**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

ПП.09.02.07-1.24.202.15

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

ПМ.02 Осуществление интеграции программных модулей

|  |  |  |
| --- | --- | --- |
| Руководитель от предприятия:  М.П. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (З.Ж. Кусиани) |
| Руководитель от техникума: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (С.Н. Касьяненко) | |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Л.А. Петросян) | |

Иркутск 2024

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc163156358)

[1 Создание проекта на GitHub 4](#_Toc163156359)

[2 Техническое задание на разработку базы данных 7](#_Toc163156360)

[3 Модуль для приема, проверки и занесения данных в базу данных 8](#_Toc163156361)

[4 Отладка для программного кода 10](#_Toc163156362)

[5 Работа с инструментом Postman 13](#_Toc163156363)

[6 Настройка инструмента PHP Doctor 15](#_Toc163156364)

[ЗАКЛЮЧЕНИЕ 17](#_Toc163156365)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 18](#_Toc163156366)

[Приложение А – Техническое задание 19](#_Toc163156367)

# **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  организация  «ООО  ГУД  АВТО»,  специализирующийся  на  прокате  автомобилей.

Целью производственной практики являются следующие задачи:

1. Приобрести необходимые навыки самостоятельной работы по анализу и умению работать с данными;
2. Получить практические навыки в качестве администратора баз данных;
3. Обеспечить сбор и обработку необходимого статистического материала для выполнения практических заданий.
4. Управление доступом к данным, обеспечение конфиденциальности и безопасности информации клиентов.

В процессе прохождения практики были усвоены основы поддержки базы данных, мониторинг и оптимизация производительности базы данных, и создание отчётов, также была работа в коллективе, где были применены на практике полученные знания из техникума

# **1 Создание проекта на GitHub**

GitHub представляет собой интернет-сервис, предназначенный для хранения и управления программным кодом. Этот онлайн ресурс можно описать как цифровую папку, в которой разработчики могут совместно работать над проектами, следить за изменениями, вносить корректировки и сохранять историю всех внесенных изменений. Платформа GitHub облегчает коллективную работу над программными проектами и управление версиями кода.

Для создания нового проекта на GitHub необходимо зарегистрироваться на официальном сайте https://github.com. На рисунке 1, показан процесс регистрации.

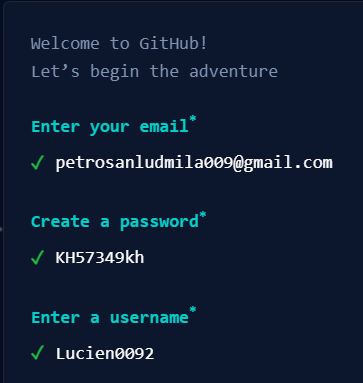


Рисунок 1 – Регистрация

После успешной регистрации открывается основная страница, где видны кнопка "Create repository", с помощью которой можно создать новый проект и ссылка "Import repository", с помощью которой можно импортировать готовый проект, рисунок 2.

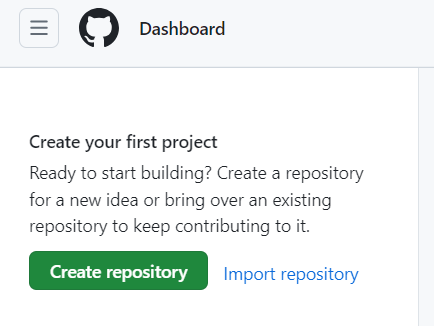


Рисунок 2 – Создание нового проекта

После нажатия на кнопку «Create repository» открывается страница, на которой можно создать новый проект, рисунок 3. На данной странице было написано название проекта и была установлена приватность проекта, публичный или частный.

