**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

ПП.09.02.07-1.24.202.14

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

ПК.02 Осуществление программных интеграции программных модулей

|  |  |  |
| --- | --- | --- |
| Руководитель от предприятия:  М.П. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (М.В.Баданова) |
| Руководитель от техникума: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (С.Н.Касьяненко) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.В. Манжуев) |

Иркутск 2024

**Содержание**

[Введение 3](#__RefHeading___1)

[2 Техническое задание 5](#__RefHeading___2)

[1 Наименование проекта и цель создания базы данных 5](#__RefHeading___13)

[3 Модуль приема данных извне MySql 8](#__RefHeading___4)

[4 Выполнение отладки 9](#__RefHeading___5)

[5 Тестовые наборы данных 10](#__RefHeading___6)

[6 Настройка PHPStan 12](#__RefHeading___7)

[Заключение 14](#__RefHeading___8)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 15](#__RefHeading___9)

# **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 IT- отдел ООО «Горизонт», специализирующийся на дистанционном управлении и обслуживании кассовых аппаратов.

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 уяснили основы работ в программе 1С, работу 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.**

Создание репозиторий: После входа в систему перейдите в меню Репозитории и нажмите кнопку Новый. Выберите тип репозитория и введите имя для вашего репозитория.

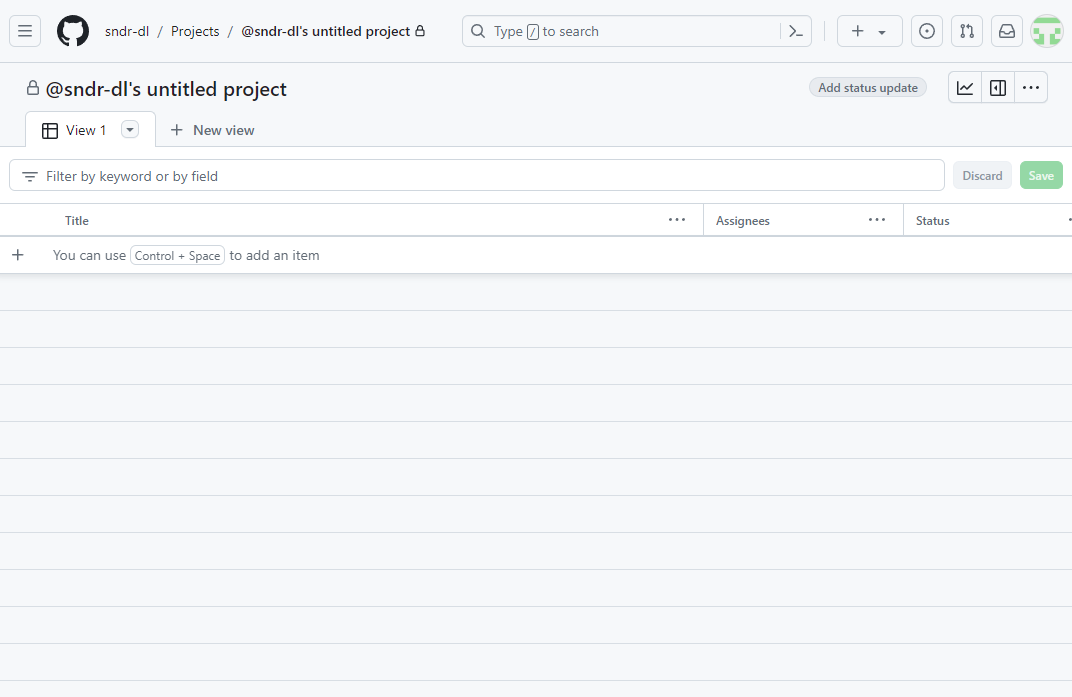


Рисунок 1 – созданный репозиторий

1. Техническое задание

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

Наименование проекта: «Таксопарк».

Цель создания: улучшение работоспособности производства.

Исполнитель: студент Иркутского авиационного техникума, группы БД-20-2, Манжуев А.В.

