**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

ПП.09.02.07-1.24.202.20

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

ПК.02 Осуществление программных интеграций

|  |  |  |
| --- | --- | --- |
| Руководитель от предприятия:  М.П. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Л.Г.Стрежнёва) |
| Руководитель от техникума: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (С.Н.Касьяненко) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Д.А. Отчесов) |

Иркутск 2024

**Содержание**

[Введение 3](#_Toc162298899)

[1 Создать проект на GitHub. 4](#_Toc162298900)

[2 Техническое задание 6](#_Toc162298901)

[Приложение А – Техническое задание 7](#_Toc162298902)

[5 Требования к документации 9](#_Toc162298903)

[3 Модуль приема данных извне MySql 12](#_Toc162298904)

[4 Выполнение отладки 13](#_Toc162298905)

[5 Тестовые наборы данных 15](#_Toc162298906)

[6 Настройка PHPStan 17](#_Toc162298907)

[Заключение 19](#_Toc162298908)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 20](#_Toc162298909)

# **Введение**

Производственная практика необходима для получение первичных профессиональных умений и навыков в области организации и администрирования. При этом должна быть достигнута систематизация, обобщение, закрепление и углубление теоретических знаний и умений, приобретенных при освоении основной образовательной программы.

В качестве предприятия по прохождению производственной практики выступает МОУ ИРМО «Карлукская СОШ».

Целью производственной практики являются следующие задачи:

1. Приобрести необходимые навыки самостоятельной работы по анализу и умению работать с данными;
2. Получить практические навыки в дистанционном обслуживании кассовых аппаратов;
3. Обеспечить сбор и обработку необходимого статистического материала для выполнения практических заданий.

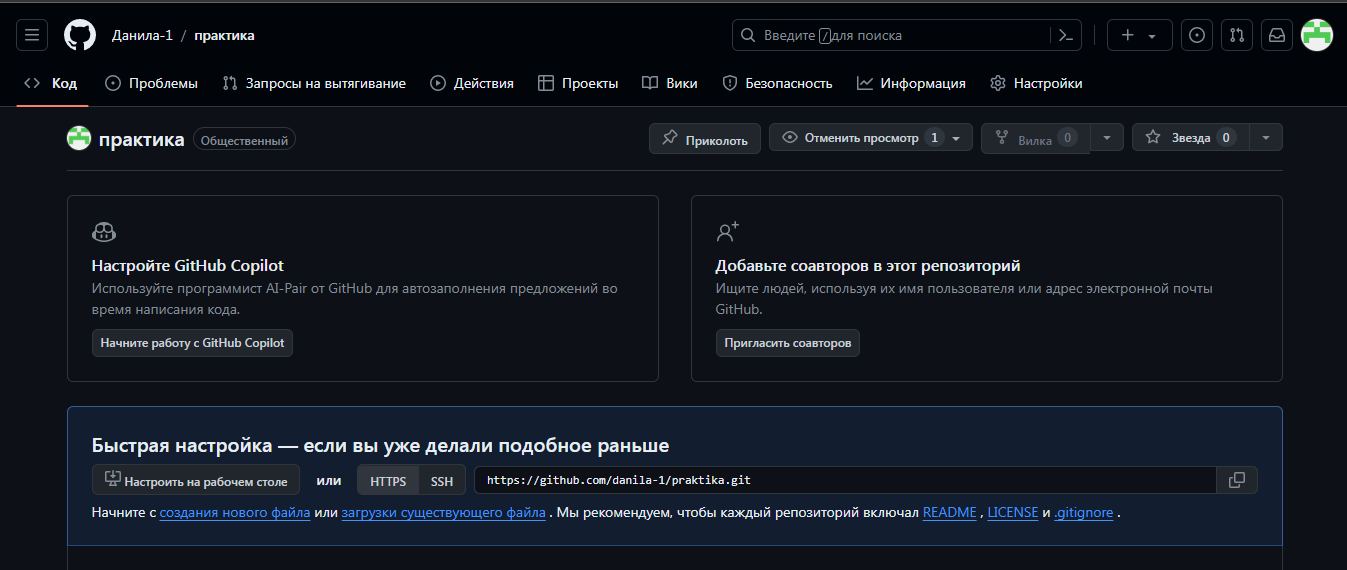
В процессе прохождения практики были усвоены основы работ в коллективе.