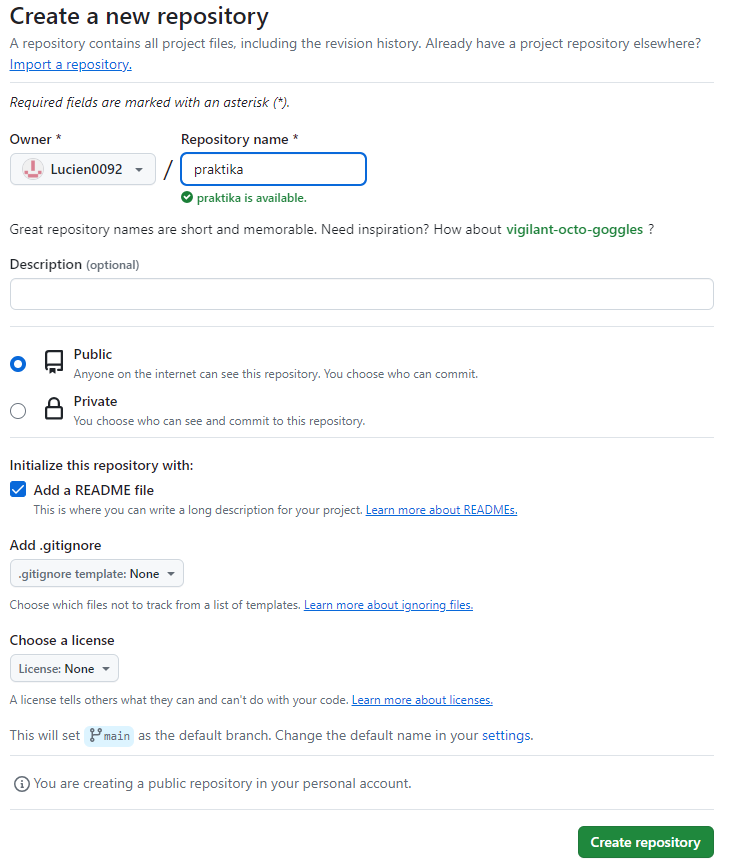


Рисунок 3 – Создание нового проекта

После нажатия на кнопку «Create repository», страница обновляется и можно увидеть, что проект был создан, рисунок 4.

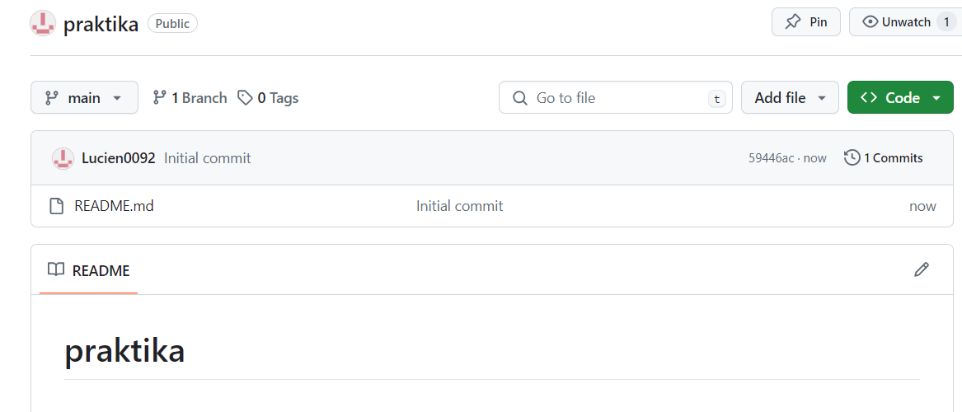


Рисунок 4 – Страница созданного проекта

Создание проекта на платформе GitHub отлично подходит для контроля версий кода, коллективной разработки и обеспечения прозрачности в процессе разработки. Этот сервис предоставляет возможность создания репозитория для хранения и управления исходным кодом проекта. Кроме этого, GitHub обеспечивает удобное отслеживание изменений в коде, вносит поправки и добавляет новый функционал, что делает совместную работу над проектом более эффективной и удобной. Этот инструмент дает возможность синхронизировать код, если работа над проектом производится команде, обеспечивает контроль за кодом и способствует более гладкому и продуктивному процессу разработки программного продукта.