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 поездках, их маршрутах, водителях, пользователях и стоимости.
* База данных должна содержать информацию данных пользователя (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 поездок, а также изменения информации о существующих.

1. **Описание структуры данных**

База данных «Таксопарк» может состоять из следующих таблиц:

* + Пользователь. (Код, Имя, Пароль).
  + Поездка. (Код, название, Код Пользователя, Код водителя, Место заказа, Место назначения, Цена).
  + Водитель. (Код, Имя, Пароль, Автомобиль).

То есть таблицы: «user», «trip», «driver», 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 удаления.
    6. Резервное копирование данных: регулярное создание резервных копий данных для предотвращения потери информации в случае сбоев или аварий.

**5 Требования к документации**

Основным документам, регламентирующими использование базы данных является руководство пользователя.

Основным документам, 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 сслыку «[загрузки существующего файла.](https://github.com/nastyahor/qwe/upload)»

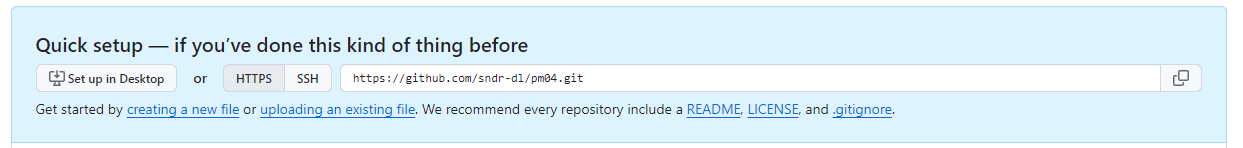


Рисунок 2 – загрузка файла

Далее помещаем файл в предоставленной окно.

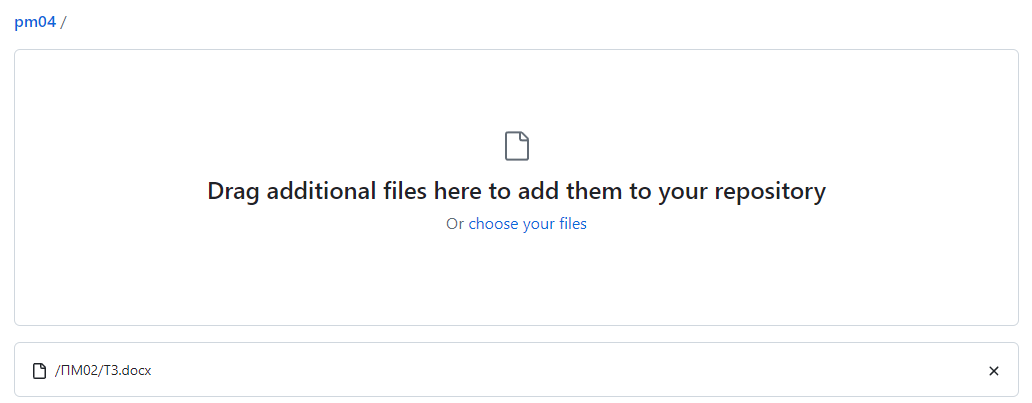


Рисунок 3 – подтверждение загрузки

Нажимаем на кнопку «Commit changes» и готово.

