Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Системы автоматизированного проектирования»

Пенза 2021

Курсовая работа

по дисциплине

«Информационное обеспечение систем автоматизированного проектирования»

на тему «Проектирование базы данных задач студента и разработки приложения их учета»

Направление подготовки 09.03.01 «Информационная и

вычислительная техника»

профиль «Системы автоматизированного проектирования»

Выполнила ст. группы 20ВВ2.1 Горбунов Н.А.

Руководитель:

к.т.н. доцент Глотова Т.В.

Пенза 2023

Изм.

Лист

№ докум.

Подпись

Дата

Лист

*1*

КР.09.03.01. 5.23 81. 01

Разраб.

Горбунов Н.А.

Провер.

*Глотова Т. В.*

Реценз.

Н. Контр.

Утверд.

Проектирование БД и разработка приложения приюта для животных

Пояснительная записка

Лит.

Листов

*1*

Гр. 20ВВ2.1

Пояснительная записка содержит 74 листов, 41 рисунков, 16 таблиц, 7 источников, 2 приложения.

ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, СТУДЕНЧЕСКАЯ ЖИЗНЬ, БАЗА ДАННЫХ, MySQL WORKBENCH, СУБД MYSQL, LOCALHOST, PHPMyAdmin, PHP, UNITY, C#, SUBLIME TEXT 3, VISUAL STUDIO 2019.

Объектом разработки является база данных задач студента и разработки приложения их учета.

Цель работы – проектирование базы данных и создание приложения задач студента и разработки приложения их учета.

Задачами, которые требовалось решить для достижения цели, являются: изучение предметной области, возможностей программных средств проектирования и разработки, проектирование и разработка базы данных и разработка программного обеспечения.

Результатом работы является приложения для помощи в решении задач студента и разработки приложения их учета, разработанные с помощью средств для проектирования баз данных MySQL Workbench, СУБД MySQL, языков PHP, C#.

**Содержание**

[Введение 6](#_Toc125066743)

[1 Инфологическое проектирование 7](#_Toc125066744)

[1.1 Инфологическая модель 7](#_Toc125066745)

[Логическое проектирование 10](#_Toc125066746)

[2.1 Нормализация 10](#_Toc125066747)

[Соответствие нормальным формам 13](#_Toc125066748)

[Физическое проектирование 16](#_Toc125066749)

[3.1 Типы данных 16](#_Toc125066750)

[3.2 Индексы 19](#_Toc125066751)

[5 Разработка приложения 22](#_Toc125066752)

[Заключение 50](#_Toc125066753)

[Список используемой литературы 51](#_Toc125066754)

[Приложение А 51](#_Toc125066755)

[Приложение Б 55](#_Toc125066756)

**Введение**

Как часто, когда вы студент, задаетесь вопросами о том, как правильно нужно выполнять лабораторную работу, где искать нужную информацию и у кого спросить совет. Решения этого вопроса стали появляться, когда интернет вошел в нашу обыденную жизнь, появились сайты «ГДЗ» и форумы, где каждый мог задать вопрос, но увы, ответы были либо слишком поверхностные, либо оскорбительные. Ведь люди, которые отвечают на них, совсем не знают преподавателей и не слушали лекции, которые должны были слушать вы. Поэтому, отличным решением будет создать приложение, где студенты с одного вуза смогут выкладывать полезную информацию для выполнения лабораторных работ, и эта информация вправду будет полезной, ведь эти студенты точно должны знать, как выполняется работа.

Для разработки бэкенда был использован язык PHP. PHP - язык программирования, специально разработанный для написания Web-приложений (скриптов, сценариев), исполняющихся на Web-сервере.

А для разработки десктопного приложения был выбран движок UNITY, который работает вместе с языком C#.

Целью курсовой работы является разработка базы данных приложения для обмена полезной информацией для лабораторных работ. База данных подразумевает хранение файлов пользователей, полезных ссылок, вопросов и ответов.

Для достижения поставленной цели были определены следующие задачи:

- изучить возможности СУБД MySQL;

- изучить возможности CASE-средства MySQL Workbench для проектирования БД;

- разработать структуру базы данных;

- создать базу данных c использованием СУБД MySQL в среде phpMyAdmin;

- разработать запросы на языке SQL скрипты на PHP для работы с базой данных;

- разработать интерфейс на Unity для работы с базой данных.

**1 Инфологическое проектирование**

* 1. **Инфологическая модель**

Предметная область – Проектирование базы данных задач студента и разработки приложения их учета.

Инфологическая модель заданной предметной области в виде диаграммы IDEFIX:

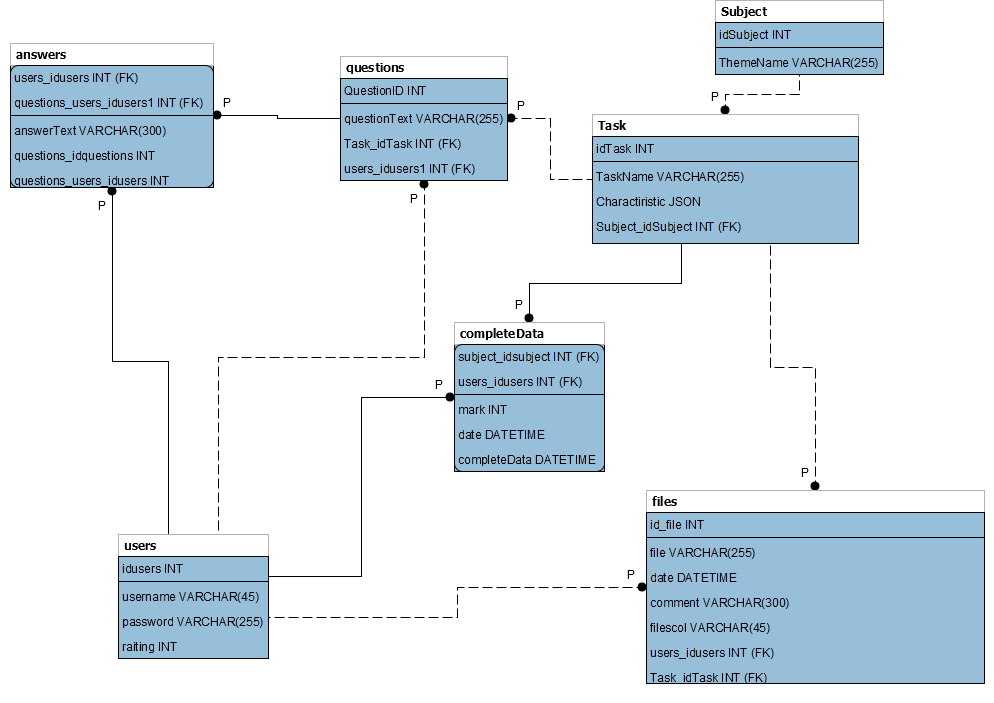


Рисунок - 1

Описание сущностей и связей.

1. Users – таблица содержащая информацию о зарегистрированном студенте и его рейтинг  
   Таблица users связана с таблицами files, completeData, Questions, answers.  
   Включает в себя такие атрибуты:

* idUsers – id пользователя
* username – ФИО пользователя
* password – Пароль пользователя
* rating – рейтинг пользователя (Нужен для того, чтобы пользователям, прежде чем смотреть полезные файлы других, нужно было внести свой вклад в развитие сайта)

1. Files – таблица содержащая файл прикрепленный студентом к лабораторной работе, она содержит в себе:

* Id\_file – id отдельного файла
* File – ссылку на полезные ресурсы
* Date – дату, когда запись была выложена
* Comment – комментарий пользователя, который выложил этот файл
* Users\_idUsers – id пользователя, который выложил файл
* Task\_idTask – id предмета, по которому был прикреплен файл

1. Subject – Предмет по которому выполняется лабораторная работа

* idSubject – id предмета
* ThemeName – Название предмета

1. Task – Сама лабораторная работа к которой прикреплены файлы студентов, которые выполнили её

* IdTask – id лабораторной работы
* TaskName – Название лабораторной работы (по ним будет производиться фильтр)
* CharacteristicJson – json файл в который будет вложена дополнительная информация по лабораторной работе
* Subject\_idSubject – id предмета, по которой выполняется лабораторная работа (по нему тоже будет производиться фильтр)

1. Questions – Вопросы по лабораторной работе, которые задают студенты

* QuestionID – id вопроса
* QuestionText – текст вопроса, который задал пользователь
* Task\_idTask – id лабораторной работы, по которой был задан вопрос
* User\_idUser – id пользователя, который задал вопрос

1. Answers – Ответы на вопросы других студентов

* User\_idUser – id пользователя, который ответил на вопрос
* Question\_idQuestion – id вопроса, к которому прикреплен данный ответ
* answerText – текст ответа

1. CompleteData – таблица привязанная к лабораторной работе и студенту, в ней содержится информация по оценке и комментарии к ней.