# **2 Техническое задание на разработку базы данных**

В самом начале процесса разработки базы данных было подготовлено техническое задание, в котором были определены основные требования к проекту. В соответствии с требованиями, изложенными в стандарте ГОСТ 34.602-89 "Информационная технология. Комплекс стандартов на автоматизированные системы. Базы данных", техническое задание должно содержать следующие разделы:

1. Название проекта и его цель, которые определяются при создании базы данных.
2. Общие требования к самой базе данных.
3. Подробное описание структуры данных, описывающее ее компоненты и взаимосвязи.
4. Требования к обеспечению целостности данных в базе.
5. Требования к документации, необходимой для понимания и использования базы данных.

Указанное техническое задание на проектирование базы данных приведено в приложении А и является основополагающим документом для дальнейшей работы по разработке базы данных. Он служит основой для понимания требований проекта, определения его параметров и ожиданий от конечного результата.

# **3 Модуль для приема, проверки и занесения данных в базу данных**

Модуль, отвечающий за приём, проверку и запись данных в базу данных, является важной частью программного компонента, занимающегося обработкой информации. Этот модуль применяется в различных приложениях для обработки данных, получаемых от пользователей или внешних систем.

Для разработки модуля, способного принимать внешние данные, проверять их на правильность и сохранять в базе данных, рекомендуется использовать PHP для обработки данных и выполнения запросов к базе данных PostgreSQL.

Первым делом в PostgreSQL была создана база данных «praktika», в которой с помощью запроса, рисунок 5, была создана таблица «users», в которую будут записываться данные полученные из вне.

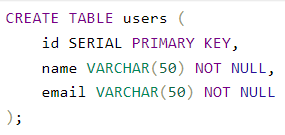


Рисунок 5 – Создание таблицы «users»

После создания базы данных и таблицы был создан файл index.php с формой для ввода данных из вне, рисунок 6.

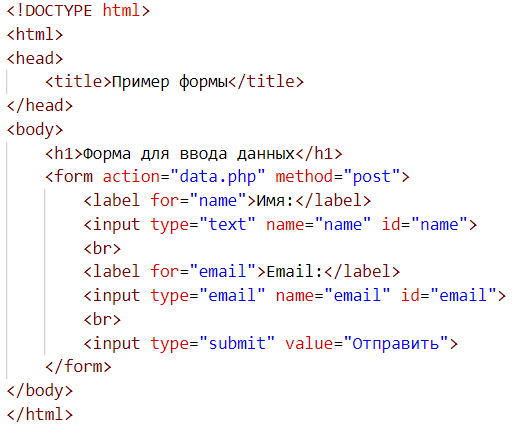


Рисунок 6 – Код создания формы

После создания формы необходимо было создать файл data.php, в котором прописано подключение к базе данных и модуль для приёма данных из вне и занесение полученных данных в таблицу «users», рисунок 7.



Рисунок 7 – Модуль приёма, проверки и занесения

Модуль приёма, проверки и занесения полученных из вне данных в таблицу, выполняет важную функцию в обработке поступающих в систему данных, и гарантирует безопасное хранение информации для дальнейшего использования приложениями или другими системами. Он является неотъемлемой частью архитектуры многих приложений и систем, где требуется эффективное управление данными. Кроме того, данный модуль обеспечивает надежность и целостность данных, что существенно для обеспечения правильной работы всего программного продукта.

# **4 Отладка для программного кода**

Отладка программного кода представляет собой процесс выявления и устранения ошибок в разрабатываемой программе. Основной целью отладки является обеспечение корректной работы программы и выполнение поставленных задач без сбоев. Для достижения этой цели применяются различные инструменты, такие как отладчики и вывод сообщений об ошибках на экран, что позволяет выявить местоположение ошибок и их причины.

Важным аспектом отладки является не только исправление самих ошибок, но и процесс анализа причин их возникновения. Путем систематического и методичного подхода к отладке программисты могут эффективно находить и устранять неполадки в программном коде, повышая качество и надежность разрабатываемых приложений. Грамотная отладка помогает разработчикам сохранить время и ресурсы, минимизируя возможность неожиданных сбоев в работе программ.