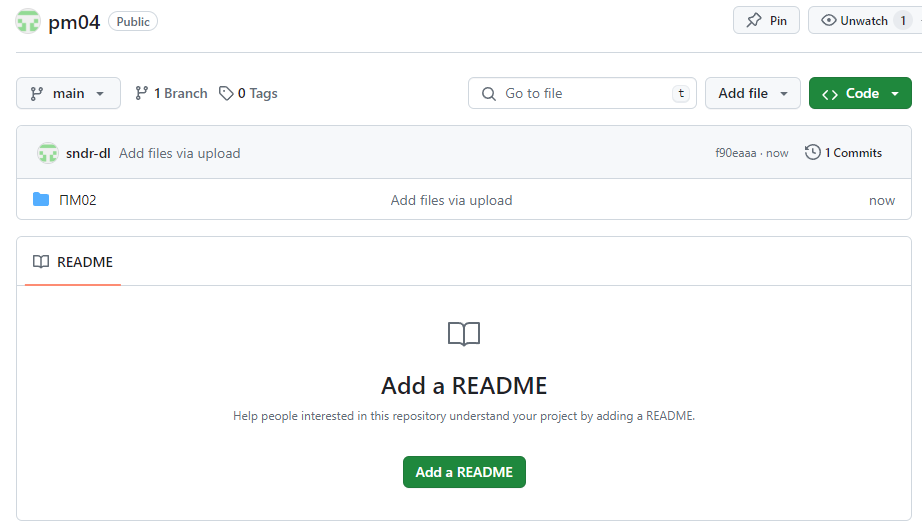


Рисунок 4 – репозиторий с загруженными файлами

https://github.com/sndr-dl/pm04

# **3 Модуль приема данных извне MySql**

Для создания модуля приема данных извне на PHP вам потребуется:

* Создать новый файл с расширением .php.
* Внутри файла написать код для обработки запроса и получения данных.
* Если это веб-приложение, то вам нужно будет настроить веб-сервер для обработки запросов к этому файлу.
* Протестировать модуль, отправив запрос с помощью инструмента для тестирования или браузера.

Код, который может был использован для создания такого модуля:



Рисунок 5 – пример 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 в программе. Она включает в себя диагностику проблем, выявление их причин и исправление ошибок. Отладка может быть выполнена с помощью различных инструментов, таких как отладчики, профилировщики и анализаторы кода.

Для отладки была использован способ  добавления проверок на ошибки с помощью которой после каждого запроса к базе данных добавляется проверка на ошибки и выводится сообщение об ошибке, если что-то пошло не так. Например:

if ($conn->query($sql\_driver) === TRUE) {

echo "Запись водителя успешно добавлена<br>";

} else {

echo "Ошибка записи водителя: " . $conn->error . "<br>";

}

if ($conn->query($sql\_user) === TRUE) {

echo "Запись пользователя успешно добавлена<br>";

} else {

echo "Ошибка записи пользователя: " . $conn->error . "<br>";

}

if ($conn->query($sql\_trip) === TRUE) {

echo "Запись поездки успешно добавлена<br>";

} else {

echo "Ошибка записи поездки: " . $conn->error . "<br>";

}

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.

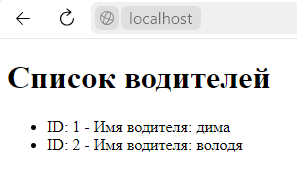
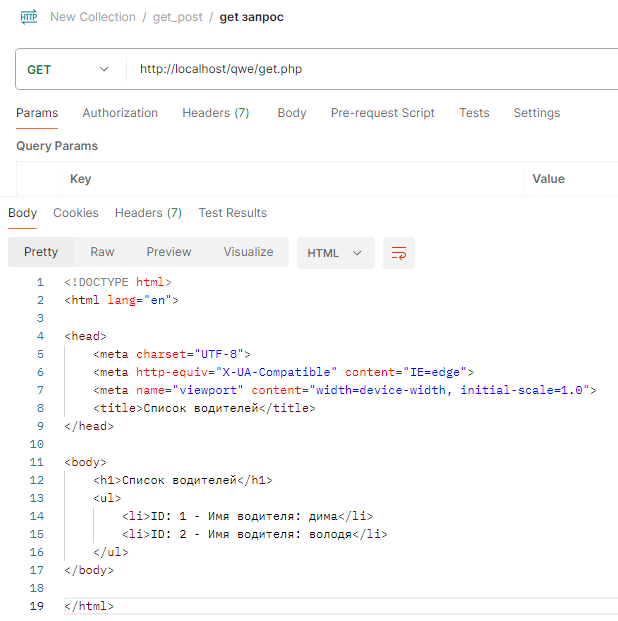