* Subject\_idSubject – id лабораторной работы, по которой содержится информация
* User\_idUser – id пользователя, который получил оценку, по данной лабораторной работе
* Mark – оценка
* Date – Дата, когда нужно было сдать работу
* CompleteData – Дата, когда пользователь сдал работу

1. Связь между Task и users – многие к многим, реализуется через CompleteData
2. Subject к Task – 1 к многим не идентифицирующая
3. Task к questions – 1 к многим не идентифицирующая
4. Task к files – 1 к многим не идентифицирующая
5. questions к answers – 1 к многим прямая
6. users к answers – 1 к многим не идентифицирующая
7. users к questions – 1 к многим не идентифицирующая

# **Логическое проектирование**

## **2.1 Нормализация**

База данных была создана с использованием СУБД MySQL на локальном сервере (пакет XAMPP: Apache/MySQL/PHP и phpMyAdmin).

Загрузили разработанную модель.

Запустили модули Apache и MySQL

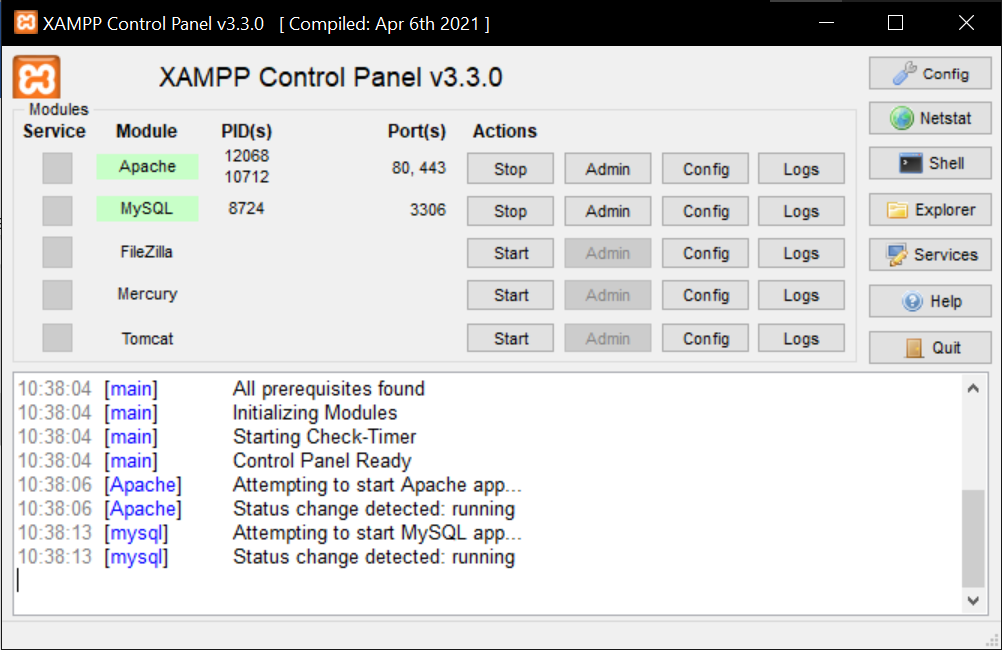


Рисунок 2 – Xammp панель

В MySQL Workbench создали новое соединение с сервером и дали ему имя, вставили скрипт и запустили его

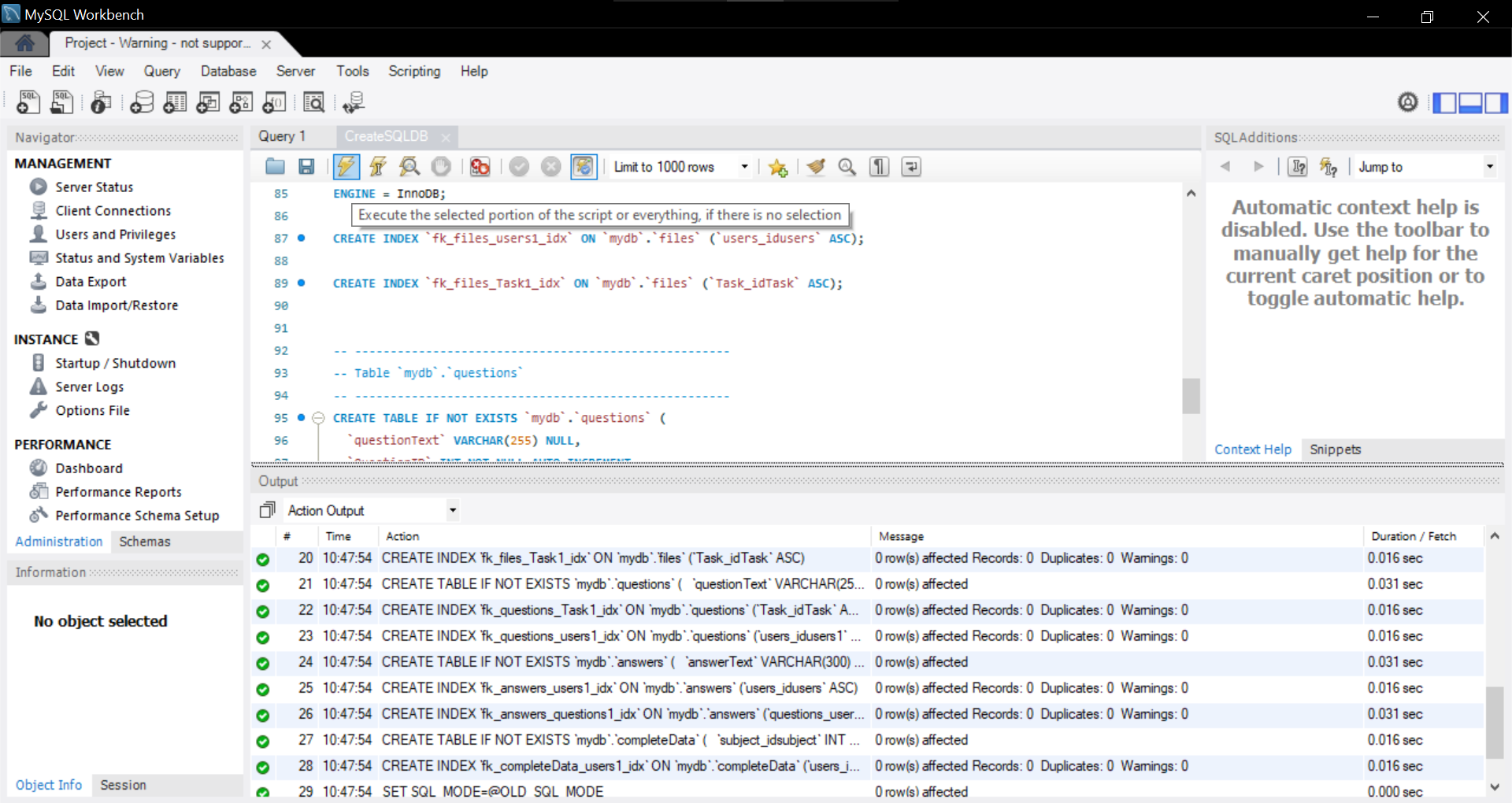


Рисунок – 3 Скрипт в MySQL Workbench

Зашли в PHPMYADMIN чтобы убедиться, что таблица создалась.