Отладка программного кода производилась с помощью расширения PHP Debug, которое было установлено в Visual Studio Code.

PHP Debug – это инструмент, который позволяет отлаживать приложения на PHP. Он предоставляет возможности для остановки выполнения кода, просмотра значений переменных, отслеживания стека вызовов и других задач отладки.

Чтобы произвести отладку было скачано расширение, которое называется PHP Debug, после установки нужно нажать на кнопку «Включить», рисунок 8

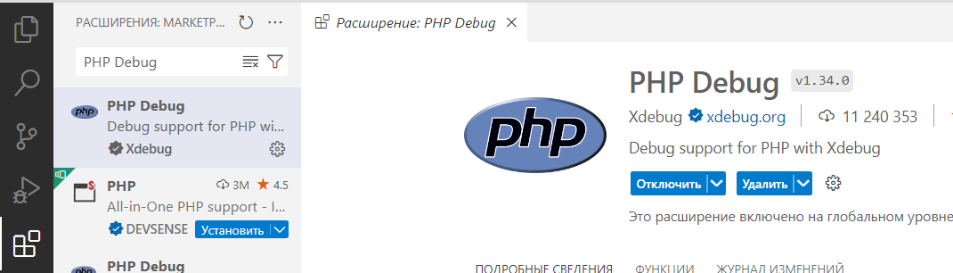


Рисунок 8 – Расширение «PHP Debug»

Стоит отметить, что PHP Debug использует Xdebug для отладки, для настройки Xdebug нужно перейти на официального сайт: https://xdebug.org/wizard. Перейдя на сайт, представлена инструкция, которая помогает найти, как необходимо загрузить и настроить PHP, для запуска Xdebug.

Следуя инструкциям, на рисунке 9 представлен код, который был добавлен в файл php.ini. Стоит отметить, что для запуска Xdebug, был установлен Xampp.

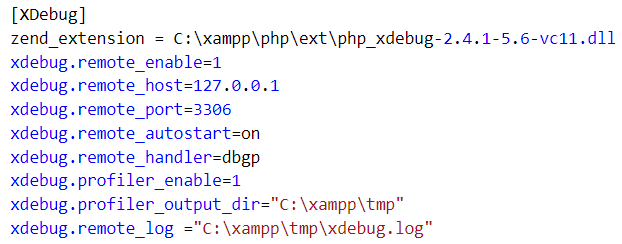


Рисунок 9 – Код установки Xdebug

После установки, необходимо настроить Visual Studio Code, для этого нужно нажать на кнопку «Запуск и отладка» и нажать на ссылку «сформировать launch.json». После нажатия на ссылку автоматически были создан файл «launch.json», рисунок 10.

