ОТЧЁТ О РАЗРАБОТКЕ MINI-API НА PHP (CRUD) С ТЕСТИРОВАНИЕМ ЧЕРЕЗ VS CODE*

Кинстлер Александра

1. Цель работы

Цель данной работы — создать минимальный API-сервис на PHP, реализующий операции CRUD (Create, Read, Update, Delete) для одной сущности tasks (задачи), подключенный к базе данных MySQL, и протестировать API с помощью встроенного инструмента *Thunder Client в Visual Studio Code*.

Дополнительно — научиться:

- создавать базу данных и таблицы в phpMyAdmin,
- работать с маршрутизацией (роутинг),
- формировать HTTP-запросы (GET, POST, PUT, DELETE),
- возвращать ответы в формате JSON,
- организовывать структуру проекта.

2. Используемые инструменты и технологии

Технология / инструмент	**Назначение**
PHP 8	Backend логика API
MySQL / MariaDB	Хранение данных
phpMyAdmin	Управление БД
XAMPP	Локальный веб-сервер (Apache + MySQL)
VS Code	Разработка
Thunder Client (VS Code)	Тестирование API
JSON	Формат обмена данными

3. Постановка задачи

Необходимо разработать:

- API для сущности tasks;
- операции:

Метод	URL	Описание
GET	/API/tasks	Получить список задач
GET	/API/tasks/{id}	Получить одну задачу
POST	/API/tasks	Создать новую задачу
PUT/PATCH	/API/tasks/{id}	Обновить задачу
DELETE	/API/tasks/{id}	Удалить задачу

API должен:

- работать через XAMPP
- подключаться к базе MySQL
- возвращать ответы строго в JSON
- обеспечивать обработку ошибок

4. Структура проекта

После подготовки файлов структура выглядит так:

C:\xampp\htdocs\API

│ ├— index.php ├— Database.php ├— TaskController.php ├— config.php └— .htaccess

Описание:

✓ index.php

Главная точка входа. Роутер. Обрабатывает HTTP-методы.

✓ ***Database.php***

Класс подключения к MySQL через PDO.

✓ ***TaskController.php***

CRUD-методы: index(), show(), store(), update(), destroy().

✓ ***config.php***

Настройки подключения к базе.

✓ ***.htaccess***

Перенаправляет все запросы на index.php.

5. Настройка окружения

✓ **Запустили XAMPP:**

- Apache
- MySQL

✓ **Перешли в phpMyAdmin**

Адрес:

http://localhost/phpmyadmin

✓ **Создали базу данных**

Название: mini_api *Кодировка:* utf8mb4_general_ci

6. Создание таблицы tasks

Вкладка *SQL* → ввод команды:

sql CREATE TABLE tasks (id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY, title VARCHAR(255) NOT NULL, description TEXT NULL, is_done TINYINT(1) NOT NULL DEFAULT 0, created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);

После выполнения таблица появилась в списке базы данных.

7. Разработка API

7.1. config.php

Файл хранит параметры подключения:

php <?php return ['db' => ['host' => '127.0.0.1', 'dbname' => 'mini_api', 'user' => 'root', 'password' => '', 'charset' => 'utf8mb4',],];

7.2. Database.php

Создание подключения через PDO:

```
php class Database { private \PDO $pdo;

public function __construct(array $config)
{
   $dsn = sprintf(
      'mysql:host=%s;dbname=%s;charset=%s',
      $config['host'],
      $config['dbname'],
      $config['charset']
   );

   $this->pdo = new PDO($dsn, $config['user'], $config['password'], [
      PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
      PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
   ]);
}

public function getConnection(): PDO
{
   return $this->pdo;
```

}

}

7.3. TaskController.php

Реализованы методы:

- index() — список задач
- show() — получить по ID
- store() — создание
- update() — обновление
- destroy() — удаление

Пример (фрагмент):

php public function store(array $data): array { if (empty($data['title'])) { throw new InvalidArgumentException('Поле "title" обязательно.'); }

```
$stmt = $this->db->prepare(
   'INSERT INTO tasks (title, description, is_done)
    VALUES (:title, :description, :is_done)'
);

$stmt->execute([
   'title' => $data['title'],
   'description' => $data['description'] ?? null,
   'is_done' => $data['is_done'] ?? 0,
]);

$id = $this->db->lastInsertId();
return $this->show($id);


}
```

7.4. Главный файл index.php

Здесь происходит:

✓ чтение HTTP-метода ✓ разбор URL ✓ вызов соответствующей функции ✓ возврат JSON

8. Настройка .htaccess

Для работы маршрутов:

apache RewriteEngine On RewriteCond %{REQUEST_FILENAME} !-f RewriteCond %{REQUEST_FILENAME} !-d RewriteRule ^ index.php [QSA,L]

9. Тестирование API в VS Code (Thunder Client)

Было установлено расширение *Thunder Client*:

➞ Extensions → *Thunder Client* → Install

9.1. Тестирование GET запроса

Метод: GET URL:

http://localhost/API/tasks

Полученный результат, если таблица пустая:

json []

9.2. Тестирование POST (создание задачи)

Вкладка *Body → JSON*

json { "title": "Первая задача", "description": "Проверка API", "is_done": 0 }

Результат:

json { "id": 1, "title": "Первая задача", "description": "Проверка API", "is_done": 0, "created_at": "2025-12-10 18:20:00" }

9.3. Тестирование DELETE

Метод:

DELETE http://localhost/API/tasks/1

Ответ:

json { "message": "Task deleted" }

10. Результаты работы

В ходе работы было успешно выполнено следующее:

✓ Установлен и настроен локальный сервер (XAMPP)

✓ Создана база данных MySQL

✓ Разработан полноценный мини-API на PHP

✓ Реализованы все CRUD операции

✓ Написана система обработки ошибок

✓ API протестирован в VS Code

✓ Данные корректно создаются, читаются, обновляются и удаляются

✓ Ответы всегда возвращаются в формате JSON

Проект полностью функционирует и может быть расширен, например:

- добавление авторизации JWT,

- пагинации,
- фильтрации задач,
- клиентского интерфейса.