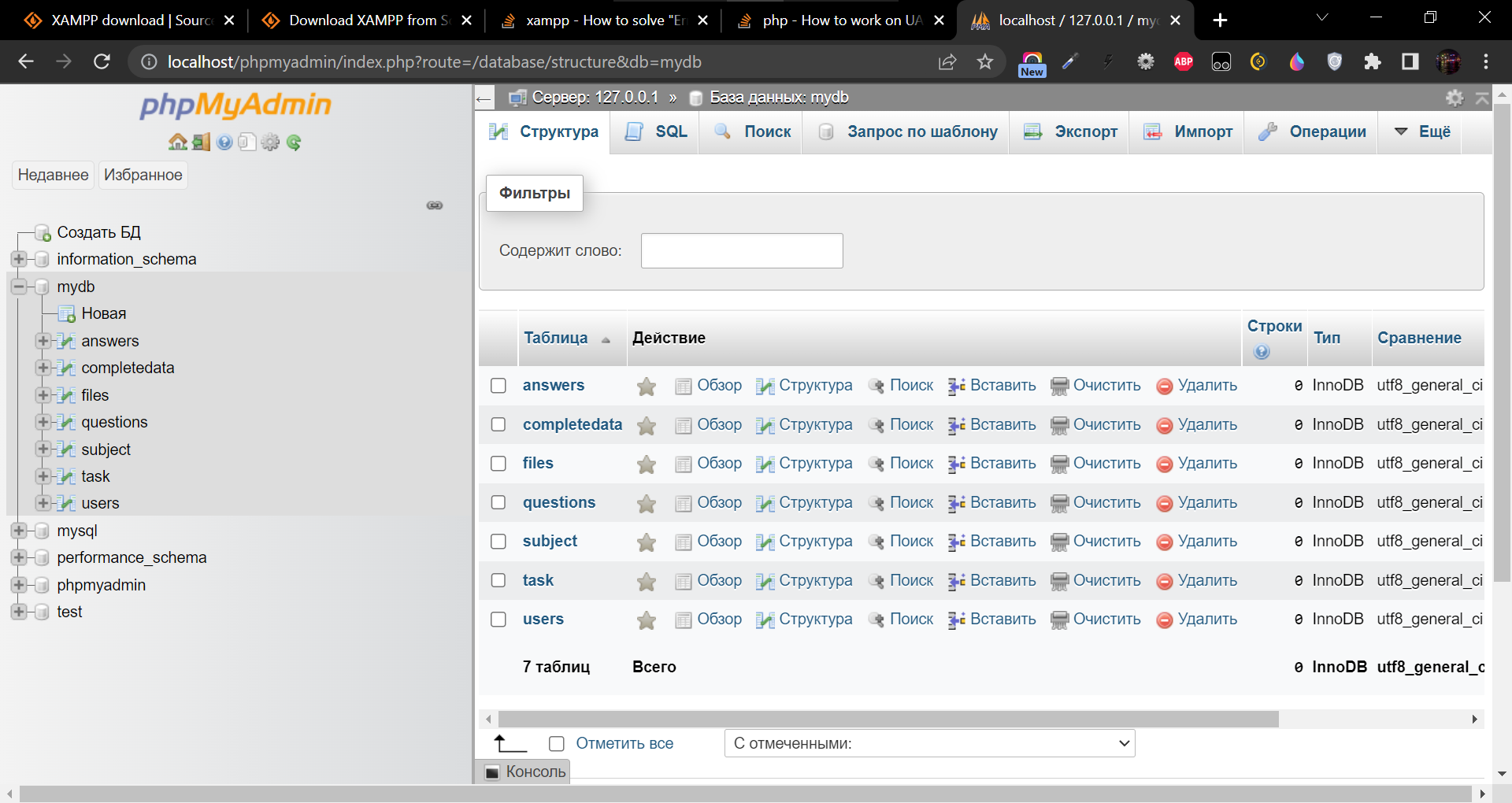


Рисунок 4 – PHPMyAdmin наша база данных

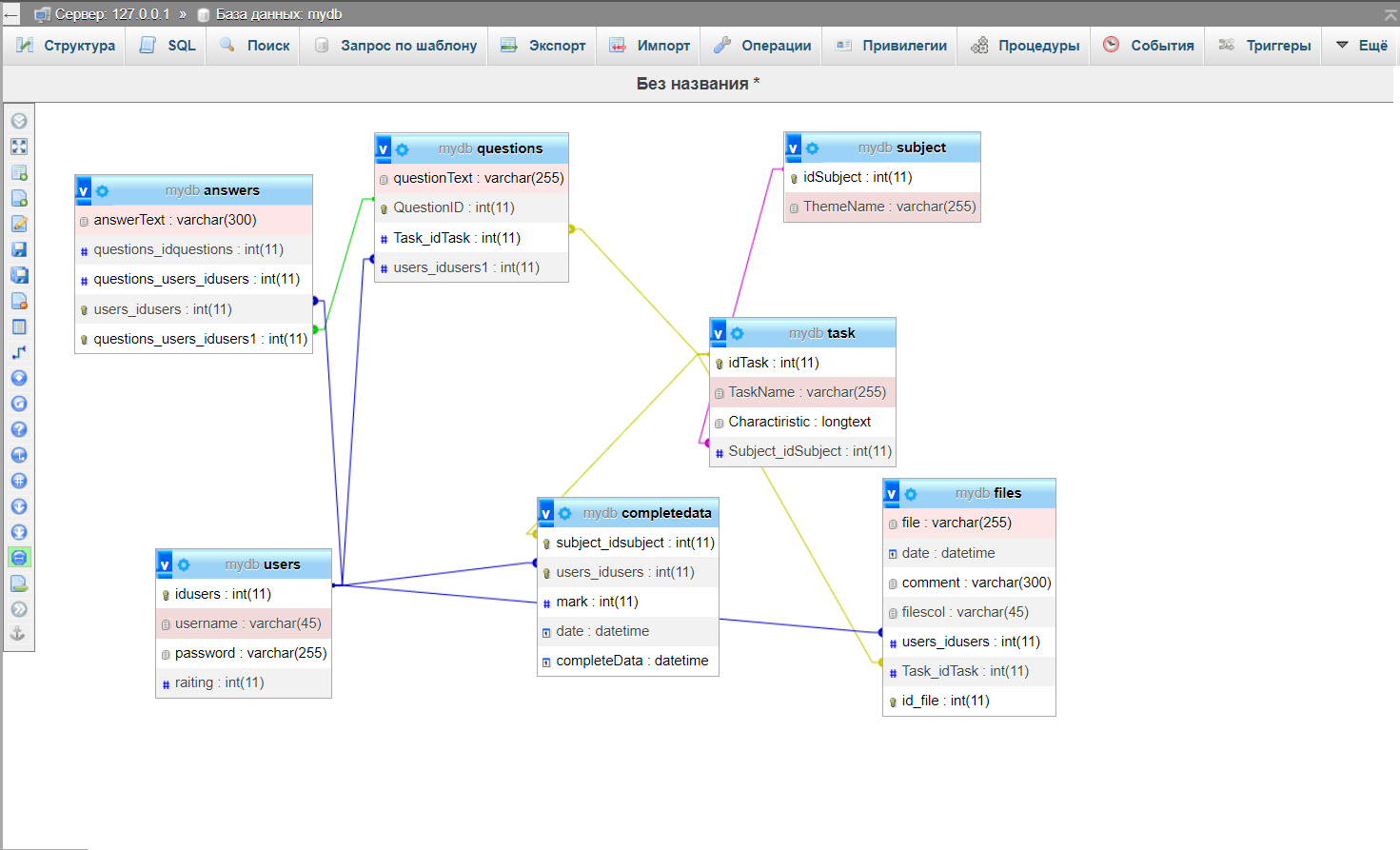


Рисунок 5 – Схема базы в PHPMyAdmin

**Соответствие нормальным формам**

**Первая нормальная форма**

Основным правилом первой формы является необходимость неделимости значения в каждом поле (столбце) строки – атомарность значений (рис. 6).