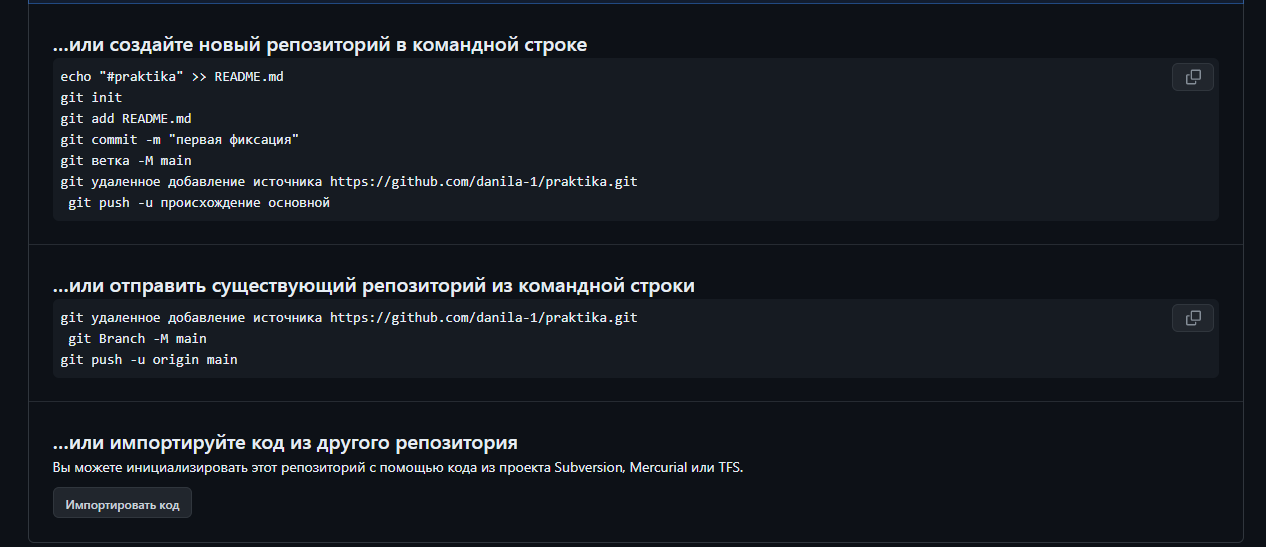
«1С» — российская компания, основанная в 1991 году и занимающаяся разработкой, изданием и поддержкой компьютерных программ, баз данных делового и домашнего назначения, а также компьютерных игр.

1. **Создать проект на GitHub.**

Зарегистрируйтесь на GitHub: Если у вас еще нет учетной записи на GitHub, вам нужно ее создать. Для этого посетите https://github.com и нажмите на кнопку “Sign up” (Зарегистрироваться).

Создайте репозиторий: После входа в систему перейдите в меню “Repositories” (Репозитории) и нажмите кнопку “New” (Новый). Выберите тип репозитория (например, “Public” или “Private”) и введите имя для вашего репозитория.





1. Техническое задание

1. Наименование проекта и цель создания базы данных.

2. Общие требования к базе данных.

3. Описание структуры данных.

4. Требования к целостности данных.

5. Требования к документации.

Приложение А – Техническое задание

**Министерство образования Иркутской области**

Государственное бюджетное профессиональное образовательное учреждение

Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**РАЗРАБОТКА БАЗЫ ДАННЫХ**

|  |  |  |
| --- | --- | --- |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (С.Н.Касьяненко) |
| Руководитель от предприятия:  М.П. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Л.Г.Стрежнёва) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Д.А. Отчесов) |

Иркутск 2024

**1 Наименование проекта и цель создания базы данных**

Наименование проекта: «Планетарий».

Цель создания: повышение интереса в сфере астрономии.

Исполнитель: студент Иркутского авиационного техникума, группы БД-20-2, Отчесов Д.А.

Разработка базы данных проходит в рамках производственной практики для ГБПОУИО «ИАТ».

Сроки разработки базы данных с 14.03.2024 по 04.04.2024 года.

**2 Общие требования к базе данных**

* База данных должна хранить информацию о запчастях, их характеристиках, состоянии и стоимости.
* База данных должна содержать информацию данных клиента (логин, пароль).
* База данных должна иметь возможность поиска по названию.
* База данных должна поддерживать целостность данных и обеспечивать безопасность доступа.
* База данных должна обеспечивать возможность добавления новых запчастей, а также изменения информации о существующих.

