can host PHP/MySQL applications on inexpensive VPS or even free tiers.

For Android developers, the combination of PHP and MySQL provides a simple, reliable way to expose data over HTTP. By following RESTful conventions, you can keep your API intuitive, making it easier for front end developers to consume.

Prerequisites Before You Start

Before diving into code, make sure you have the following ready:

1. **Development Environment:** A local web server stack such as XAMPP (Windows/Linux/macOS) or MAMP for macOS. Alternatively, use Docker to isolate your environment.
2. **MySQL Client:** phpMyAdmin or any MySQL client to manage databases.
3. **PHP 7.4 or newer:** Ensure your server supports the latest PHP version for security and performance.
4. **Android Studio:** For building and testing your Android app.
5. **Postman or Insomnia:** To test your API endpoints before hooking them up to the app.
6. **Basic Knowledge of HTTP methods, JSON, and SQL.**

Step 1: Setting Up the Local Server

We'll use XAMPP for this tutorial. Download and install XAMPP from <https://www.apachefriends.org>. Once installed, start the Apache and MySQL services.

Open a terminal or command prompt and navigate to the XAMPP installation directory. Create a new project folder inside htdocs called android_api:

```
cd /opt/lampp/htdocs
mkdir android_api
cd android_api
```

All your PHP files will live inside this folder.

Step 2: Designing the Database Schema

Let's assume we're building a simple task management app. We'll need two tables: users and tasks. Here's a quick diagram:

- users – id, username, password_hash, email, created_at
- tasks – id, user_id, title, description, status, due_date, created_at, updated_at

Open phpMyAdmin, create a new database named android_app_db, and run the following SQL to create the tables:

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50) NOT NULL UNIQUE,
  password_hash VARCHAR(255) NOT NULL,
  email VARCHAR(100) NOT NULL UNIQUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE tasks (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  title VARCHAR(255) NOT NULL,
  description TEXT,
  status ENUM('pending', 'in_progress', 'completed') DEFAULT 'pending',
  due_date DATE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Step 3: Configuring Database Connection in PHP

Create a file named config.php inside android_api. This file will store your database credentials and establish a PDO connection:

```
<?php
// config.php
$DB_HOST = 'localhost';
$DB_NAME = 'android_app_db';
$DB_USER = 'root';
```

```

$DB_PASS = "";

try {
    $pdo = new PDO("mysql:host=$DB_HOST;dbname=$DB_NAME;charset=utf8mb4", $DB_USER, $DB_PASS);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    http_response_code(500);
    echo json_encode(['error' => 'Database connection failed']);
    exit;
}
?>

```

All API scripts will include this file to gain access to the \$pdo object.

Step 4: Building the RESTful Endpoints

We'll create a single entry point, api.php, that will route incoming requests based on the URL path and HTTP method. This keeps the structure clean and scalable.

4.1 Basic Routing Logic

Below is the skeleton of api.php:

```

<?php
require 'config.php';

header('Content-Type: application/json');
header('Access-Control-Allow-Origin: *'); // Adjust for production
header('Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS');
header('Access-Control-Allow-Headers: Content-Type, Authorization');

// Handle preflight CORS requests
if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') {
    http_response_code(200);
    exit;
}

$uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
$uri = str_replace('/android_api/api.php', '', $uri);
$uri = trim($uri, '/');
$uri_parts = explode('/', $uri);

$method = $_SERVER['REQUEST_METHOD'];

// Dispatch based on first URI segment
switch ($uri_parts[0]) {

```

```

case 'users':
    handleUsers($method, $uri_parts);
    break;
case 'tasks':
    handleTasks($method, $uri_parts);
    break;
default:
    http_response_code(404);
    echo json_encode(['error' => 'Endpoint not found']);
}
?>

```

4.2 Handling User Endpoints

We'll support the following operations on /users:

- POST – Register a new user.
- GET – Retrieve a list of users (admin only) or a single user.