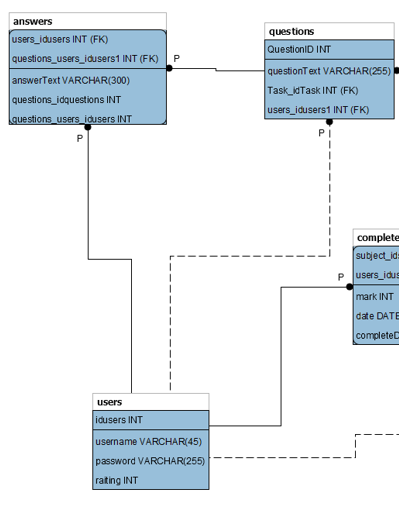
****

Рисунок 6 – Первая нормальная форма

Помимо атомарности к первой нормальной форме относятся следующие правила:

• Строки таблиц неупорядочены, т.е. размещение записей в таблице не имеет никакого значения.

• Аналогичная ситуация со столбцами записей. Все столбцы имеют имена. Порядок не должен влиять на понимание информации.

• Каждая строка должна быть уникальна, поэтому для нее определяется первичный ключ, состоящий из одного либо нескольких полей (составной ключ). Первичный ключ не может повторяться в пределах таблицы и служит идентификатором записи.

**Вторая нормальная форма**

Условием этой формы является отсутствие зависимости неключевых полей от части составного ключа и соответствие первой нормальной форме. Рассмотрим пример на таблице answers (рис. 7).

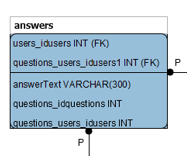
****

Рисунок 7 – Вторая нормальная форма

Атрибут Data зависит от полного составного ключа, а не от его части.

**Третья нормальная форма**

Условием этой формы является соответствие второй нормальной форме, и чтобы ни один неключевой атрибут не находился в зависимости от других неключевых атрибутов. Рассмотрим пример на таблице Animal (рис. 8).

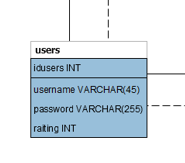
****

Рисунок 8 – Третья нормальная форма

Каждый неключевой атрибут таблицы зависит от первичного ключа

1. Таблица **answers**:

user\_id, question\_id → answerText

user\_id (Таблица **users**) → idusers

question\_id (Таблица **questions**) → QuestionID

Соответствует 3 нормальной форме.

1. Таблица **completedata**

task\_id (Таблица **task**) → idTask

user\_id (Таблица **users**) → idusers

task\_id, user\_id → mark

task\_id, user\_id → date

task\_id, user\_id → completeData

Соответствует 3 нормальной форме.

1. Таблица **files**

id\_file → file

id\_file → date

id\_file → comment

users\_idusers (Таблица **users**) → idusers

Task\_idTask (Таблица **task**) → idTask

Соответствует 3 нормальной форме.

1. Таблица **questions**

QuestionID → questionText

users\_idusers1 (Таблица **users**) → idusers

Task\_idTask (Таблица **task**) → idTask

Соответствует 3 нормальной форме.

1. Таблица **subject**

idSubject → ThemeName

Соответствует 3 нормальной форме.

1. Таблица **task**:

idTask → TaskName

idTask → Charactiristic

idTask → TaskDate

Subject\_idSubject (Таблица **subject**) → idSubject

Соответствует 3 нормальной форме.

1. Таблица **users**:

idusers → username

idusers → password

idusers → raiting

Соответствует 3 нормальной форме.

**Физическое проектирование**

**3.1 Типы данных**

1) Используемые типы данных в таблице answers изображены на рисунке 9.

****

Рисунок 9 – Типы данных в таблице answers

2) Используемые типы данных в таблице completedata изображены на рисунке 10.

****

Рисунок 10 – Типы данных в таблице completedata

3) Используемые типы данных в таблице files изображены на рисунке 11.

****

Рисунок 11 – Типы данных в таблице files

4) Используемые типы данных в таблице questions изображены на рисунке 12.

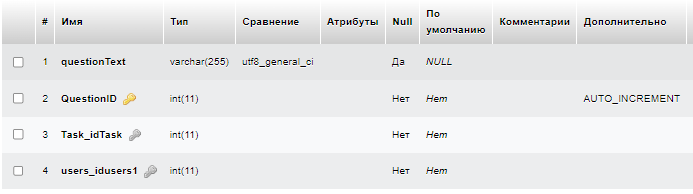
****

Рисунок 12 – Типы данных в таблице questions

5) Используемые типы данных в таблице subject изображены на рисунке 13.

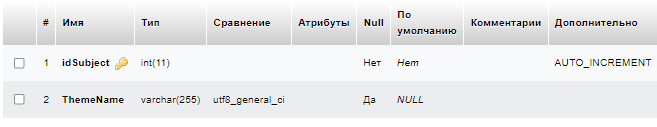
****

Рисунок 13 – Типы данных в таблице subject

6) Используемые типы данных в таблице task изображены на рисунке 14.

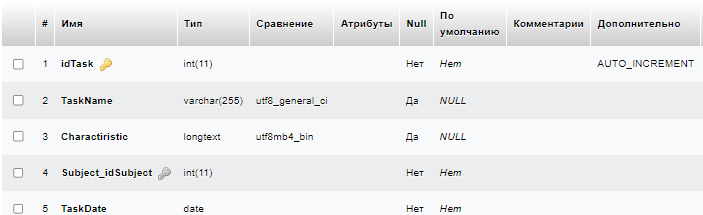
****

Рисунок 14 – Типы данных в таблице task

7) Используемые типы данных в таблице users изображены на рисунке 15.

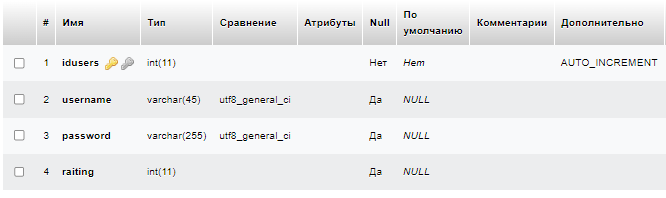
****

Рисунок 15 – Типы данных в таблице users

**3.2 Индексы**

Индексы в таблицах созданы для всех полей первичного ключа и внешних ключей.