1. **Описание структуры данных**

База данных «Планетарий» может состоять из следующих таблиц:

* + Пользователи. (Код, Имя, Почта, Пароль, Номер телефона).
  + Билеты. (Код, Название сеанса, Код сеанса, Код пользователя).
  + Сеансы. (Код, Название, Время начала, Время окончания, Дата).

То есть таблицы: «tickets», «users», «session», «halls» которые имеют связь по внешнему ключу между собой.

1. **Требования к целостности данных**

Требования к целостности данных таковы:

* + 1. Уникальность данных: каждая запись в базе данных должна быть уникальной, чтобы избежать дублирования информации.
    2. Согласованность данных: данные в базе данных должны быть согласованными и точными, чтобы избежать противоречий и ошибок.
    3. Ограничения целостности: использование ограничений целостности, таких как уникальные ключи, внешние ключи и проверочные ограничения, для обеспечения правильности данных.
    4. Актуальность данных: данные должны быть актуальными и обновленными, чтобы предоставлять точную информацию пользователям.
    5. Безопасность данных: обеспечение защиты данных от несанкционированного доступа, изменений и удалений.
    6. Резервное копирование данных: регулярное создание резервных копий данных для предотвращения потери информации в случае сбоев или аварий.

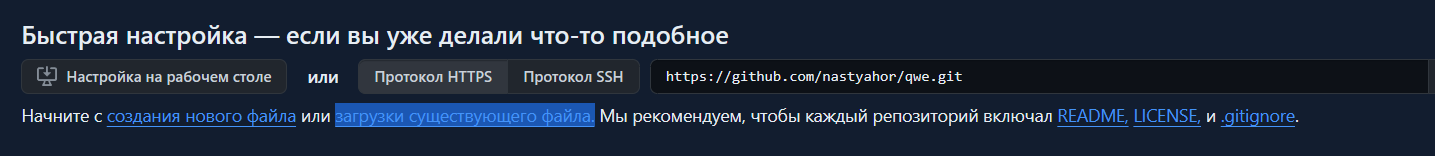
**5 Требования к документации**

Основным документом, регламентирующими использование базы данных является руководство пользователя.

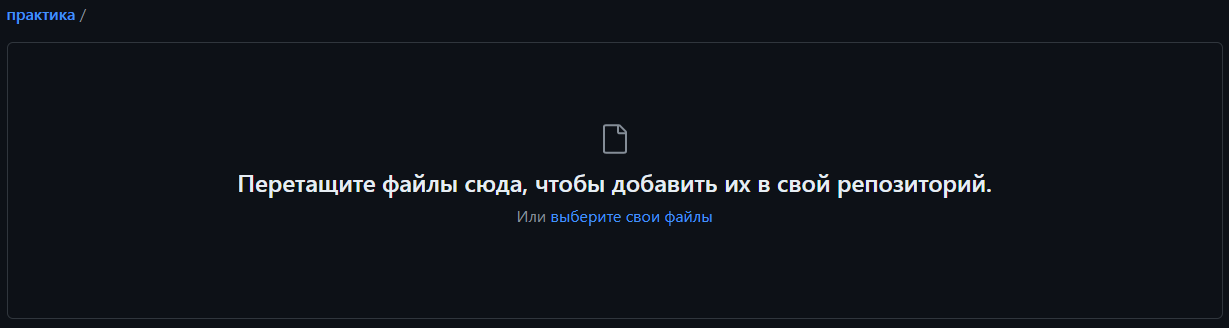
Основным документом, регламентирующими разработку базы данных является техническое задание.