Append the following function to api.php:

```

function handleUsers($method, $uri_parts) {
    global $pdo;
    switch ($method) {
        case 'POST':
            registerUser();
            break;
        case 'GET':
            getUsers();
            break;
        default:
            http_response_code(405);
            echo json_encode(['error' => 'Method not allowed']);
    }
}

function registerUser() {
    $data = json_decode(file_get_contents('php://input'), true);
    if (!isset($data['username'], $data['password'], $data['email'])) {
        http_response_code(400);
        echo json_encode(['error' => 'Missing required fields']);
        return;
    }

    $username = trim($data['username']);
    $email = trim($data['email']);

```

```

$password_hash = password_hash($data['password'], PASSWORD_BCRYPT);

try {
    $stmt = $pdo->prepare('INSERT INTO users (username, email, password_hash) VALUES (?, ?, ?)');
    $stmt->execute([$username, $email, $password_hash]);

    http_response_code(201);
    echo json_encode(['message' => 'User created', 'user_id' => $pdo->lastInsertId()]);
} catch (PDOException $e) {
    http_response_code(409);
    echo json_encode(['error' => 'User already exists']);
}
}

function getUsers() {
    // For demo, return all users; in production, restrict to admins
    $stmt = $pdo->query('SELECT id, username, email, created_at FROM users');
    $users = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($users);
}

```

4.3 Handling Task Endpoints

The /tasks endpoint will support:

- POST – Create a new task.
- GET – Retrieve tasks for a user or a single task.
- PUT – Update a task.
- DELETE – Delete a task.

Add the following to api.php:

```

function handleTasks($method, $uri_parts) {
    global $pdo;
    switch ($method) {
        case 'POST':
            createTask();
            break;
        case 'GET':
            getTasks($uri_parts);
            break;
        case 'PUT':
            updateTask($uri_parts);
            break;
        case 'DELETE':
            deleteTask($uri_parts);

```

```

        break;
    default:
        http_response_code(405);
        echo json_encode(['error' => 'Method not allowed']);
    }
}

```

```

function createTask() {
    $data = json_decode(file_get_contents('php://input'), true);
    if (!isset($data['user_id'], $data['title'])) {
        http_response_code(400);
        echo json_encode(['error' => 'Missing required fields']);
        return;
    }
}

```

```

$stmt = $pdo->prepare('INSERT INTO tasks (user_id, title, description, status, due_date) VALUES
(?, ?, ?, ?, ?)');
$stmt->execute([
    $data['user_id'],
    $data['title'],
    $data['description'] ?? null,
    $data['status'] ?? 'pending',
    $data['due_date'] ?? null
]);

http_response_code(201);
echo json_encode(['message' => 'Task created', 'task_id' => $pdo->lastInsertId()]);
}

```

```

function getTasks($uri_parts) {

```

Baca Juga (Artikel Terkait)

- [cinco personas que encontraras en el cielo las](http://www.bbglobalsolutions.com/cinco-personas-que-encontraras-en-el-cielo-las.pdf)

URL: <http://www.bbglobalsolutions.com/cinco-personas-que-encontraras-en-el-cielo-las.pdf>

- [cinco megatendencias y sus posibles implicaciones](http://www.bbglobalsolutions.com/cinco-megatendencias-y-sus-posibles-implicaciones.pdf)

URL: <http://www.bbglobalsolutions.com/cinco-megatendencias-y-sus-posibles-implicaciones.pdf>

- [cinco horas con mario](http://www.bbglobalsolutions.com/cinco-horas-con-mario.pdf)

URL: <http://www.bbglobalsolutions.com/cinco-horas-con-mario.pdf>

- [cincinnati public schools pages](http://www.bbglobalsolutions.com/cincinnati-public-schools-pages.pdf)

URL: <http://www.bbglobalsolutions.com/cincinnati-public-schools-pages.pdf>

Tags: #create #rest #android #using #mysql