1. Индексы в таблице answers изображены на рисунке 16

****

Рисунок 16 – Индексы в таблице Animal

**PRIMARY** – первичный ключ столбец answers;

**fk\_answers\_users1\_idx** – внешний ключ, передаваемый из таблицы users;

**fk\_answers\_questions1\_idx** – внешний ключ, передаваемый из таблицы question.

1. Индексы в таблице completeData изображены на рисунке 17

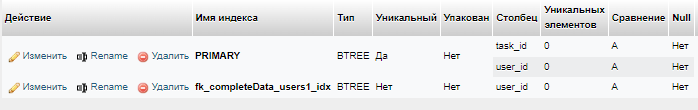
****

Рисунок 17 – Индексы в таблице completeData

**PRIMARY** – первичный ключ столбец completeData.

**fk\_completeData\_users1\_idx** – внешний ключ, передаваемый из таблицы users;

1. Индексы в таблице files изображены на рисунке 18

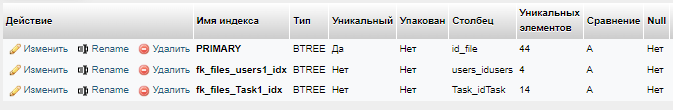
****

Рисунок 18 – Индексы в таблице files

**PRIMARY** – первичный ключ столбец files.

**fk\_files\_users1\_idx** – внешний ключ, передаваемый из таблицы users.

**fk\_files\_Task1\_idx** – внешний ключ, передаваемый из таблицы task.

1. Индексы в таблице questions изображены на рисунке 19

****

Рисунок 19 – Индексы в таблице questions

**PRIMARY** – первичный ключ столбец questions;

**fk\_questions\_Task1\_idx**– внешний ключ, передаваемый из таблицы task.

**fk\_questions\_users1\_idx** – внешний ключ, передаваемый из таблицы users.

1. Индексы в таблице subject изображены на рисунке 20

****

Рисунок 20 – Индексы в таблице subject

**PRIMARY** – первичный ключ столбец id subject.

1. Индексы в таблице task изображены на рисунке 21

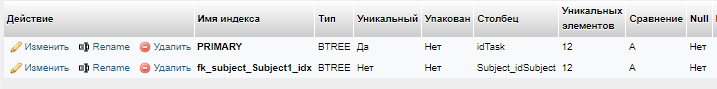
****

Рисунок 21 – Индексы в таблице task

**PRIMARY** – первичный ключ столбец id task;

**fk\_subject\_Subject1\_idx** – внешний ключ, передаваемый из таблицы Subject\_idSubject.

1. Индексы в таблице users изображены на рисунке 22

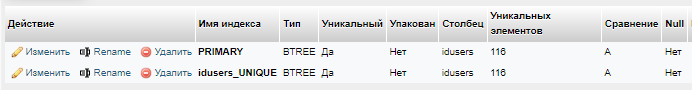
****

Рисунок 22 – Индексы в таблице users

**PRIMARY** – первичный ключ столбец id users;

**idusers\_UNIQUE –** уникальныйидентификатор;

Question.php

Task.php

Registration.php

DataBaseConnection.php

DataBaseConnection.php – файл для создания соединения с сервером

Registration.php – Создание новых пользователей или авторизации

Task.php – Создание и удаление дисциплин и лабораторных работ

Question.php – Создание и удаление вопросов и ответов.

# **5 Разработка приложения**

Первым делом нужно спланировать, как будет связываться приложение с базой данных.   
Я решил, что лучше всего связывать приложение будет с помощью POST запросов.

У каждого запроса будет свой ID, чтобы сервер мог различать, что нужно присылать в ответ.

Далее создадим новый проект на Unity и сверстаем UI меню для регистрации или входа пользователя.