Рисунок 6 – главная страница

Страница успешно открывается

# **5 Тестовые наборы данных**

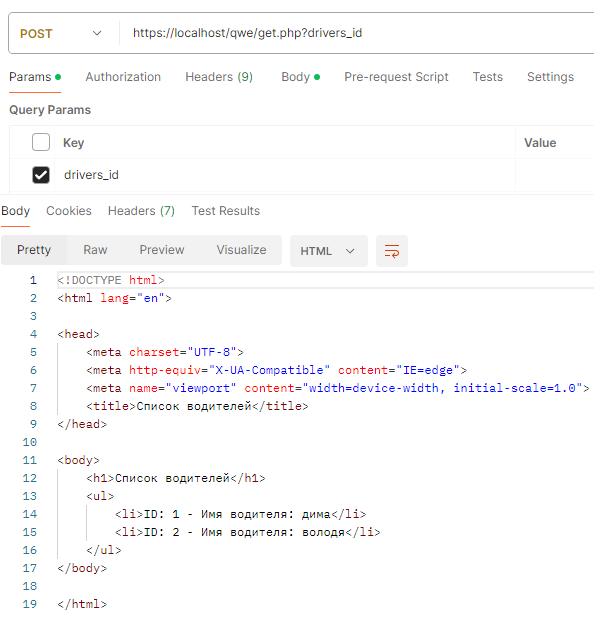
Postman - это инструмент для тестирования и отладки API (Application Programming Interface). Он предоставляет удобный интерфейс для отправки HTTP-запросов к веб-сервисам, а также для просмотра и анализа ответов от сервера.  
 С помощью Postman вы можете создавать и отправлять различные типы запросов (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 рисунке 7 представлен результат запроса GET в формате HTML.



**Рисунок 7 -** результат запроса GET

Для передачи конфиденциальных данных рекомендуется использовать метод POST.

Так же изображен результат запроса POST в формате HTML.



Postman широко используется разработчиками программного обеспечения для тестирования и отладки веб-сервисов, а также для взаимодействия с API сторонних сервисов./

# **Настройка PHPStan**

PHPSTAN - это инструмент статического анализа кода для PHP, который помогает разработчикам обнаруживать потенциальные ошибки, несоответствия типов, проблемы производительности и другие проблемы в их PHP-коде. PHPSTAN выполняет анализ кода на основе статических типов, документации и правил, что позволяет выявлять ошибки на ранних этапах разработки.

Установка:

composer require --dev phpstan/phpstan

Исполняемый файл будет лежать в vendor/bin .

Запуск:

Запустим анализ нашего кода в директории src

vendor/bin/phpstan analyse src

Очистка кэша:

vendor/bin/phpstan clear-result-cache [options]

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

Уровни анализа:

* 0 - базовые проверки, неизвестные классы, неизвестные функции, неизвестные методы, вызываемые с помощью $this, неправильное количество аргументов, передаваемых этим методам и функциям, всегда неопределенные переменные
* 1 - возможные неопределенные переменные, неизвестные магические методы и свойства в классах с \_\_call и \_\_get
* 2 - неизвестные методы проверяются для всех выражений (не только для $this ), проверяя PHPDocs
* 3 - возвращаемые типы, типы, присвоенные свойствам
* 4 - базовая проверка мертвого кода - всегда ложные проверки instanceof и других типов, мертвые ветви else, недоступный код после возврата; и т.д.
* 5 - проверка типов аргументов, передаваемых методам и функциям
* 6 - сообщить о пропущенных типографских указателях
* 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 возможные некорректные ситуации
* 8 - сообщать о вызове методов и доступе к свойствам для типов с возможностью обнуления
* 9 - будьте строги в отношении смешанного типа - единственная разрешенная операция, которую вы можете с ним выполнить, - это передать его другому смешанному типу

# **Заключение**

В ходе практики 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. Настройка PHPStan – https://backloger.ru/ru/articles/phpstan
2. Postman – https://web.postman.co/
3. Сервер – http://localhost
4. Инструкция по работе с Postman – https://www.youtube.com/watch?v=qc8At4ytwVA
5. PHPStan инструкция – https://www.youtube.com/watch?v=YvIeusei4kw&t=46s