|  |  |
| --- | --- |
| Логотип КБК копия | **Автономная некоммерческая организация профессионального образования**  **КАЛИНИНГРАДСКИЙ БИЗНЕС-КОЛЛЕДЖ** |

Кафедра информационных технологий

Специальность: 09.02.07 Информационные системы и программирование

**Курсовая работа**

На тему: **«Проектирование Базы Данных, создания и тестирование API»**

По дисциплине: **«**05.01 Проектирование и дизайн ИС**»**

Выполнил студент

группы 23-ИСП-3/2

Солобоев Д. Д.

Проверил:

Дубинин А. В.

Оценка:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Калининград**

**2024**

|  |  |
| --- | --- |
|  | **Автономная некоммерческая организация профессионального образования**  **КАЛИНИНГРАДСКИЙ БИЗНЕС-КОЛЛЕДЖ** |

Логотип КБК копияСпециальность 09.02.07 Информационные системы и программирование  
Форма обучения: Очная  
Группа 23- ИСП- 3/2

**ЗАДАНИЕ**  
по курсовой работе

студенту Солобоеву Даниилу Денисовичу

1. Тема курсовой работу «Проектирование Базы Данных, создания и тестирование API»

2. Срок сдачи готовой работы на кафедру «25» декабря 2024 г.

3. Цели работы (исходные данные): разработать базу данных для аудита фирмы для вычисления зарплаты. Написать API и протестировать через Postman.

4. Содержание разделов курсовой работы

[Глава 1. Проектирование БД](#_Toc43291210)

[1.1. Концептуальное проектирование](#_Toc43291211)

[1.2. Логическое проектирование](#_Toc43291212)

[1.3. Физическое проектирование](#_Toc43291213)

[Глава 2. Разработка API](#_Toc43291227)

[2.1. Подключение базы данных к API](#_Toc43291228)

[2.2. Подключение таблиц](#_Toc43291229)

[2.3. Функции API](#_Toc43291230)

[2.4. Открытие API на Open server](#_Toc43291232)

[Глава 3. Тестирование API с помощью Postman](#_Toc43291235)

[3.1. Тестирование функции create](#_Toc43291236)

3.2. Тестирование функции read

3.3. Тестирование функции update

3.4. Тестирование функции delete.

3.5. Тестирование функции search

5. Перечень графического и (или) иллюстрационного материала: Условия данные для создания БД. Графические примеры схем концептуального и логического проектирования. Итоговые варианты концептуального, логического и физического проектирования. Интерфейсы программ что используются для реализации курсовой работы. Изображения скриптов API. Результаты тестирования.

6. Консультанты по разделам дипломного проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Раздел | Ф.И.О. Консультанта | Подпись | Дата |
| Изучение проектирования и реализация базы данных. | Дубинин А. В. |  |  |
| Изучение и написание API. | Дубинин А. В. |  |  |
| Тестирование API с помощью Postman. | Дубинин А. В. |  |  |

7. Дата выдачи задания «1» ноября 2024 г.

8. Руководитель дипломного проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (Ф.И.О.)

9. Задание получил «1» ноября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

Оглавление

[1. Введение 5](#_Toc185941692)

[2. Проектирование базы данных 6](#_Toc185941693)

[2.1. Концептуальное проектирование 6](#_Toc185941694)

[2.2. Логическое проектирование 9](#_Toc185941695)

[2.3. Физическое проектирование 11](#_Toc185941696)

[3. Разработка API 13](#_Toc185941697)

[3.1. Подключение базы данных к API 14](#_Toc185941698)

[3.2. Подключение таблиц 15](#_Toc185941699)

[3.3. Функции API 17](#_Toc185941700)

[3.4. Открытие API на Open server 20](#_Toc185941701)

[4. Тестирование API с помощью Postman 21](#_Toc185941702)

[4.1. Тестирование функции create 22](#_Toc185941703)

[4.2. Тестирование функции read 24](#_Toc185941704)

[4.3. Тестирование функции update 24](#_Toc185941705)

[4.4. Тестирование функции delete 25](#_Toc185941706)

[4.5. Тестирование функции search 26](#_Toc185941707)

[Заключение 28](#_Toc185941708)

[Список использованных источников 29](#_Toc185941709)

# 1. Введение

База данных — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования.

API (от англ. application programming interface, дословно интерфейс программирования приложения) — программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими (в противоположность пользовательскому интерфейсу, используемому для взаимодействия конечного пользователя с программой). Обычно входит в описание какого-либо интернет-протокола (например, SCIM), программного каркаса (фреймворка) или стандарта вызовов функций операционной системы. Часто реализуется отдельной программной библиотекой или сервисом операционной системы. Используется программистами при написании всевозможных приложений.

Проще говоря, это способ взаимодействия какого-то программного кода с набором каких-то программных компонентов, с помощью которых одна компьютерная программа (например, бот или сайт) может использовать другую программу.

Была поставлена цель – Спроектировать и реализовать базу данных согласно данным условиям. Разработать API для спроектированной базы данных (В дальнейшем БД) и убедится в работоспособности путём тестирования через Postman.

На рисунке 1 вы можете наблюдать условия, данные нам для БД:

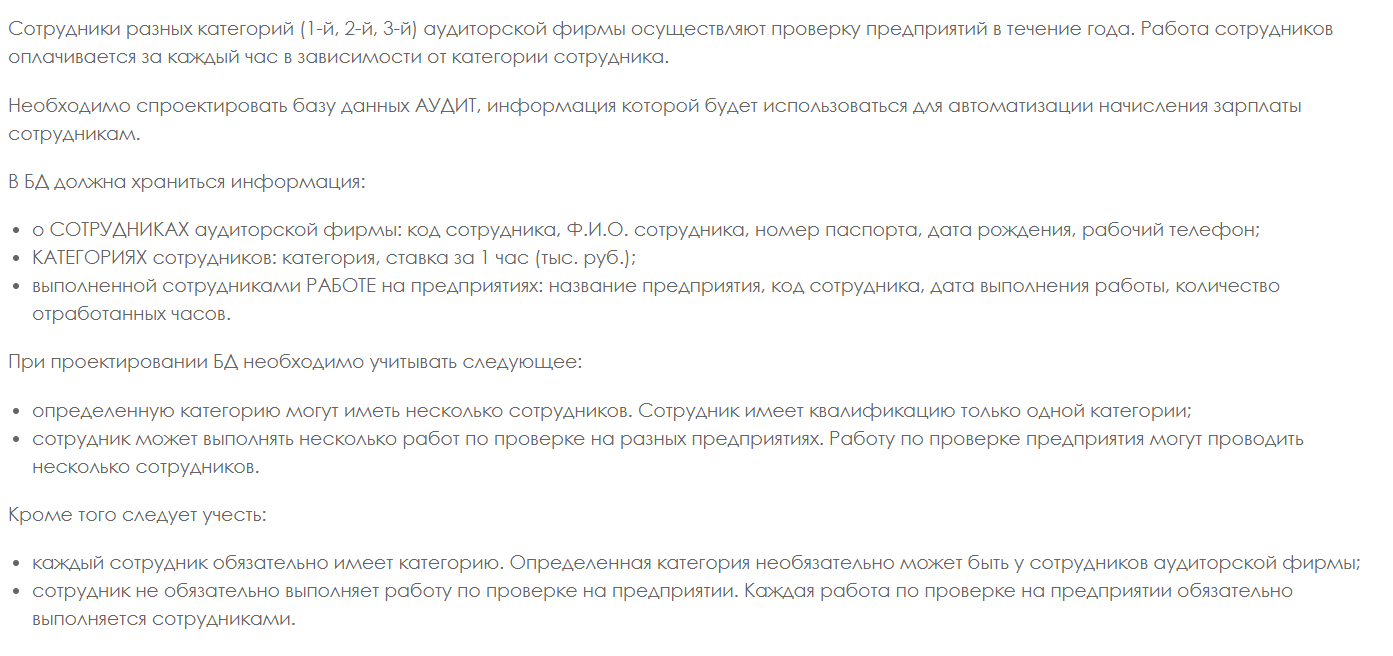


Рис.1.Данные условия для БД

Получив условия и цель, можно приступать к проектированию БД.