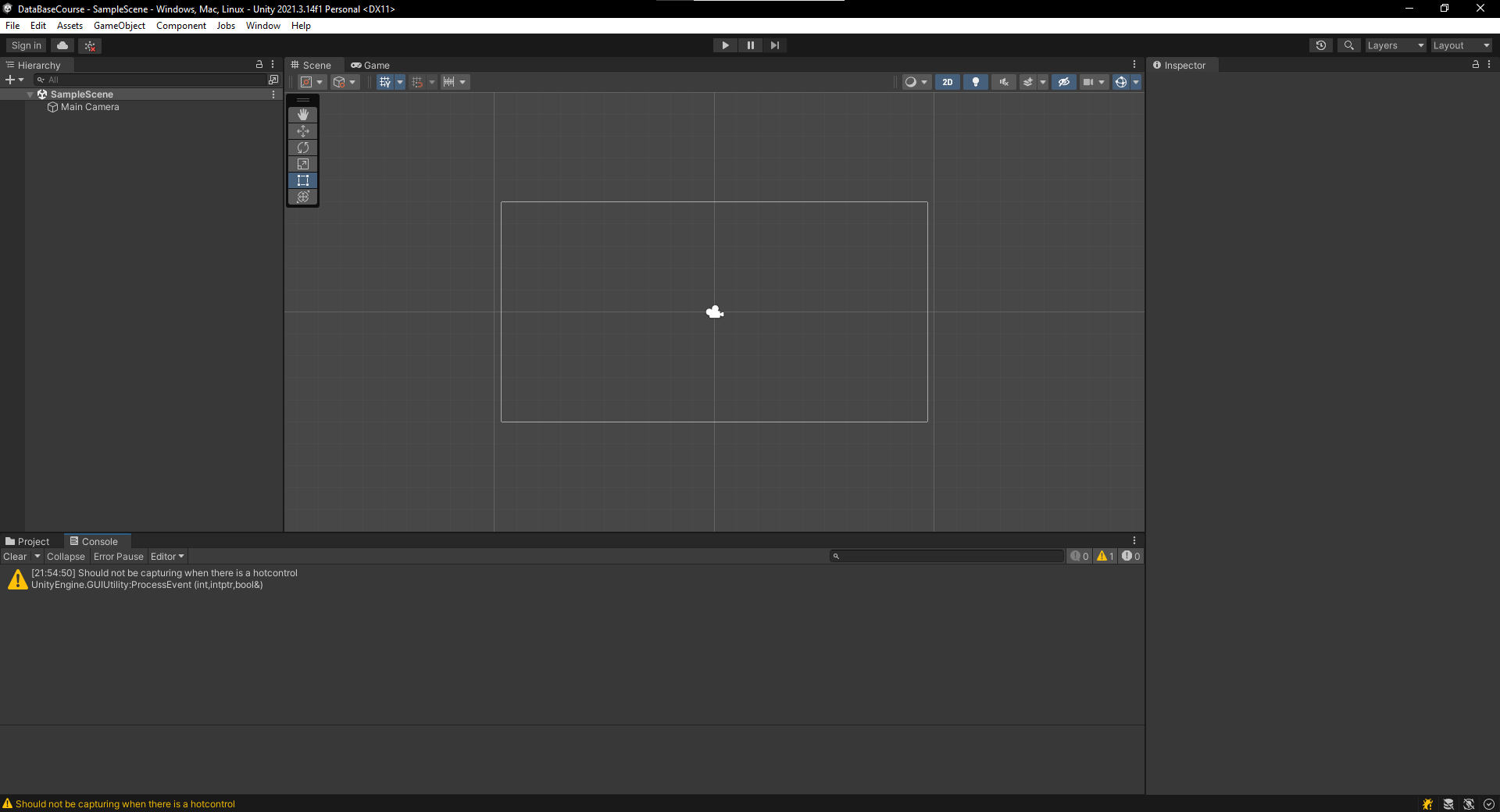


Рисунок 23 – Новый проект

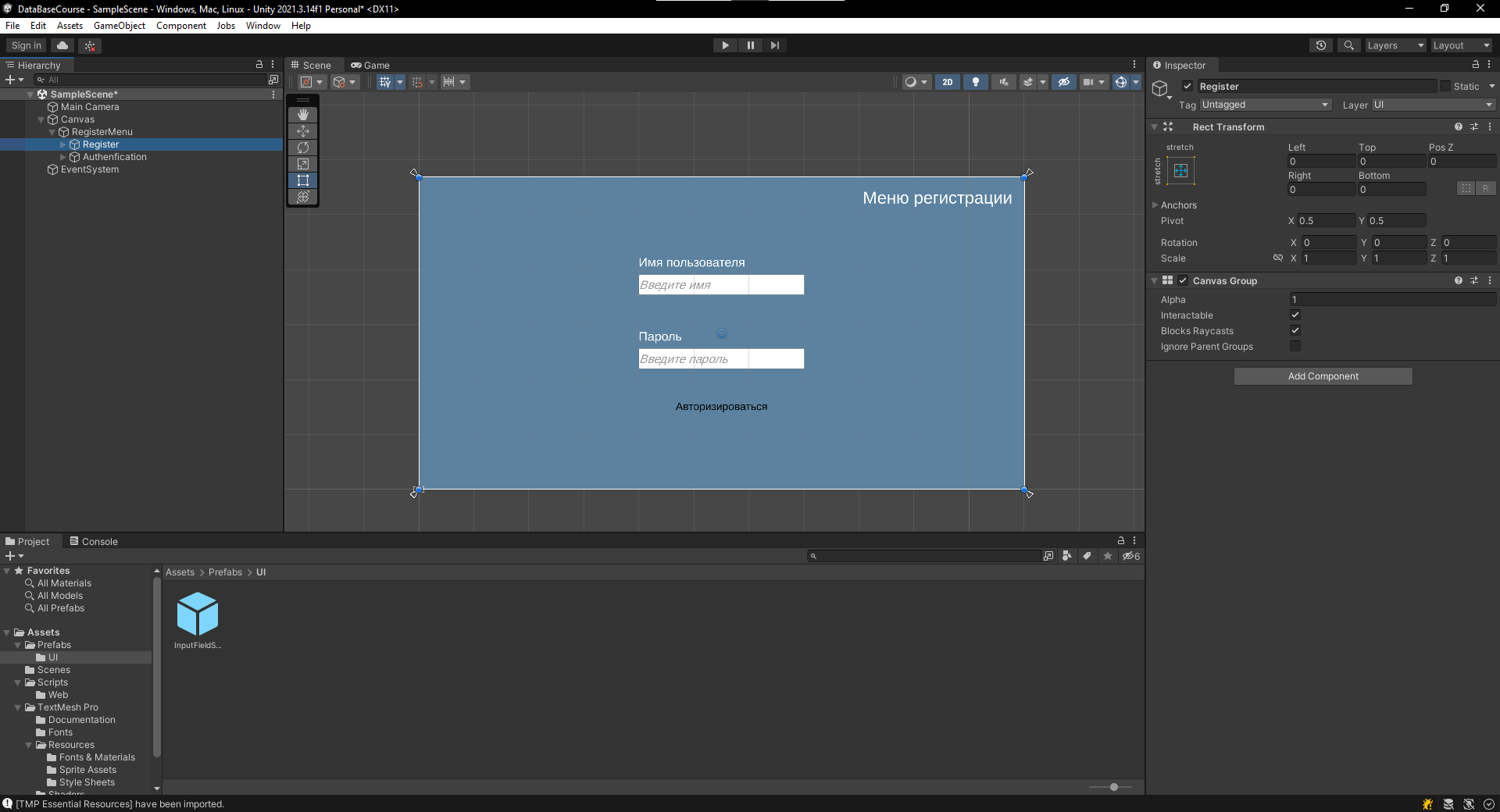


Рисунок 24 – Меню регистрации

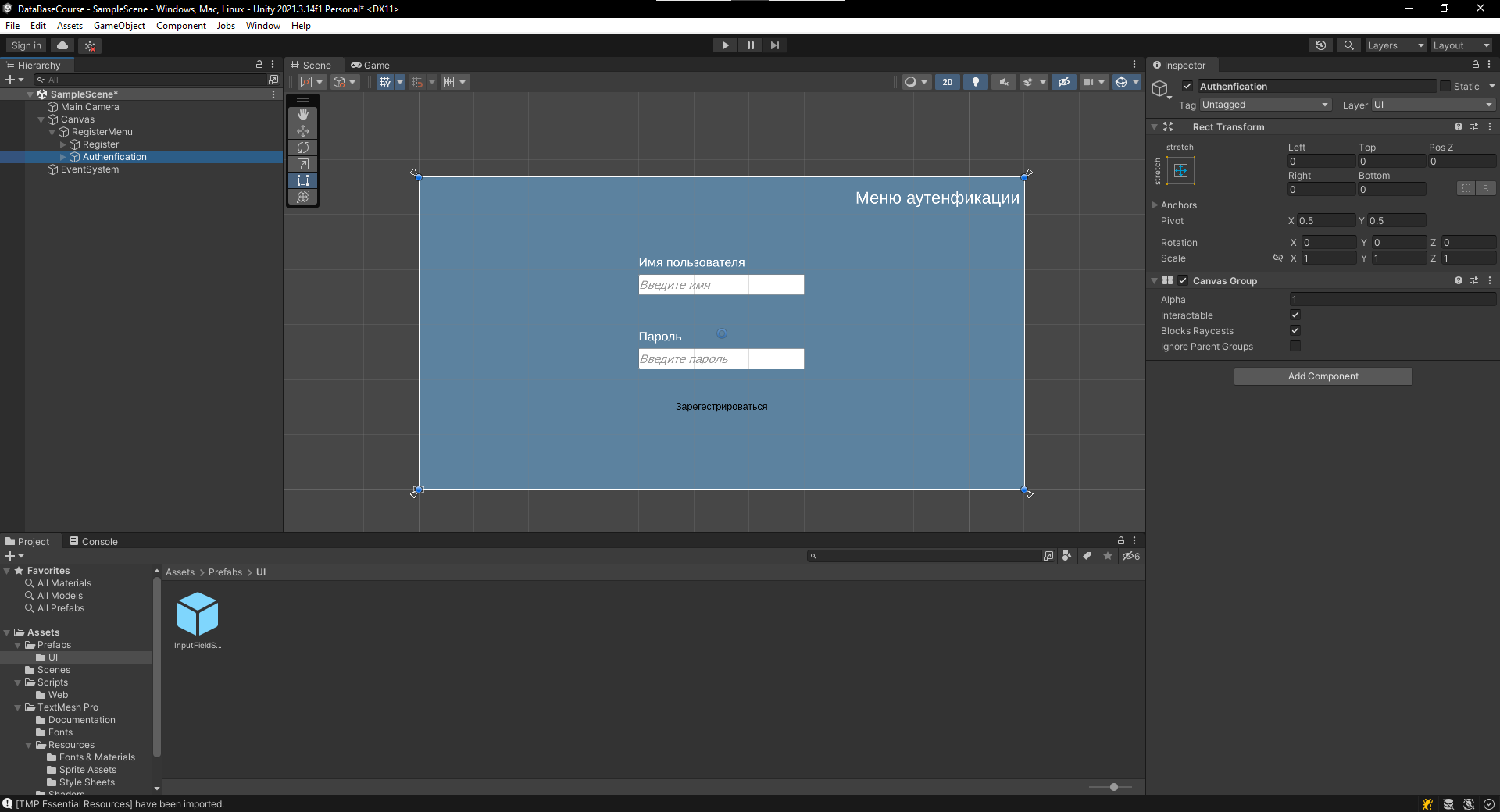
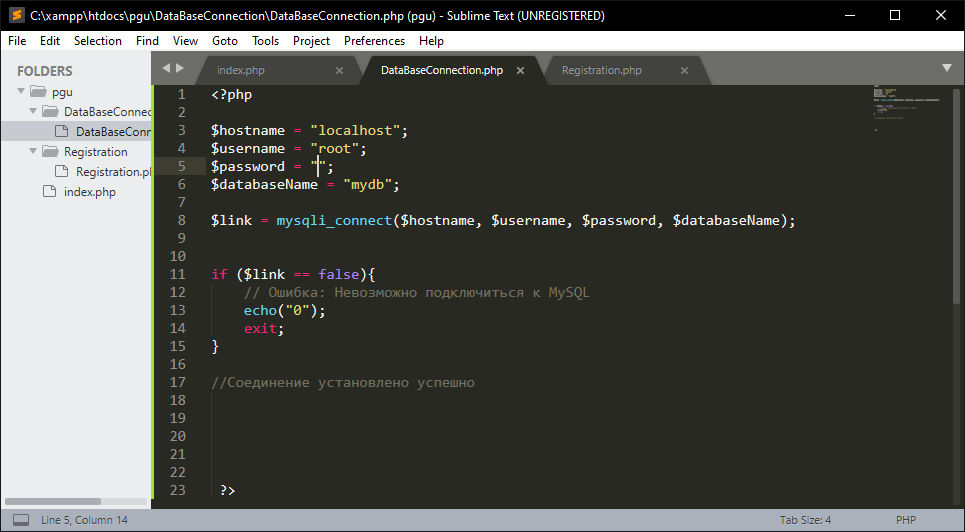


Рисунок 25 – Меню авторизации

Написал быстрый тестовый скрипт для того, чтобы проверить, работает ли передача данных с сервера на приложение.