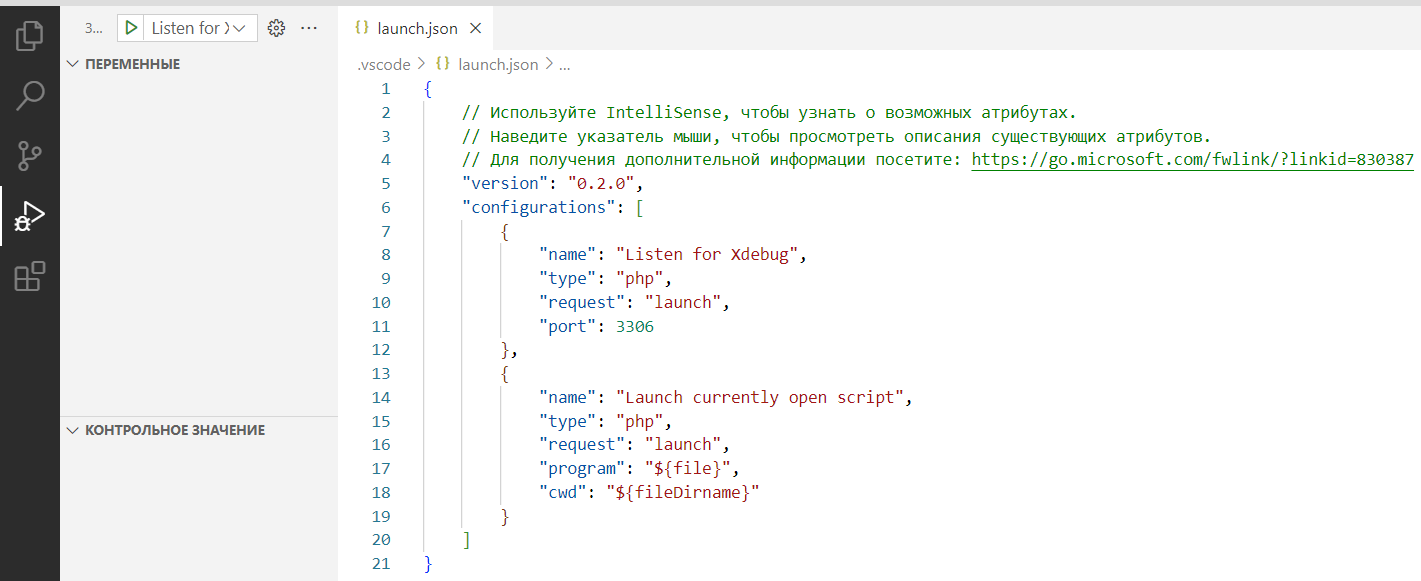


Рисунок 10 –Файл «launch.json»

Для начал отладки был открыть браузер, в котором запущено веб-приложение. После этого была открыта папка с проектом в Visual Studio Code и установлены точки остановы в нужных строках. Далее была открыта панель «Запуск и отладка» и запущена отладка с помощью команды «Listen for Xdebug», рисунок 11.

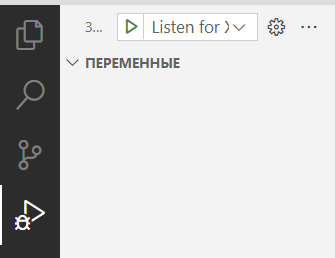


Рисунок 11 – «Запуск и отладка»

После запуска Xdebug нужно зайти в браузер обновить страницу и появятся переменные и результат отладки тех переменных, которые были выделены точкой остановы.

Помимо поиска и исправления ошибок, отладка программного кода также включает в себя тестирование программы на различных наборах входных данных, с разнообразными сценариями использования и условиями, с целью проверить ее надежность и соответствие заявленным требованиям. Этот этап процесса разработки программного обеспечения требует внимательности, систематичности и выдержки, поскольку отладка может оказаться довольно трудоемким занятием, особенно при работе с масштабными и сложными проектами.

# **5 Работа с инструментом Postman**

Postman представляет собой программный инструмент, который облегчает тестирование и взаимодействие с веб-сервисами (API). С помощью Postman можно отправлять запросы к API, анализировать полученные ответы и проверять их. Кроме этого, приложение позволяет создавать и сохранять запросы для дальнейшего использования, а также делиться ими с другими участниками проекта, что способствует коллективной работе и упрощает совместное тестирование API.

Для проведения тестирования POST и GET запросов с использованием Postman потребуется процедура регистрации или авторизации. В данном случае была выполнена регистрация, чтобы иметь доступ ко всем функциям инструмента для работы с API.

На рисунке 12 представлен GET-запрос, предназначенный для получения данных с веб-сервера. Полученные данные отображаются в формате JSON и представлены в виде списка, что упрощает их анализ и визуальное восприятие.