# 2. Проектирование базы данных

Проектирование БЖ состоит их трёх главных этапов:

1)Концептуальное проектирование - является первым этапом, на котором формируется общая идея будущей базы данных без учета физических аспектов ее реализации. Основная цель этого этапа - создание общей модели, которая отражает представления пользователей о процессах и данных, используемых в автоматизируемом участке компании.

2)Логическое проектирование - общая модель преобразуется в более подробную и структурированную форму. В этом процессе определяются сущности предметной области, их атрибуты и связи между ними.

3)Физическое проектирование - является последним этапом и включает выбор конкретной системы управления базами данных и определение физической реализации базы данных. На этом этапе оптимизируются доступ к данным, определяются таблицы, индексы, ограничения целостности и другие физические аспекты базы данных.

Важно выполнять эти этапы по порядку, ведь возвращаться к предыдущим уже будет нельзя, и на каждом этапе стоит быть очень внимательным.

## 2.1. Концептуальное проектирование

Чтобы приступить к концептуальному проектированию, для начала нужно изучить условия поподробнее и провести анализ предметной области. Выбрать основную информацию, выписать информацию, которая будет хранится в базе, а также условия, которые нужно учитывать при проектировании.

Наша основная задача: спроектировать БД АУДИТ, информация которой будет использоваться для автоматизации начисления зарплаты.

Предметная область: Аудиторская фирма

1)Каждый сотрудник может иметь только одну категорию, а в одной категории может быть несколько сотрудников.

2)На проверке предприятия могут быть несколько сотрудников, а каждый сотрудник может проводить несколько проверок разных предприятий.

3)Каждый сотрудник обязательно должен иметь категорию. Определённая категория необязательно может быть у сотрудников аудиторской фирмы.

4)Сотрудник не обязательно выполняет работу по проверке на предприятии, но каждая работа по проверке должна выполнятся сотрудником.

База должна включать в себя информацию:

1)О сотрудниках (Код сотрудника, Фамилию, Имя, Отчество, номер паспорта, дата рождения, рабочий телефон)

2)О категориях (Код категории, название категории, оплата в час)

3)О выполненной сотрудниками работе на предприятиях (Название предприятия, код сотрудника, дата выполнения работы, количество отработанных часов)

Само проектирование будет выполнено в форме ER-модели (Entity-Relationship model). Она также известна как модель «сущность-связь». Модель была разработана Питером Ченом и её цель заключается в создании понятной и наглядной модели, которая будет понята как неспециалистам, так и опытным программистам. Сама модель обычно изображается как простая блок схема, где каждая сущность и атрибут связаны между с собой линиями-связями.

На рисунке 2 вы можете наблюдать пример ER-модели:

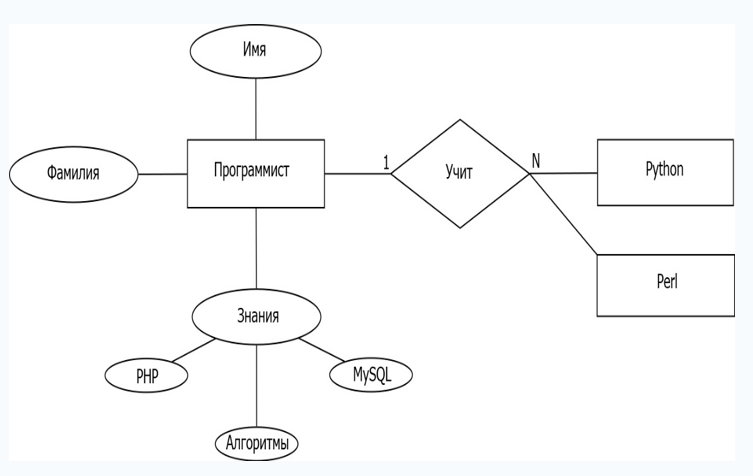


Рис.2.Пример ER-модели

На самой схеме в прямоугольниках изображаются сущности – таблицы в БД. В овалах изображаются атрибуты – столбцы таблиц, находящихся в БД. В ромбах обозначается связь двух сущностей.

Несколько сущностей связаны между собой несколькими видами связей:

1)Один к одному – когда из каждой таблицы формируется связь из данных, которые не могут использоваться повторно (Пример: Студент имеет только один студенческий билет. Студенческий билет может принадлежать только одному студенту)

2)Один ко многим – когда данные из одной таблицы могут использоваться повторно, а из второй нет (Пример: Студент может быть только в одной группе. Группа состоит из нескольких студентов)

3)Многие ко многим – данные из обеих таблиц могут использоваться неограниченное количество раз (Пример: Работник выполняет заказы. Заказ выполняется работниками). Такую связь невозможно создать напрямую. Поэтому в таких случаях используют ассоциативную таблицу. Эта таблица служит неким переходником, куда подключаются обе сущности по связи один ко многим, и вот уже в этой таблице хранятся повторяющиеся данные из обеих таблиц.

Разобрав всё необходимое, можно проектировать модель, основанную на данных условиях. В конце концов модель получилась из трёх сущностей и одной ассоциативной таблицы.

На рисунке 3 вы можете наблюдать итоговую ER-модель, удовлетворяющую условиям:

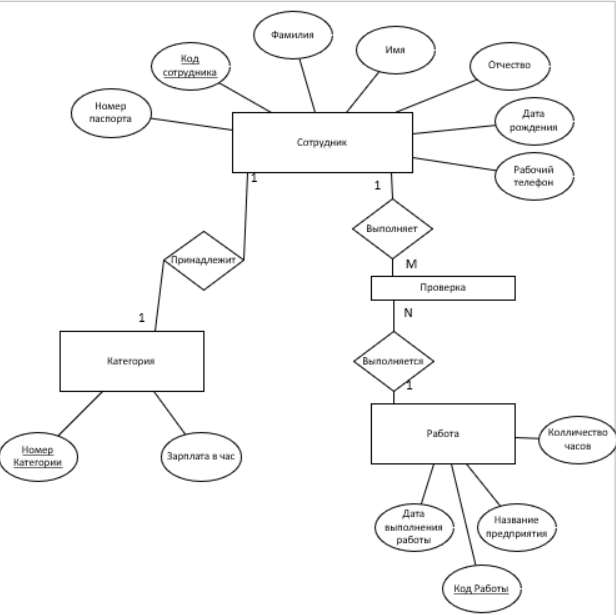


Рис.3.Итоговая ER-модель.

## 2.2. Логическое проектирование

Логическое проектирование является более техническим и детализированным этапом, где мы применяем конкретные методы и инструменты, такие как нотация IDEF1X, для построения логической модели базы данных. Этот процесс помогает нам определить структуру базы данных, типы данных, ограничения целостности и другие аспекты, необходимые для реализации и использования базы данных в рамках конкретного проекта

Задачей логического проектирования является создание эффективной, надежной и оптимальной базы данных, которая соответствует бизнес-потребностям и обеспечивает эффективное хранение, доступ и обработку данных.

Нотация IDEF1X похожа на итоговый вариант физического проектирования. По завершению этапа на руках уже будет готовая схема со всеми таблицами, атрибутами и связями.