**2.1 Загрузка файла технического задания**

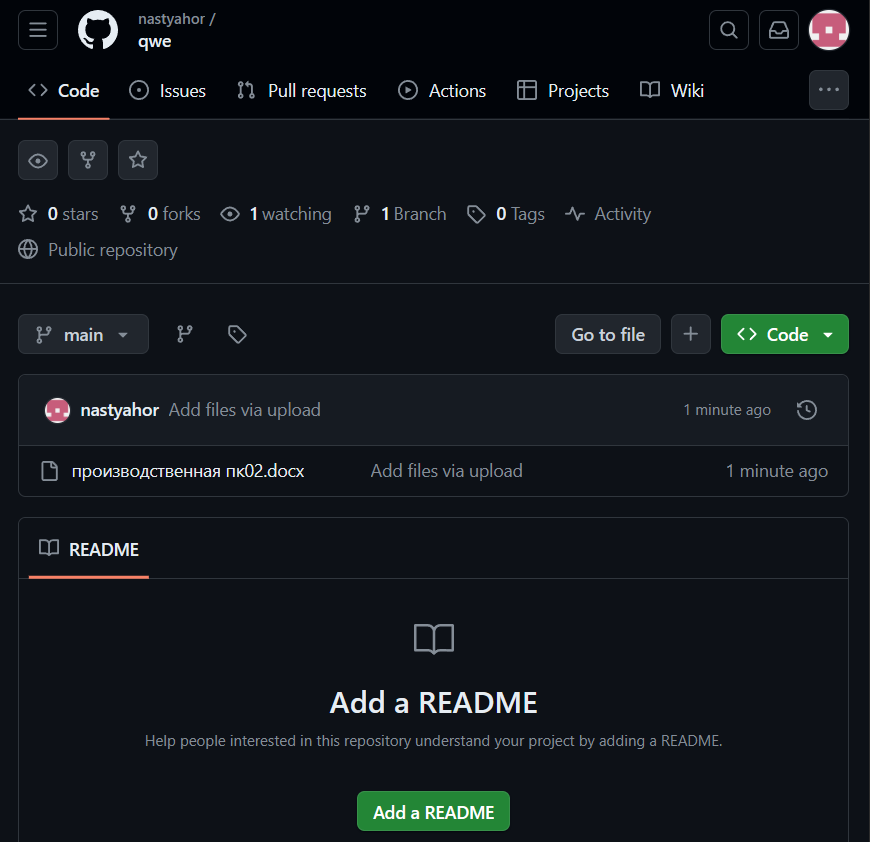
Для загрузки файла на GitHub необходимо зайти в созданный проект. Далее нажать на ссылку «[загрузки существующего файла.](https://github.com/nastyahor/qwe/upload)».



Далее помещаем файл в предоставленной окно.



Нажимаем на кнопку «Commit changes» и готово.

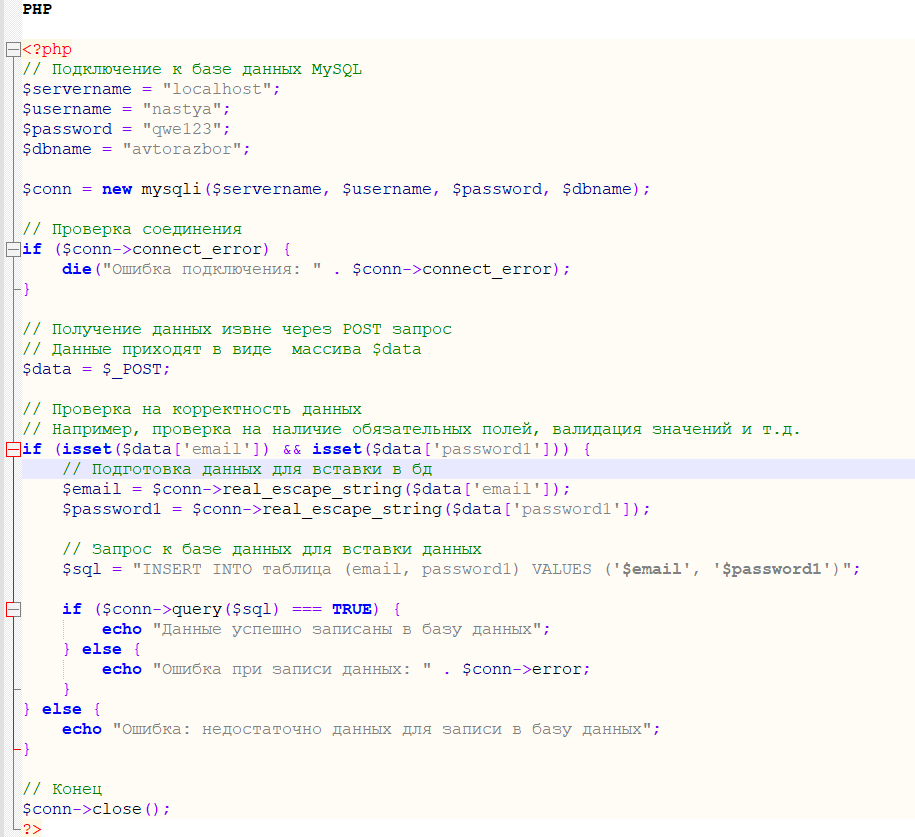


https://github.com/nastyahor/qwe

# **3 Модуль приема данных извне MySql**

Для создания модуля на языке PHP, который принимает данные извне, проверяет их на корректность, а затем записывает их в базу данных MySQL, вам понадобится использовать PHP для обработки данных и выполнения запросов к базе данных.