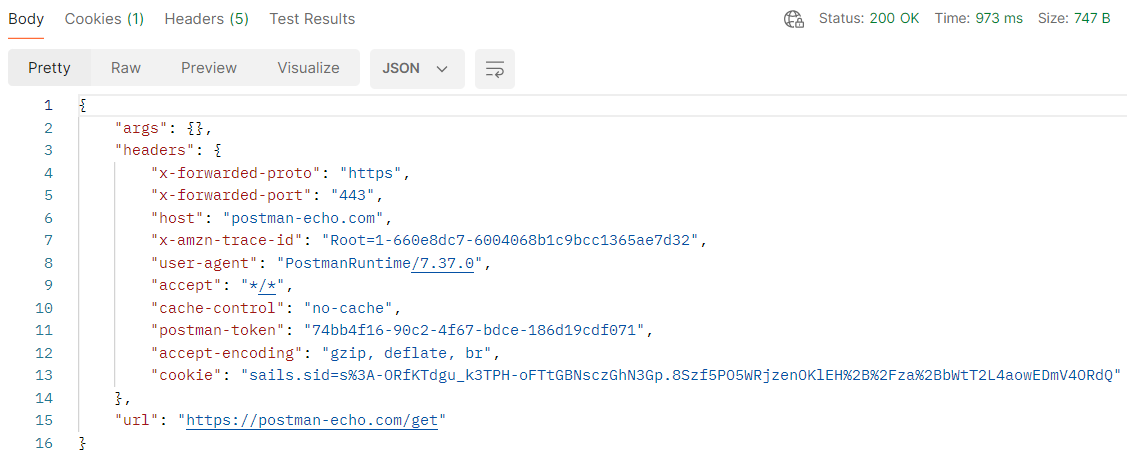


Рисунок 12 –GET-запрос

На рисунке 13 изображен POST запрос, который применяется для передачи информации на сервер с целью последующей обработки. Аналогично предыдущему примеру, полученные данные были представлены в формате JSON и отображены в удобном списочном виде.

POST-запросы широко используются для отправки данных, которые необходимо добавить или изменить на сервере.

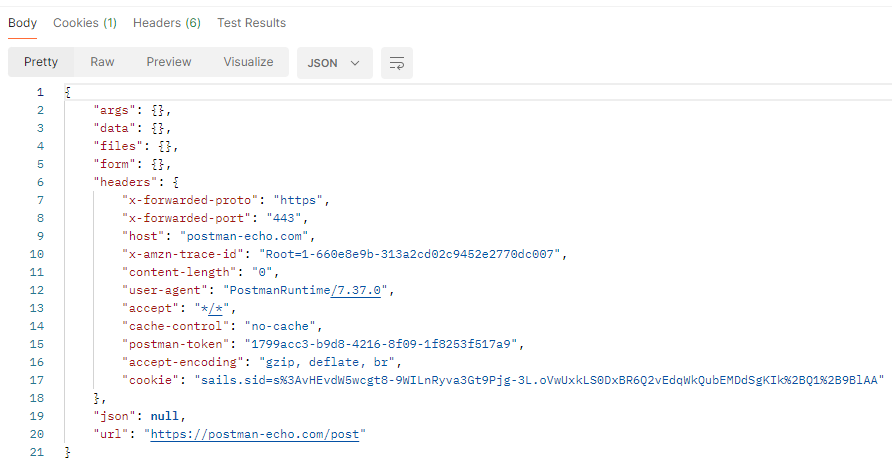


Рисунок 13 –POST-запрос

При изучении процесса создания POST и GET запросов в Postman были усвоены особенности каждого типа запроса, их цели и возможности. POST запросы часто применяются для отправки данных на сервер, таких как формы или информация для обновления базы данных, в то время как GET запросы используются для получения данных с сервера без необходимости внесения изменений на стороне сервера.

Понимание и умение создавать POST и GET запросы в Postman позволили более эффективно тестировать работу приложений и взаимодействие с API, обеспечивая более надежную и оптимизированную разработку программного обеспечения.

# **6 Настройка инструмента PHP Doctor**

PHP Doctor – инструмент для анализа PHP кода и предоставления рекомендаций по улучшению его качества. Настройка инструмента PHP Doctor позволит проводить статический анализ PHP кода и выявлять потенциальные проблемы, ошибки и улучшения, что помогает повысить качество и надежность приложения.