На рисунке 4 вы можете наблюдать пример нотации IDEF1X:

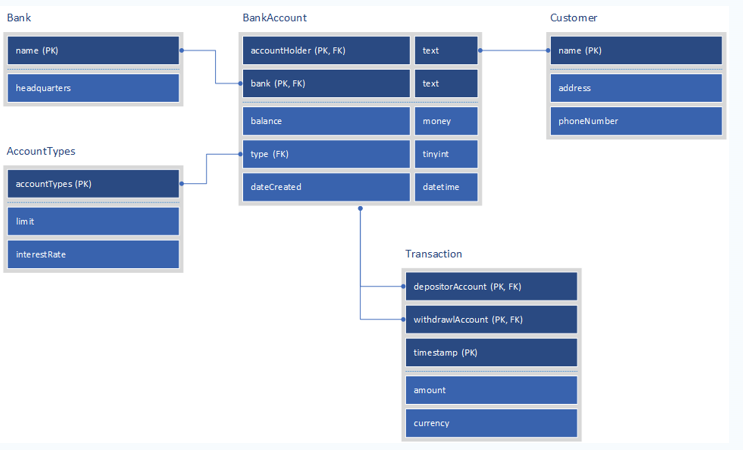


Рис.4.Пример нотации IDEF1X

Логическое проектирование можно разбиить на следующие шаги:

1)Идентификация основных сущностей в предметной области, которые должны быть включены в логическую модель. Сущности представляют конкретные объекты или понятия, о которых необходимо хранить информацию. В данном случае выходит три сущности: Сотрудники, Категории и Работа.

2)Определение атрибутов: для каждой сущности необходимо определить соответствующие атрибуты, которые описывают характеристики или свойства этой сущности. Например, для сущности "Клиент" могут быть определены атрибуты, такие как "Имя", "Фамилия", "Адрес" и т. д. Все атрибуты были чётко обозначены в условии и уже отображены в концептуальном проектировании. В логическое проектирование оно переезжает без изменений.

3)Определение связей: затем необходимо определить связи между сущностями. Связи указывают на связи и зависимости между различными сущностями в предметной области. Например, связь между сущностями "Клиент" и "Заказ" может указывать на то, что каждый клиент может иметь несколько заказов. В данном случае имеется две связи: Один к одному между сотрудником и категорией. И многие к многим между сотрудником и работой. Данная связь выполнена с помощью ассоциативной таблицы «Проверка».

4)Определение ограничений целостности: далее необходимо определить ограничения целостности для модели данных. Ограничения целостности гарантируют правильность и непротиворечивость данных в базе данных. Например, ограничение уникальности может требовать, чтобы каждый клиент имел уникальный идентификатор. В данном случае были созданы уникальные поля с идентификатором в каждой из сущностей.

На рисунке 5 вы можете видеть итог логического проектирования по нотации IDEF1X:

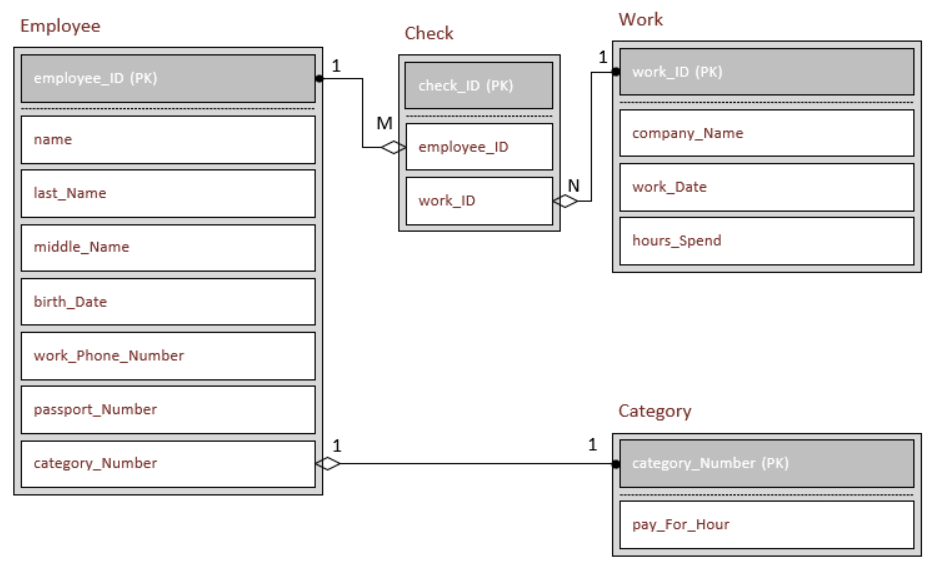


Рис.5.Итог логического проектирования

## 2.3. Физическое проектирование

Физическое проектирование — создание схемы базы данных для конкретной СУБД (Система управления баз данных).

Существует большое количество СУБД, но выбор был остановлен на Postgres, как на одной из самых простых для ознакомления. Вдобавок Postgres имеет весьма простой скриптовый язык, с помощью которого можно легко взаимодействовать с Базой данных.

Для более удобного использования Postgre будет использоваться визуальная оболочка DBeaver (В дальнейшем DB). DB показывает базу данных более наглядно, и позволяет взаимодействовать с БД без скриптового языка.

На рисунке 6 вы можете наблюдать интерфейс DBeaver и открытый PostgreSQL скрипт:

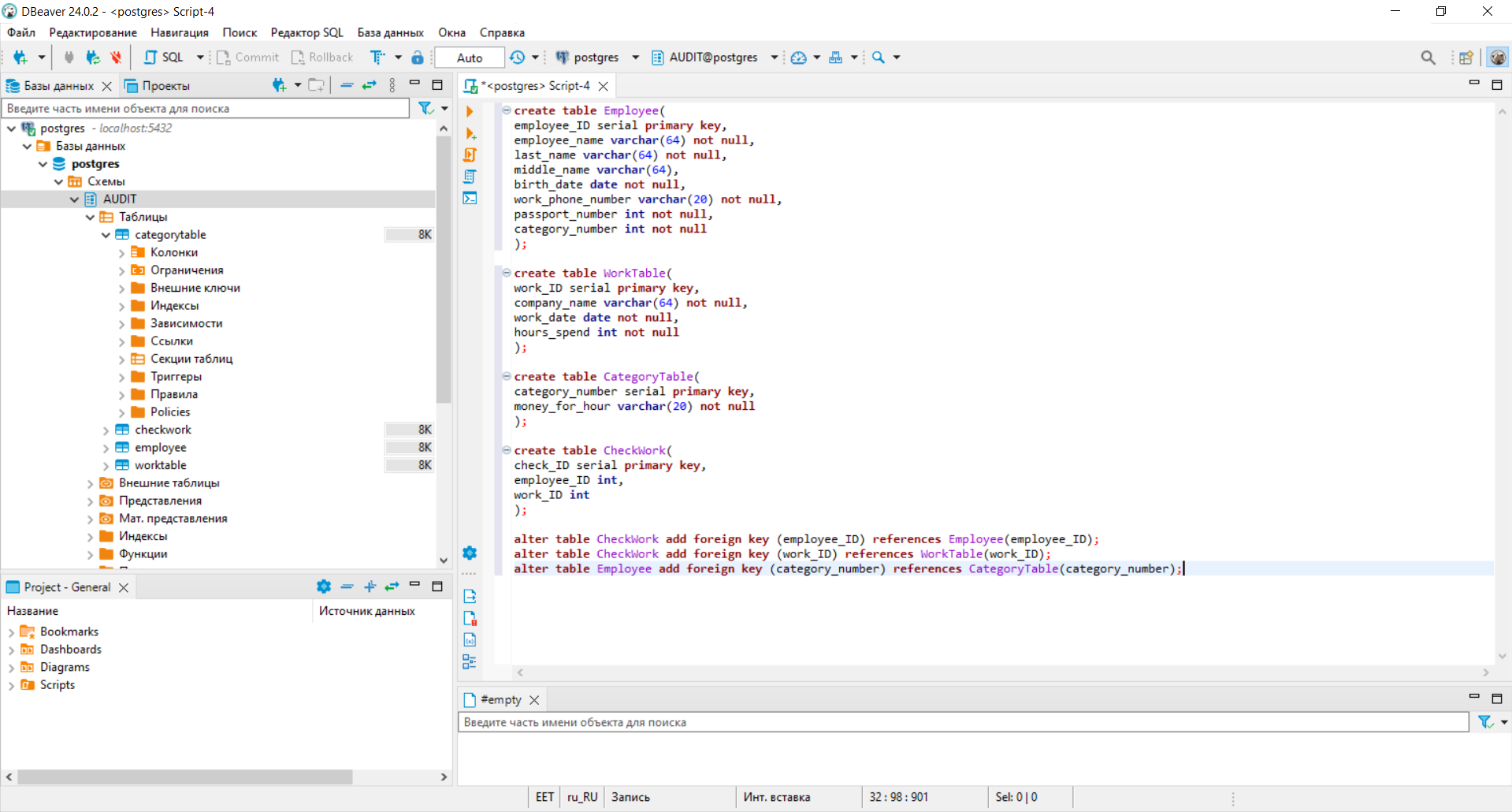


Рис.6.Интерфейс DBeaver

Физическое проектирование выполняется весьма просто. Основная задача — это перенести логическое проектирование на скрипт Postgres и сделать нотацию интерактивной. Важно понимать, что при логическом и физическом проектировании, названия должны быть на английском. От чего таблица «Сотрудник» становится «Employee», таблица «Проверка» превращается в «Checkwork» и т.д.

В скрипте всё выполняется посредством команд, что в конце концов даёт полную БД.

На рисунке 7 вы можете наблюдать скрипт Postgres, в котором создаётся таблица «Checkwork» и связь с таблицей «Employee»:

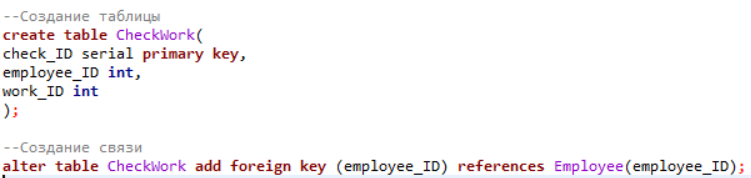


Рис.7.Скрипт Postgres

На рисунке 8 вы можете наблюдать итоговую диаграмму физического проектирования:

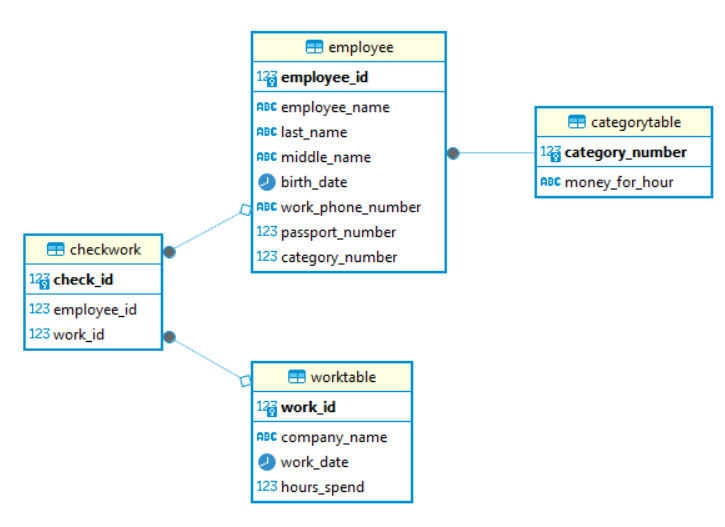


Рис.8.Итоговая диаграмма физического проектирования.

# 3. Разработка API

API является неким интерфейсом, с помощью которого одна программа может взаимодействовать с другой. Например, программа на одном языке не может взаимодействовать напрямую с программой на другом языке, но с помощью API это возможно.

Для написания API был выбран язык PHP, так как он имеет много возможностей, и является весьма простым и базовым для работы с API. Для открытия файлов PHP будет использоваться OpenServer, а сам код будет писаться в текстовом редакторе Visual Studio Code (в дальнейшем VSCode). Данный редактор был выбран так как он поддерживает синтаксис многих языков программирования, а также имеет множество расширений, позволяющих упростить и ускорить работу.

На рисунке 9 вы можете наблюдать интерфейс VSCode с открытым API скриптом:

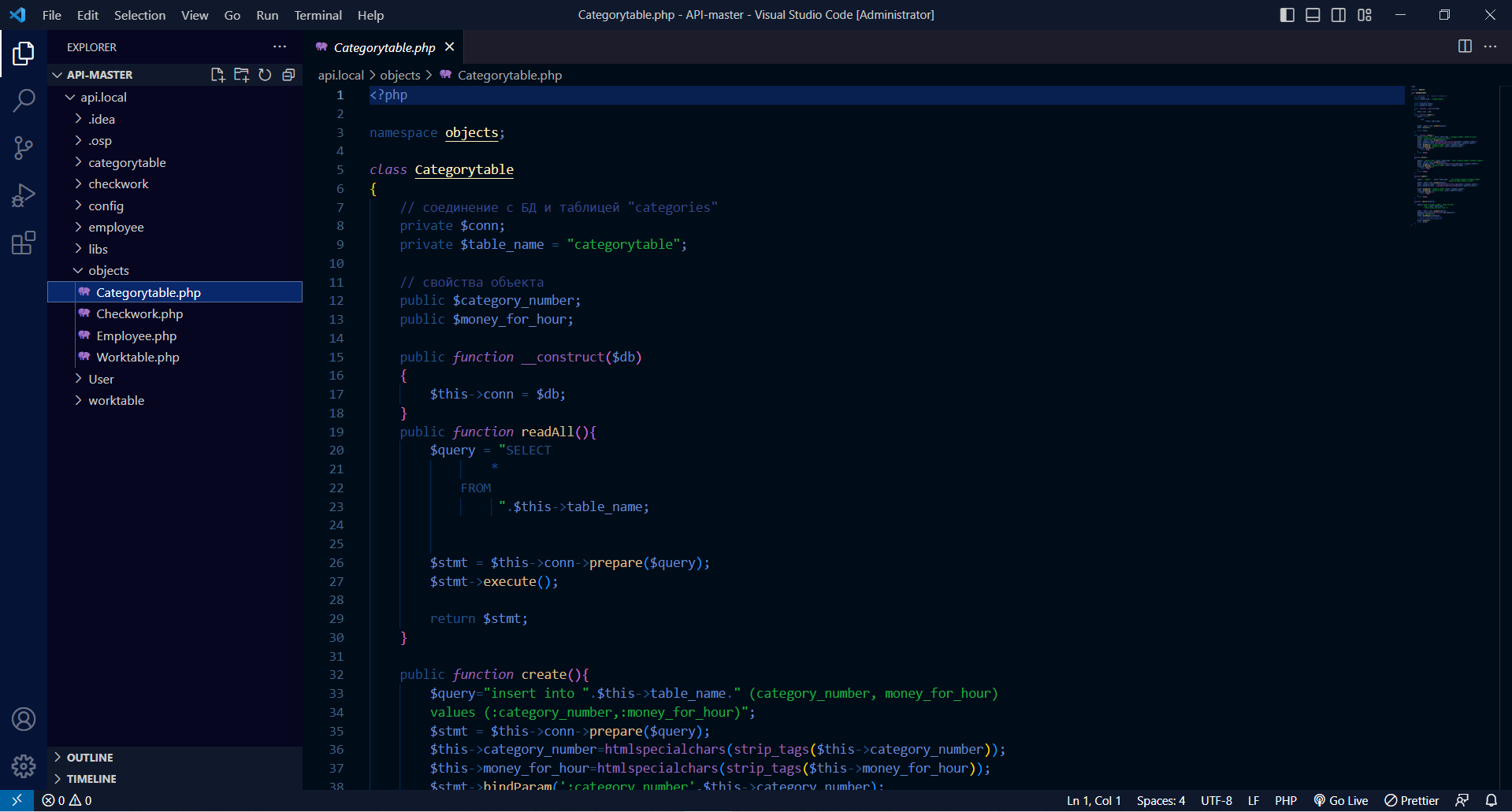


Рис.9.Интерфейс VSCode с открытым скриптом “Categorytable”

Структура API состоит из:

1)Технических файлов, служащих для подключение API к OpenServer и нашей БД (.osp, config)

2)Файлы для подключения таблиц из БД, все они хранятся в папке objects а их название совпадает с названием таблиц. (Catogurytable, Checkwork, Employee, Worktable)

3)Файлы, отвечающие за взаимодействие с таблицей. Добавление, удаление, редактирование или поиск записей. У каждой таблицы файлы свои, так как название и количество колонок везде отличаются.