Вот примерный код, который может быть использован для создания такого модуля:

****

Этот код демонстрирует базовый пример модуля на PHP для приема данных извне, их проверки и записи в базу данных MySQL.

# **4 Выполнение отладки**

Отладка программного кода - это процесс поиска и устранения ошибок, которые могут привести к неправильной работе программы. Во время разработки программы или при ее исполнении могут возникать различные проблемы, такие как неправильное поведение, неверные результаты или даже сбои в работе программы.

Для отладки была использована команда  $errors, с помощью которой пользователь видит сообщение о том, что какие-то данные были введены некорректно. Например:

{

    $errors = [];

    validateEmail($updateProfileFormData, $errors);//& - передача по ссылке. Если бы его не записали, то исходный массив не поменялся бы.

    validateOldPassword($updateProfileFormData, $errors);

    validateNewPassword($updateProfileFormData, $errors);

    validateOldAndNewPasswordsMatch($updateProfileFormData, $errors);

    if (! $updateProfileFormData['email'] === $\_SESSION['user']['email']) {

        if (! isEmailUnique($updateProfileFormData['email'])) {

            $errors[] = 'Ошибка! Пользователь с введённым email уже зарегистрирован';

        }

    }

Или

{

    if ($updateProfileFormData['old\_password'] !== '' && strlen($updateProfileFormData['old\_password']) < 6) {

        $errors[] = 'Ошибка! Длина пароля должна быть не меньше 6 символов';

    }

}

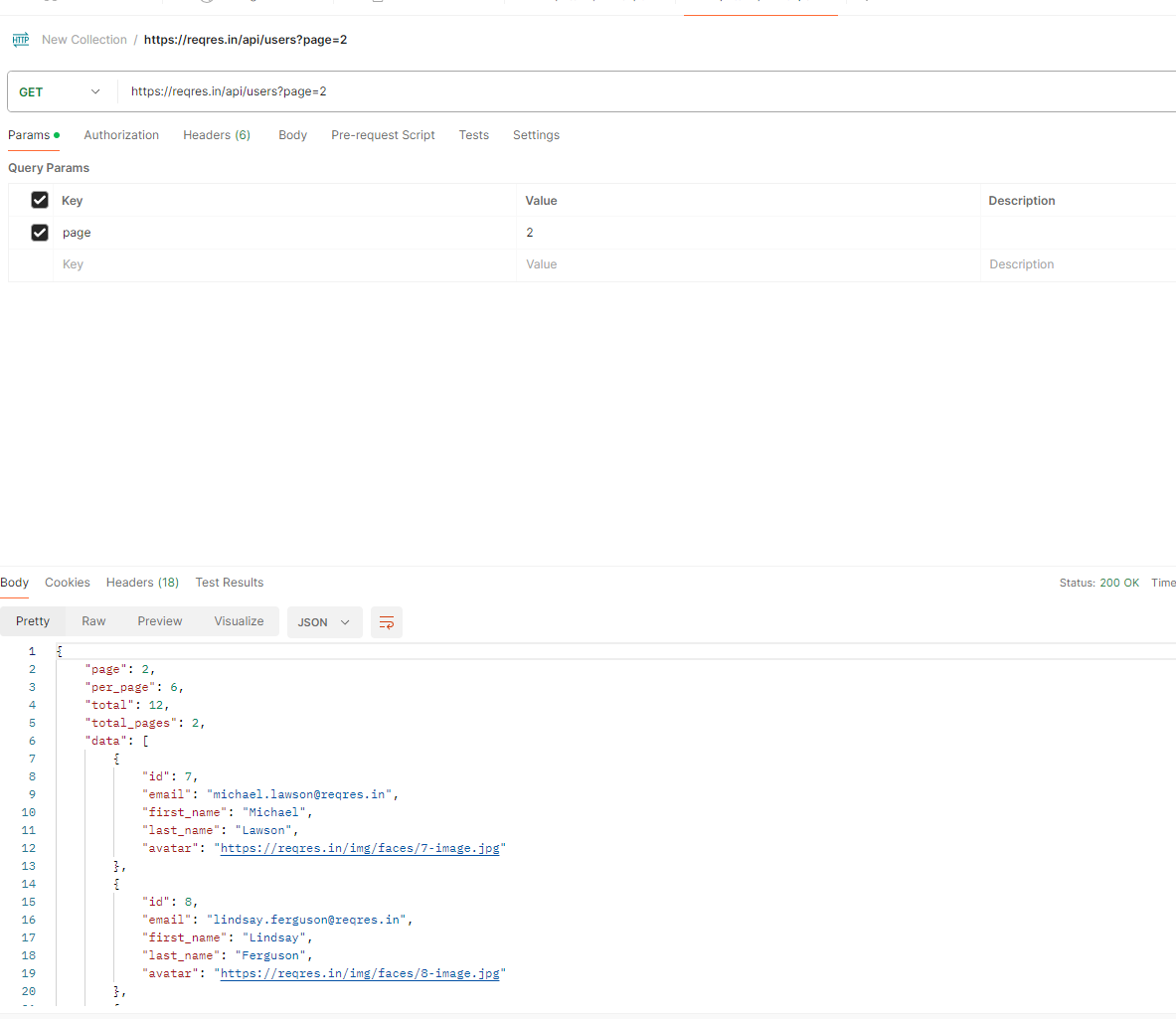
Отладка является важной частью процесса разработки программного обеспечения и помогает создавать качественные и стабильные программы. Она требует тщательности, внимательности и систематичного подхода для успешного нахождения и устранения ошибок в коде.