PHP Doctor позволяет проводить статический анализ PHP кода с целью выявления потенциальных проблем и ошибок, а также предоставляет рекомендации по улучшению кода. Этот инструмент помогает улучшить качество кода, повысить надежность приложения и уменьшить количество ошибок.

Для начала работы с PHP Doctor использовался Composer, который упрощает управление зависимостями в PHP проекте. Для этого был создан новый проект и выполнены необходимые шаги для установки Composer. После успешной установки была запущена команда в терминале, показанная на рисунке 14, чтобы установить необходимые зависимости для PHP Doctor.



Рисунок 14 – Команда установки PHP Doctor

После установки зависимостей был создан конфигурационный файл «phpdoctor.xml», который находится в корне созданного проекта. На рисунке 15 представлен код конфигурационного файла.

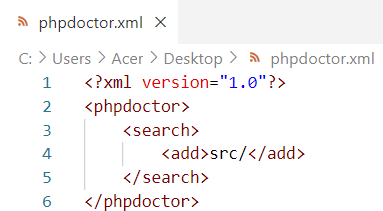


Рисунок 15 – Файл «phpdoctor.xml»

Для запуска PHP Doctor в терминале Visual Studio Code можно прописать команду, рисунок 16. После выполнения команды PHP Doctor проанализирует код и выдаст рекомендации по улучшению его качества.

****

Рисунок 16 –Команда запуска PHP Doctor

Использование PHP Doctor помогло обнаружить потенциальные проблемы в PHP коде, а также обеспечило ценные рекомендации по оптимизации кода и улучшению его структуры.

Правильная настройка PHP Doctor позволяет производить статический анализ PHP кода, выявлять потенциальные проблемы и ошибки, а также можно получить рекомендации по улучшению качества программного кода. PHP Doctor упрощает процесс разработки, помогает предотвращать возможные ошибки и повышает надёжность программного продукта.

# **ЗАКЛЮЧЕНИЕ**

Практика в области программных интеграций является ключевым компонентом современного развития программного обеспечения. Понимание процесса интеграции различных систем, сервисов и приложений играет важную роль в обеспечении эффективной работы всей системы.

В рамках практики по программным интеграциям были выполнены следующие этапы:

1. Создание проекта на GitHub.
2. Разработка технического задания для определения требований к создаваемой базе данных.
3. Написание модуля для обработки входящих данных из вне, их проверка на корректность и сохранение в базе данных.
4. Отладка программного кода для обеспечения его правильной работы.
5. Работа с Postman, включая проверку и тестирование функциональности GET и POST запросов.
6. Настройка PHP Doctor для обеспечения статического анализа PHP кода и выявления потенциальных проблем.

Практические упражнения по программным интеграциям позволили осознать сложность и специфику данного процесса, что позволило более эффективно выполнять задачи интеграции. Помимо этого, важно учитывать значимость тестирования, обеспечения безопасности, мониторинга и документирования во время работы над интеграциями, чтобы создать стабильную, безопасную и легко поддерживаемую систему.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. peej.co/PHPDoctor: The PHP API Documentation Creator /. URL: https://www.peej.co.uk/phpdoctor/ – (дата обращения: 02.04.2024) – Текст: электронный.
2. skillbox./Postman: что это такое и как им пользоваться/. URL: https://skillbox.ru/media/code/postman-chto-eto-takoe-i-kak-im-polzovatsya/ – (дата обращения: 25.03.2024) – Текст: электронный.
3. cntd/Межгосударственный стандарт информационная технология /. URL: https://docs.cntd.ru/document/1200006924 – (дата обращения: 25.03.2024) – Текст: электронный.
4. skillbox/Инструкция: заливаем проект на GitHub без командной строки/. URL: https://skillbox.ru/media/code/instruktsiya\_zalivaem\_proekt\_na\_github\_bez\_komandnoy\_stroki/– (дата обращения: 25.03.2024) – Текст: электронный.
5. gb/ Как тестировать API, или Postman для чайников/. URL: https://gb.ru/blog/kak-testirovat-api-postman/ – (дата обращения: 25.03.2024) – Текст: электронный.