На рисунке 10 вы можете наблюдать папку с функциональными файлами для таблицы Checkwork:

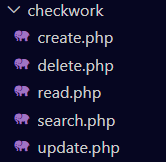


Рис.10.Функциональные файлы для таблицы Checkwork

## 3.1. Подключение базы данных к API

Подключение БД к API происходит в папке config. В ней есть два файла core.php и database.php. Первый это ядро, которое остаётся без изменений. Второй это файл подключения БД. В нём на нужно указать:

1) Хост. Обычно этим значением будет localhost если работать локально на компьютере, но в данном случае был задействован настроенный сервер, а потому будет указан его IP

2)Название БД и имя пользователя. В данном случае оба значения совпадают.

3)Пароль от пользователя, чтобы API смог взаимодействовать с БД и менять значения в таблицах.

На рисунке 11 вы можете наблюдать скрипт подключения БД к API в файле database.php:

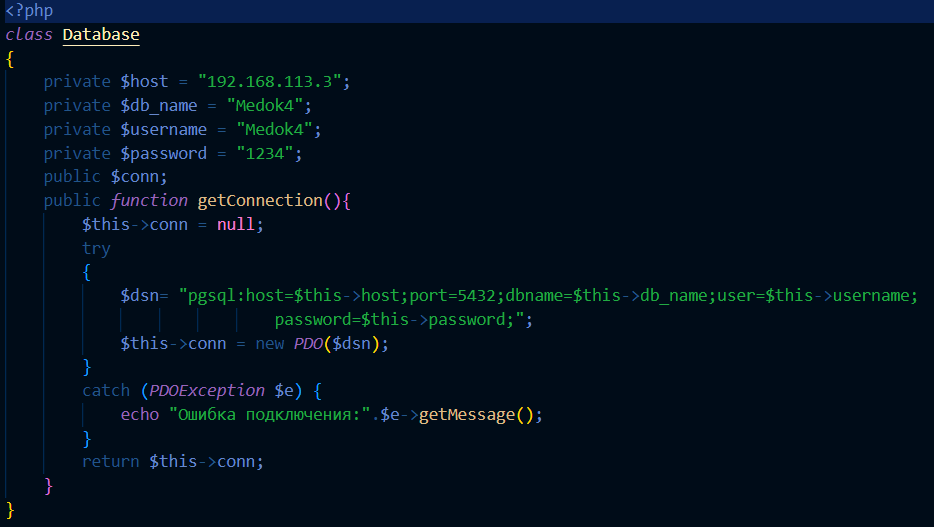


Рис.11.Скрипт подключения БД к API

## 3.2. Подключение таблиц

В папке objects находится несколько файлов. В данном случае их четыре, по одному на каждую таблицу. Структурно все они одинаковые. Отличается лишь название таблиц, количество столбцом и их имена, а также функция search, для каждой таблицы она будет индивидуальна.

В файле с подключением находится:

1)Само подключение, где описываются все колонки и название самой таблицы.

2)Функции, описанные выше, которые в дальнейшем будут взаимодействовать с БД напрямую. (create, delete, update, search, read). В каждой из них даётся список столбцов, которые будут использоваться тем или иным образом. (Например: В функции delete даётся только id таблицы, так как запись можно удалить использую только один этот столбец, ведь он уникальный и не содержит повторений)

На рисунке 12 вы можете наблюдать подключение и функцию read в файле Checkwork.php:



Рис.12.Подключение таблицы checkwork и функция read.

На рисунке 13 вы можете наблюдать функцию create и delete:



Рис.13.Функции create и delete в файле Checkwork.php

На рисунке 14 вы можете наблюдать функции update и search в файле Checkwork.php:



Рис.14.Функции update и search в файле Checkwork.php

## 3.3. Функции API

Функции API предназначены для взаимодействия с базой данных. В данном случае их будет всего 5, но при необходимости их можно написать больше:

1)create (создать) – позволяет создать в таблице внутри БД новую запись.

2)delete (удалить) – позволяет удалить запись в таблице.

3)update (обновить) – позволяет отредактировать уже существующую запись внутри таблицы

4)search (поиск) – позволяет пользователю искать конкретные записи внутри таблицы.

5)read (читать) – показывает всю таблицу из БД целиком. Эта функция единственная, которую можно тестировать на Openserver напрямую, без использования Postman, ведь она не изменяет базу данных, а лишь показывает её содержимое.

Во всех функциях используются одни и те же переменные, совпадающие с названием таблиц и их столбцов, однако каждая таблица должна иметь собственный набор одних и тех же функций. В данном случае выходит 4 файла самих таблиц, и 5 файлов функций для каждой из них, что даёт 20 файлов в сумме. Все они похожи и различаются лишь названиями переменных.

На рисунке 15 вы можете наблюдать функцию create.php для таблицы checkwork:



Рис.15.Функция create.php для таблицы checkwork

На рисунке 16 вы можете наблюдать функцию delete.php для таблицы checkwork:

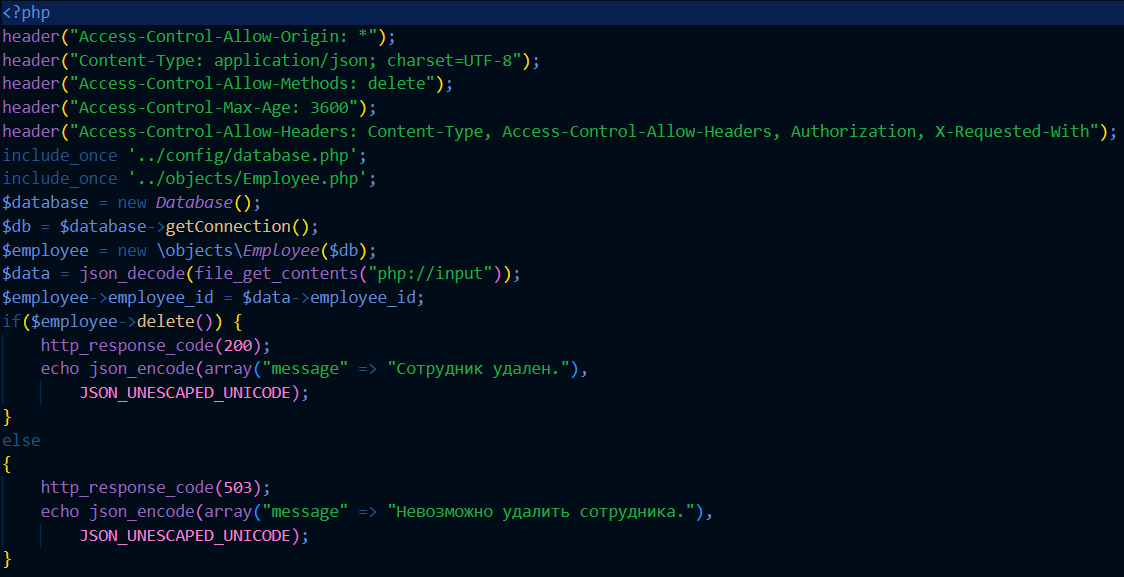


Рис.16.Функция delete.php для таблицы checkwork

На рисунке 17 вы можете наблюдать функцию read.php для таблицы checkwork:



Рис.17.Функцию read.php для таблицы checkwork

На рисунке 18 вы можете наблюдать функцию update.php для таблицы checkwork:

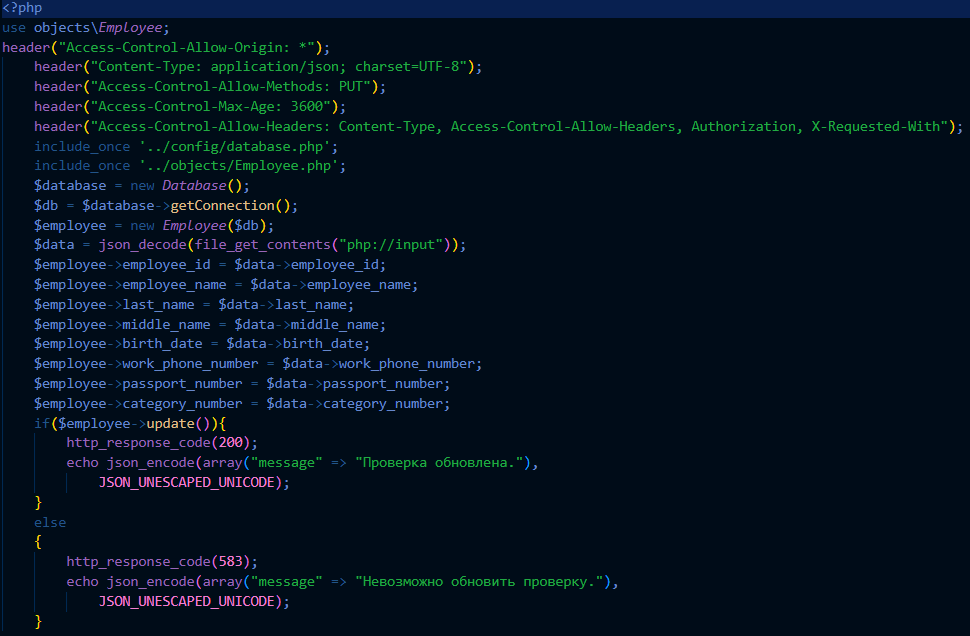


Рис.18.Функция update.php для функции checkwork.

На рисунке 19 вы можете наблюдать функцию search.php для таблицы checkwork:



Рис.19.Функция search.php для функции checkwork

## 3.4. Открытие API на Open server

Open Server — это портативный локальный WAMP/WNMP сервер, имеющий многофункциональную управляющую программу и большой выбор подключаемых компонентов.

Алгоритм работы с Open Server в данном проекте выглядит следующим образом:

1)Установка Open Server. Сам установщик скачивается с официального сайта ospanel.io, и устанавливается в несколько кликов. В дополнение в самом установщике нужно выбрать модули PHP.

2)В корневой папке Open Server в папку home вставляется написанный API-проект.

3)Через значок Open Server в области уведомлений активируется API и открывается в браузере.

На рисунке 20 вы можете наблюдать открытое в браузере API с помощью Open Server:

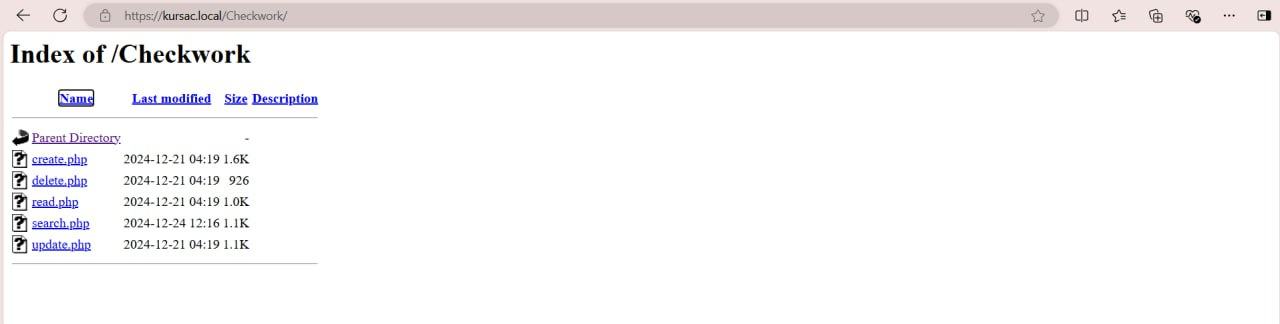


Рис.20.API открытое с помощью Open Server

# 4. Тестирование API с помощью Postman

Postman — приложение для работы с API, которое позволяет создавать, тестировать и документировать API с помощью конструкторов запросов, визуализации ответов и автоматизации проверки.

Простыми словами, Postman позволяет протестировать написанный API с наглядной демонстрацией итогов каждого теста. Сам Postman очень удобен в использовании и довольно прост в освоении.

В Postman будут протестированы все функции кроме read, т. к. эта функция отвечает за показ всей таблицы, и она спокойно открывается в браузере с помощью Openserver.

На рисунке 21 вы можете наблюдать интерфейс программы Postman c открытым тестом функции create для таблицы checkwork:

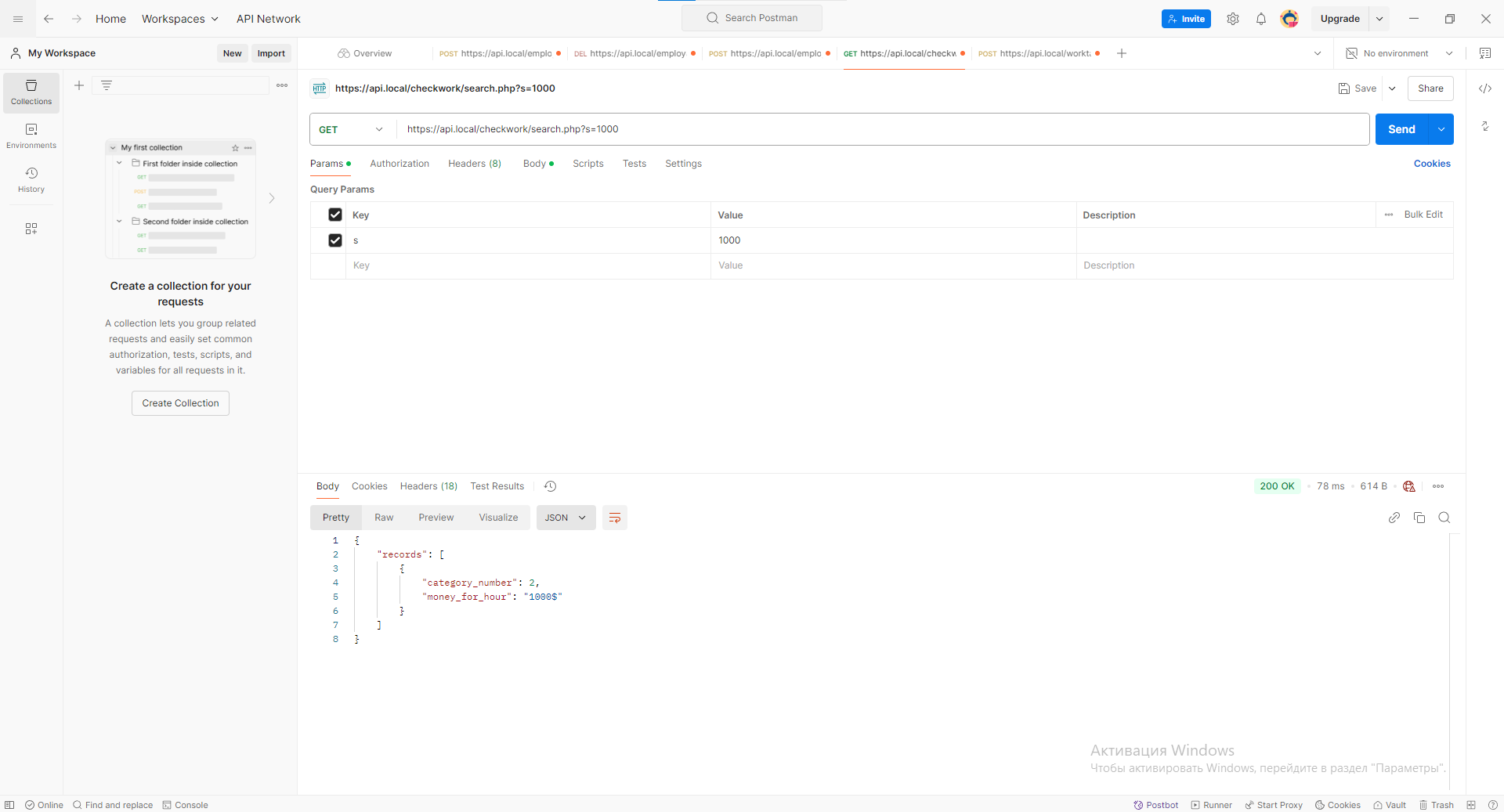


Рис.21.Интерфейс программы Postman c открытым тестом search

Для взаимодействия с БД, программе Postman необходима ссылка на файл с функцией. Ссылка берётся из браузера, когда API открыто через Open Server. Указав нужную ссылку, нужно выбрать метод взаимодействия. В следующих тестах это метод get для search, post для create и update, delete для одноимённой функции.

Чтобы указать условия теста, в меню под ссылкой выбирается пункт Body, а после подпункт raw, и уже в нём прописываются условия теста. Исключением в данном случае является функция search, ведь для её работы необходимо остаться на первой вкладке params, и указывать ключ что был описан в скриптах, в данном случае ключом будет «s». Также нудно указать значение, которое будет искаться тестом. Значение что нужно найти также было описано в скрипте. В данном случае это будет уникальное значение таблицы «id». Важно понимать, что если вписывать другое значение, то Postman ничего не найдёт, ведь он будет пытаться искать значение в id, и даже если оно есть в самой записи, Postman его не заметит.

Подытожив всё выше сказанное, можно составить общий алгоритм для всех тестов:

1)Вставить ссылку на необходимую функцию из API, и выбрать необходимый метод.

2)Перейти в подкладку raw в вкладке body, игнорировать в случае с тестом search.

3)Составить условия теста. Описать запись, которую нужно создать или редактировать. Указать id записи, которую необходимо удалить или указать значение для поиска записи.

4)Нажать на кнопку send и получить результат.

## 4.1. Тестирование функции create

Функция create должна создавать запись в таблице. Для этого необходимо в условиях теста описать каждый столбец, который будет создан в новой записи. Важно понимать, что все колонки что будут описаны в условиях уже были указаны в скрипте. И если в условии количество колонок не совпадает с количеством в скрипте, то тест выдаст ошибку. Поэтому необходимо заполнять каждое поле.

Далее будет показан процесс теста функции create. Этот тест и все последующие будут показаны на примере таблицы checkwork.

На рисунке 22 вы можете наблюдать пустую таблицу checkwork через интерфейс программы dBeaver:

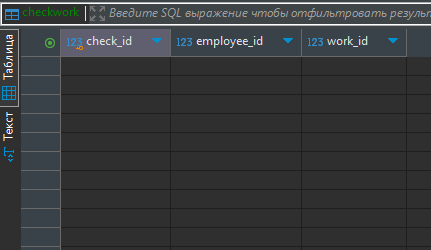


Рис.22.Пустая таблица checkwork, просмотренная через dbeaver

На рисунке 23 вы можете наблюдать условия теста create:

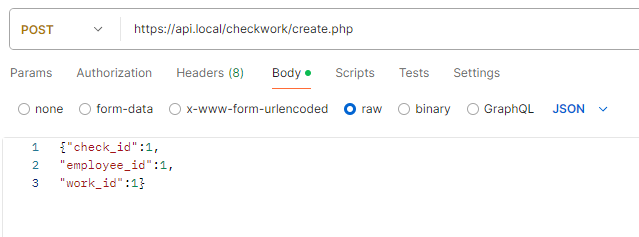


Рис.23.Условия теста create

В скрипте каждого из условий было указано выводить сообщение в случае успеха, или ошибку в случае неудачного выполнения теста. Так как в данном случае всё правильно, Postman выводит сообщение с успехом.

На рисунке 24 вы можете наблюдать сообщение об успешном окончании теста:

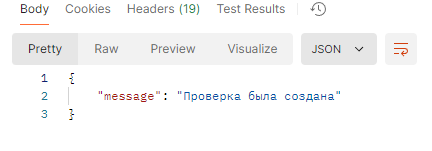


Рис.24.Сообщение об успешном окончании теста.

Теперь если открыть таблицу в dbeaver, можно заметить, что в ней появилась запись, где все колонки совпадают с условиями, указанными в тесте.

На рисунке 25 вы можете заметить созданную тестом запись в таблице:

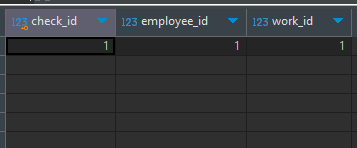


Рис.25.Созданная тестом запись в таблице

## 4.2. Тестирование функции read

Для тестирования функции read, postman не нужен. Так как данная функция не изменяет данные в базе данных, а лишь просматривает отдельные таблицы, то её спокойно можно открыть в браузере через файл read.php

В данном случае открыв функцию для таблицы checkwork, будет открыта вся таблица, и можно будет заметить созданную в прошлом тесте запись.

На рисунке 26 вы можете наблюдать таблицу checkwork, просмотренную через функцию read:

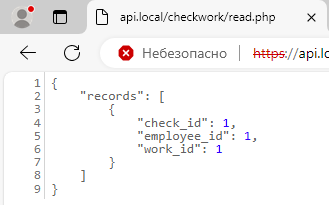


Рис.26.Таблица checkwork, открытая через функцию read

## 4.3. Тестирование функции update

Функция update схожа с create в плане заполнения условий, нужно также указать все столбцы в таблицы. Однако данная функция не добавляет запись, а редактирует уже существующую. В данном случае будет изменён параметр work\_id с 1 на 2, чтобы удостоверится что функция работает.

На рисунке 27 вы можете наблюдать условия теста update:

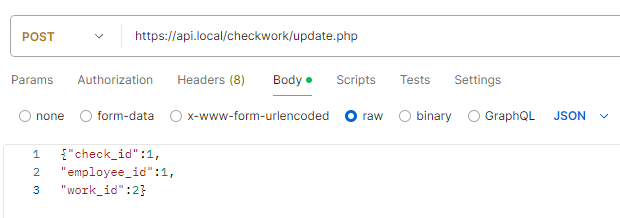


Рис.27.Условия теста update

На рисунке 28 вы можете наблюдать сообщение об успешном окончании теста update:

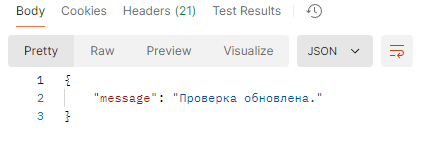


Рис.28.Сообщение об успешном окончании теста

Как итог, открыв таблицу после теста можно заметить, что параметр и вправду изменился, а значит всё прошло успешно.