# **5 Тестовые наборы данных**

Postman — это сервис для создания, тестирования, документирования, публикации и обслуживания API. Он позволяет создавать коллекции запросов к любому API, применять к ним разные окружения, настраивать мок-серверы, писать автотесты на JavaScript, анализировать и визуализировать результаты запросов.

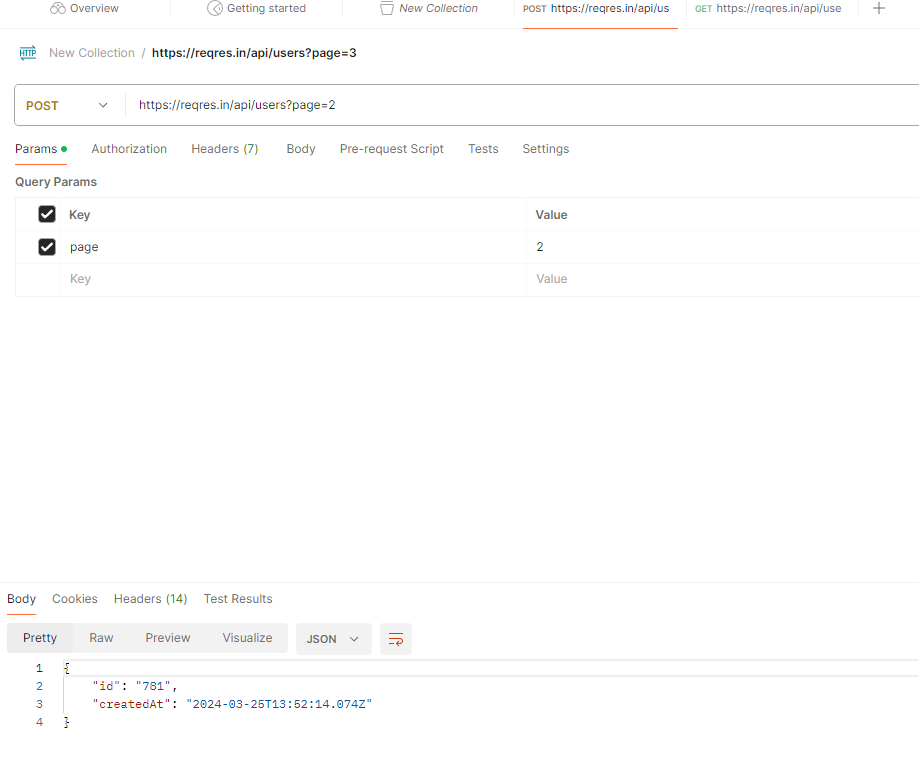
Программа поддерживает разные виды архитектуры API: HTTP, REST, SOAP, GraphQL и WebSockets.