**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 от предприятия:  М.П. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (З.Ж. Кусиани) |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (С.Н. Касьяненко) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Л.А. Петросян) |

Иркутск 2024

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

Наименование проекта: Каршеринг.

Цель создания проекта: Автоматизация пользователей, управление и бронирование автомобилей.

Исполнитель: студент Иркутского авиационного техникума группы БД-20-2, Петросян Л.А.

Разработка базы данных проходит в рамках производственной практики.

Срок разработки базы данных с 15.03.24 по 04.04.24 года.

**2 Общие требования к базе данных.**

Общие требования к базе данных включают в себя стандартные принципы нормализации и обеспечение целостности данных, оптимизацию производительности путем эффективного написания запросов, а также выбор подходящих типов данных и создание индексов для ускорения доступа к данным. Кроме этого, необходимо регулярно создавать резервные копии базы данных, чтобы обеспечить возможность восстановления данных в случае сбоев или ошибок. Важно обеспечить надежность и безопасность хранения данных, что позволит избежать потери информации и обеспечить непрерывную работу вашего приложения.

**3 Описание структуры данных.**

Для структуры базы данных предназначенной для каршеринга необходимо создать три таблицы:

1. Таблица «Пользователи»:

* id (идентификатор пользователя);
* имя (имя пользователя);
* фамилия (фамилия пользователя);
* электронная почта (email пользователя);
* пароль (хэшированный пароль пользователя);
* телефон (номер телефона пользователя);
* дата регистрации (дата регистрации пользователя);

1. Таблица «Автомобили»:

* id (идентификатор автомобиля);
* марка (марка автомобиля);
* модель (модель автомобиля);
* цвет (цвет автомобиля);
* год выпуска (год выпуска автомобиля);
* стоимость аренды в час (цена аренды автомобиля за час);
* статус (доступен/занят);

1. Таблица «Бронирование»:

* id (идентификатор бронирования);
* id\_пользователя (идентификатор пользователя, который бронирует автомобиль);
* id\_автомобиля (идентификатор автомобиля, который бронируется);
* дата начала аренды (дата начала бронирования);
* дата окончания аренды (дата окончания бронирования);
* статус (ожидание/подтверждено/отменено);

**4.Требования к целостности данных.**

Среди требований к целостности данных можно выделить следующие аспекты:

1. Гарантированная уникальность данных: каждый работник и каждая деталь должны иметь свой уникальный идентификационный номер, чтобы исключить возможность дублирования записей.
2. Надежная целостность связей: связи между таблицами должны быть сформированы корректно и обеспечивать правильное взаимодействие и ссылочную целостность между связанными объектами.
3. Соблюдение ограничений целостности: важно установить ограничения на поля данных, например, требование заполнения поля с должностью работника, чтобы обеспечить корректное и полное представление информации в базе данных. Такие ограничения помогут предотвратить ошибки и обеспечить согласованность данных в базе.

Обеспечение целостности данных является важным аспектом проектирования базы данных, который гарантирует правильность и надежность информации, хранящейся в системе. Правильная реализация требований к целостности данных помогает избежать ошибок, защищает от нарушения целостности информации и обеспечивает правильную работу базы данных в целом.

**5 Требования к документации**

Основными документами при разработке базы данных являются два документа:

1. Руководство пользователя является ключевым документом, определяющим правила и процедуры использования базы данных для пользователей. Этот документ предоставляет инструкции по работе с базой данных, объясняет основные функции системы, описывает возможности и правила доступа к данным.
2. Техническое задание – основной документ, который регламентирует разработку базы данных технических аспектов. В нем определяются требования к структуре базы данных, ее функциональности, а также технологии и инструменты, которые будут использоваться при создании базы данных.

Оба документа играют важную роль в процессе разработки и использования базы данных, обеспечивая понимание требований, регламентирующих работу с данными, исключая возможные ошибки и несоответствия. Они помогают обеспечить эффективное взаимодействие пользователя с базой данных и гарантируют правильное функционирование информационной системы в целом.