На рисунке 29 вы можете наблюдать таблицу checkwork после теста, открытую в dbeaver:

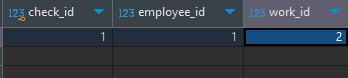


Рис.29.Таблица checkwork после теста update

## 4.4. Тестирование функции delete

Функция delete удаляет запись из таблицы исходя из уникального столбца указанного условия. В скрипте был указан только id самой таблицы, а потому и указывать в условиях нужно именно его.

На рисунке 30 вы можете наблюдать условия для теста delete:

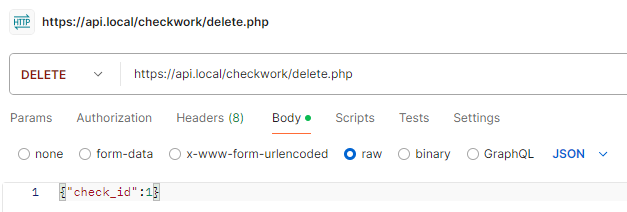


Рис.30.Условия теста delete

На рисунке 31 вы можете наблюдать сообщение об успешном окончании теста delete:

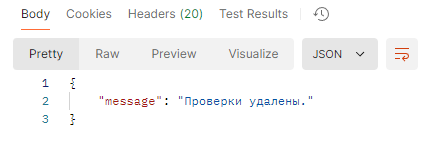


Рис.31.Сообщение об успешном окончании теста

На рисунке 32 вы можете наблюдать таблицу checkwork после выполнения теста delete:

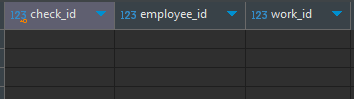


Рис.32.Таблица checkwork после теста delete

## 4.5. Тестирование функции search

Функция search является уникальной, ведь её можно настроить таким образом что выводить именно те данные что необходимы пользователю. В данном случае будут производится поиски по уникальному полю таблицы.

Для начала следует создать несколько записей, чтобы было что искать в таблице. С помощью create создаётся 5 записей с уникальными значениями.

На рисунке 33 вы можете наблюдать таблицу checkwork, предварительно подготовленную к тесту search:

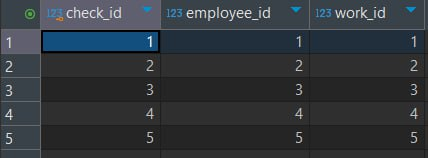


Рис.33. Таблица checkwork, заполненная уникальными записями

В данном тестировании будет искаться запись с check\_id равное 3. Для этого в Postman нужно вставить ссылку на функцию search и указать необходимое значение вместе с ключом, что был указан в скрипте.

На рисунке 34 вы можете наблюдать условие теста search:

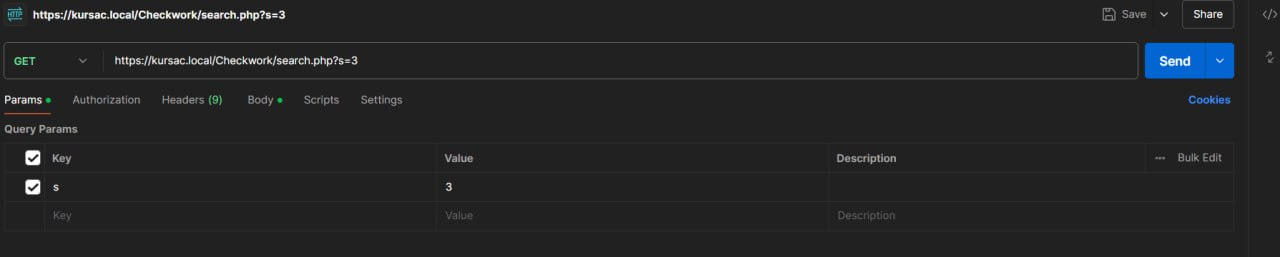


Рис.34.Условия теста search

На рисунке 35 вы можете наблюдать результат теста search:

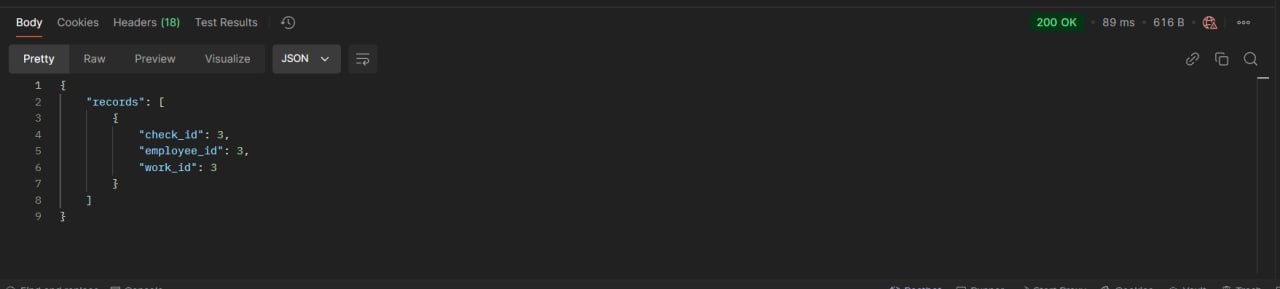


Рис.35.Результат теста search

Поскольку search является уникальной функцией, она не выводит сообщение об успешной проведении теста, она сразу выводит результат.

# Заключение

Получившееся база данных должна помочь рассчитывать зарплату сотрудников аудит фирмы благодаря удобному интерфейсу и грамотному расположению таблиц и их связей. Вдобавок функции всегда можно подредактировать под нужды фирмы, что добавляет базе гибкости и позволяет расширить её функционал по мере необходимости. API функционирует и проверки через Postman прошли успешно.

В заключении курсовой работы можно сделать следующие выводы:

1)Был изучен большой объём информации касательно разработки базы данных, разработки и тестирования API.

2)Были изучены документации по Разработке Базы данных, API, а также их тестированию.

2)Была разработана база данных в соответствии с требованиями на трёх уровнях проектирования: концептуальный, логический и физический.

3)Было разработано API с подключением всех таблиц и созданием функции для каждой из таблиц.

4)Было протестировано API с помощью программы Postman.

5)Были внесены правки в дизайн, чтобы соответствовать техническому заданию.

В ходе работы над курсовой работой были получены навыки сбора и обработки информации, практические умения в сфере баз данных и работы с ними.

Весь опыт, получены в разработке базы данных и API, пригодиться в будущем для выполнение более крупных и комплексных проектов.

# Список использованных источников

1. Wikipedia статья «База данных» [Электронный ресурс]: [https://ru.wikipedia.org/wiki/База\_данных (дата](https://help.figma.com/hc/en-us/sections/4405269443991-Figma-for-Beginners-tutorial-4-parts-(дата) обращения 19.12.2024)
2. Wikipedia статья «API» [Электронный ресурс]: [https://ru.wikipedia.org/wiki/API (дата](https://ru.wikipedia.org/wiki/Figma(дата) обращения 19.12.2024)
3. Презентация «Инфологическое проектирование баз данных по модели Чена» [Электронный ресурс]: <https://kbkdist.ru/mod/resource/view.php?id=16877> (дата обращения 19.12.2024)
4. Презентация «Логическое проектирование и ограничения целостности построение структуры базы данных в нотации IDEF1X» [Электронный ресурс]: <https://kbkdist.ru/mod/resource/view.php?id=16926> (дата обращения 19.12.2024)
5. Презентация «Физическое проектирование» [Электронный ресурс]: <https://kbkdist.ru/mod/resource/view.php?id=16937> (дата обращения 19.12.2024)
6. Документация Postgresql [Электронный ресурс]: <https://www.postgresql.org/docs/> (дата обращения 19.12.2024)
7. Статья «Open Server — профессиональный инструмент веб-разработчика под Windows» [Электронный ресурс]: <https://habr.com/ru/articles/137388/> (дата обращения 23.12.2024)
8. Документация PHP [Электронный ресурс]: <https://www.php.net/manual/ru/index.php> (дата обращения 23.12.2024)