С помощью Postman вы можете создавать и отправлять различные типы запросов (GET, POST, PUT, DELETE и т.д.) к API, указывать параметры запроса, заголовки, тело запроса и многое другое. Postman также позволяет сохранять запросы, создавать коллекции запросов, автоматизировать тестирование API и генерировать документацию.  
 GET - это один из методов HTTP-запроса, который используется для получения данных с сервера. Когда клиент отправляет GET-запрос на сервер, он запрашивает данные и ресурсы с сервера без изменения состояния или данных на сервере. На первом скриншоте представлен результат запроса GET в формате JSON.

****

Для передачи конфиденциальных данных рекомендуется использовать метод POST.

Так же изображен результат запроса POST в формате JSON.

****

Postman широко используется разработчиками программного обеспечения для тестирования и отладки веб-сервисов, а также для взаимодействия с API сторонних сервисов. Этот инструмент облегчает процесс разработки, ускоряет тестирование API и помогает обнаруживать и устранять ошибки в работе веб-сервисов.

# **Настройка PHPStan**

**PHPStan** – фантастический инструмент для поиска ошибок при написании кода. Вам не нужно ничего запускать. Вы можете попробовать онлайн-версию здесь. Для использования **PHPStan** требуется версия 7.1 или более поздней версии и композер. Тем не менее, он способен обнаруживать ошибки из более старой версии. Psalm. **Это** основанный на **PHP** Parser, может найти ошибки и помочь поддерживать разработку более безопасного приложения.

Установка:

composer require --dev phpstan/phpstan

Запуск:

vendor/bin/phpstan analyse [options] [--] [<paths>...]  
example:  
vendor/bin/phpstan analyse -l 5 -c phpstan.neon ./src/

Настройка PHPStan

Все опции запуска PHPStan можно вынести в файл конфигурации phpstan.neon и запускать скрипт следующим образом:

vendor/bin/phpstan analyse -c phpstan.neon

Пример файла конфигурации PHPStan:

parameters:  
 level:5  
 paths:  
 - src

**Уровни анализа (-l | --level):**

Уровень 0: базовые проверки, неизвестные классы, неизвестные функции, неизвестные методы, вызываемые для $this, неправильное количество аргументов, переданных этим методам и функциям, всегда неопределенные переменные.

Уровень 1: возможно, неопределённые переменные, неизвестные магические методы и свойства классов с \_\_call и \_\_get.

Уровень 2: неизвестные методы проверяются по всем выражениям (не только $this), проверка PHPDocs.

Уровень 3: типы возвращаемых значений, типы, присвоенные свойствам.

Уровень 4: базовая проверка мертвого кода - всегда ложные проверки экземпляра и других типов, мертвые ветки else, недостижимый код после возврата; и т. д.

Уровень 5: проверка типов аргументов, передаваемых методам и функциям.

Уровень 6: сообщить об отсутствующих подсказках

Уровень 7: сообщать о частично неправильных типах объединения - если вы вызываете метод, который существует только для некоторых типов в типе объединения, уровень 7 начинает сообщать об этом; другие возможно неправильные ситуации

Уровень 8: методы вызова отчетов и доступ к свойствам для типов, допускающих значение NULL.

Уровень 9: будьте строги в отношении смешанного типа: единственная разрешенная операция, которую вы можете с ним сделать, это передать его другому смешанному типу.

# **Заключение**

В ходе практики по теме "Осуществление программных интеграций" были изучены основные принципы и методы интеграции различных программных систем. Разработаны навыки работы с API, протоколами обмена данными, форматами данных и инструментами для реализации интеграций.  
 Практические задания включали в себя создание и настройку запросов к API, обработку ответов, работу с различными форматами данных (JSON, XML), а также реализацию сценариев интеграции между различными приложениями.  
 В процессе выполнения практических заданий были закреплены навыки:

* Создания проекта на GitHub.
* Составления технического задания на разработку Базы данных в соответствии с ГОСТ.
* Написание модуля приема данных извне
* Выполнение отладки любым из доступных инструментов.
* Подготовка тестовых наборов данных для запросов GET и POST с помощью Postman.
* Настройка PHPStan.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. https://www.youtube.com/watch?v=h3rjNlhDxe8<https://studfile.net/preview/16566619/page:26/>
2. https://backloger.ru/ru/articles/phpstan
3. <https://1c.ru/?ysclid=ltsmbuusqq2933223>
4. https://www.postman.com/downloads/<https://forum.avast.com/index.php?topic=87978.0>
5. <https://habr.com/ru/articles/721340/>
6. https://github.com/