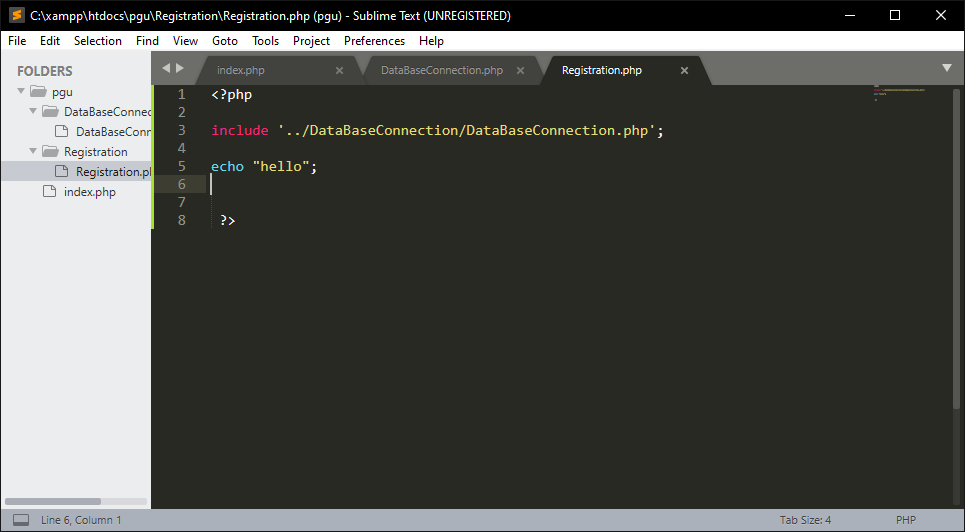


Рисунок 26 – Скрипт проверки подключения

С приложения мы обращаемся по адресу "http://localhost/Registration/Registration.php"   
По этой ссылке выполняется скрипт написанный выше, а он вызывает подскрипт подключения к базе данных, после чего возвращает ответ “hello”

В приложении скрипт работает так, что он запрашивает ответ по ссылке:



Рисунок 27 – Static метод для вызова с верхнего уровня

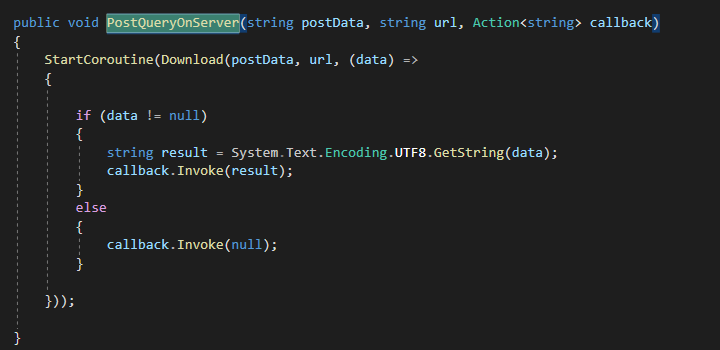


Рисунок 28 – Метод вызова корутины

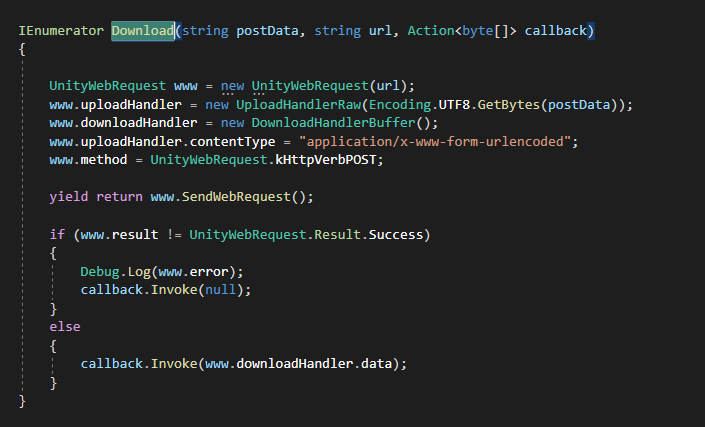


Рисунок 29 – Корутина с запросом на сервер

Был написан класс WebManager. У него паттерн проектирования SINGLETON для того, чтобы я мог вызывать его из любого узла программы с помощью Static методов. Внутри этого класса я добавил класс WebSender который уже отсылает запросы на сервер. Он запускает корутину и ждет ответа, после чего отсылает его обратно в WebManager callback’ом с типом string, для дальнейшей обработки.

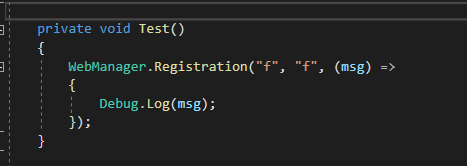


Рисунок 30 – Тестовый метод

А всё вышесказанное вызывает тестовый метод, который выводит результат на консоль движка:

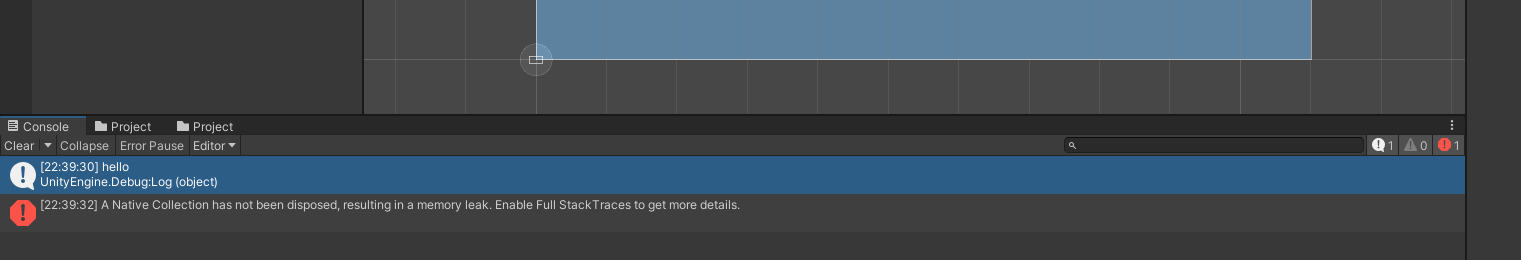


Рисунок 31 – Тестовый ответ от сервера

Всё верно, результат “hello”.

ID POST запроса будет указываться в параметре act=.  
Для регистрации параметр act = 1, для авторизации act = 2, все запросы будут прописаны в таблице позже, какие в них параметры должны быть, и какой ответ будет высылать сервер.

Далее на уровне сервера я написал скрипт авторизации и регистрации:

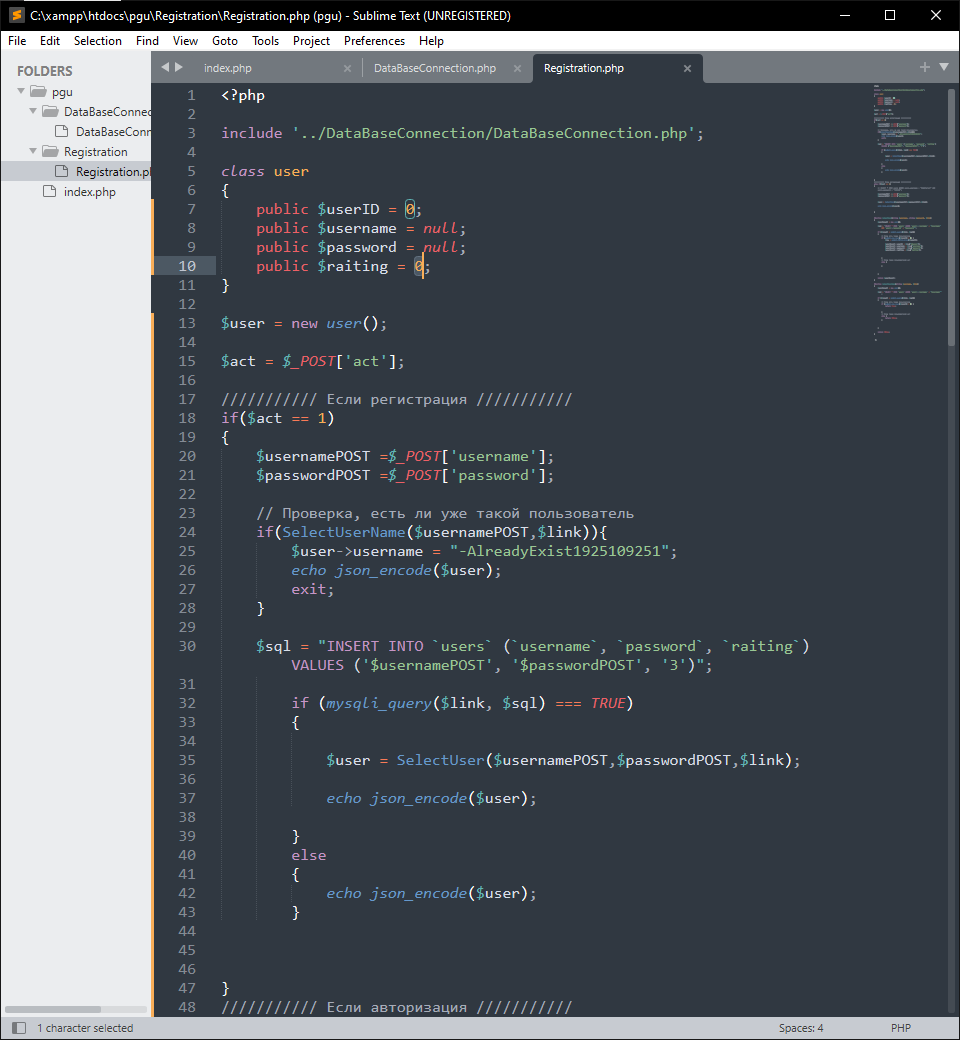


Рисунок 32 – Скрипт регистрации на php

Тут мы отталкиваемся от POST запроса, если act = 1 – то это регистрация.  
Далее нужно проверить существует ли уже пользователь с таким именем.

Если существует, то в имя пользователя мы заносим специальную строку для проверки в приложении.

Если act = 2, то это авторизация и нужно просто вернуть все данные о пользователе

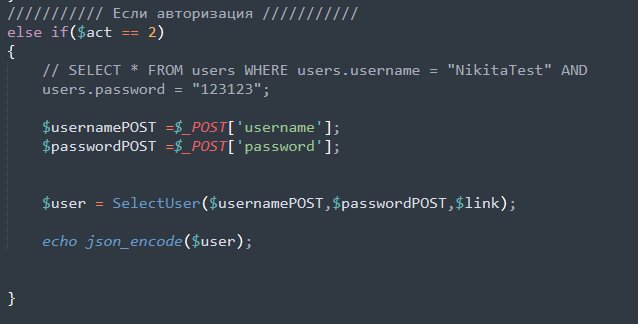


Рисунок 33 – Скрипт php при авторизации



Рисунок 34 – Функция выборки пользователя из базы данных

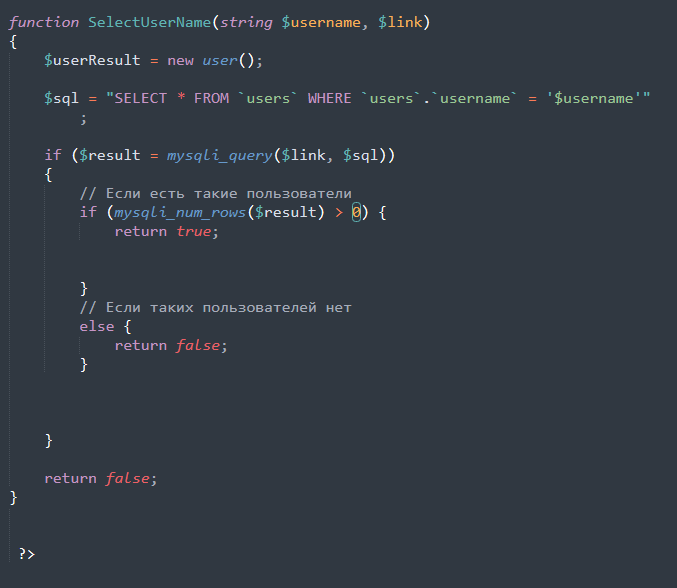


Рисунок 35 – Функция проверки существования пользователя с таким именем на php

Далее создал окно профиля, через которое можно будет создавать новые лабораторные работы, отвечать на вопросы и искать ответы от других пользователей. В Профиле отображается имя пользователя и его рейтинг – они обновляются с помощью static делегата, который вызывается при присвоении нового пользователя.

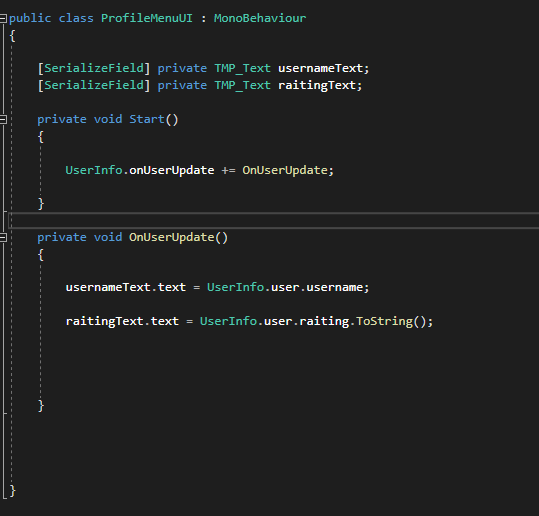
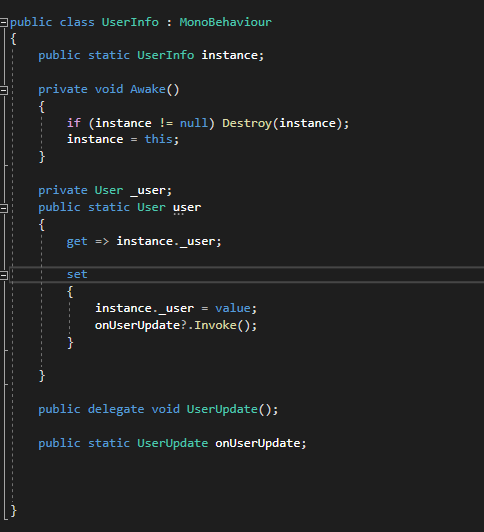


Рисунок 36 – создание объектов на сцене

Далее все меню были созданы по оному образу:



Рисунок 37 – Меню профиля

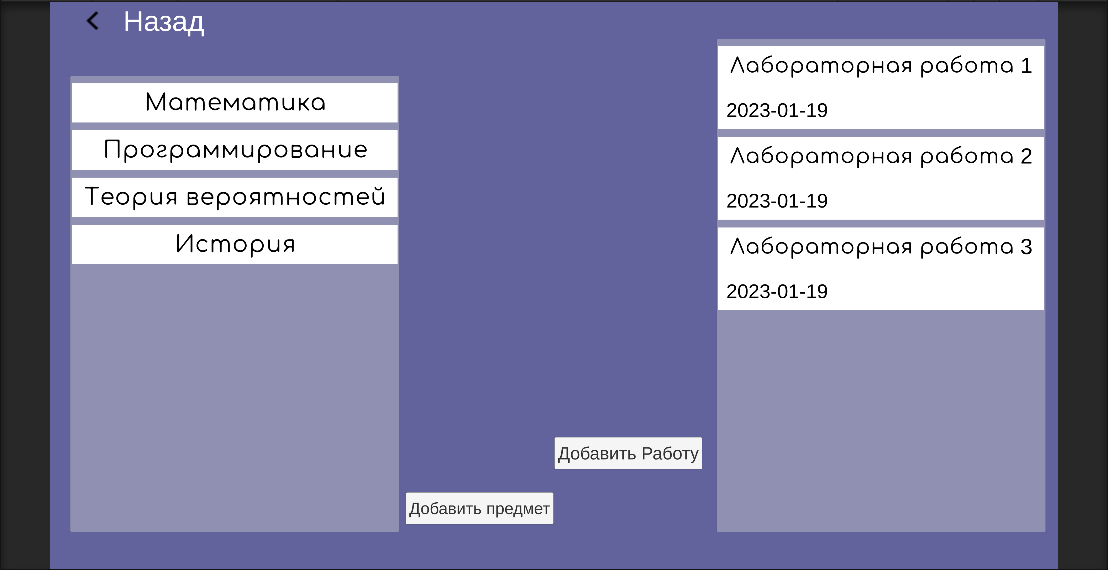


Рисунок 38 – Меню лабораторных работ

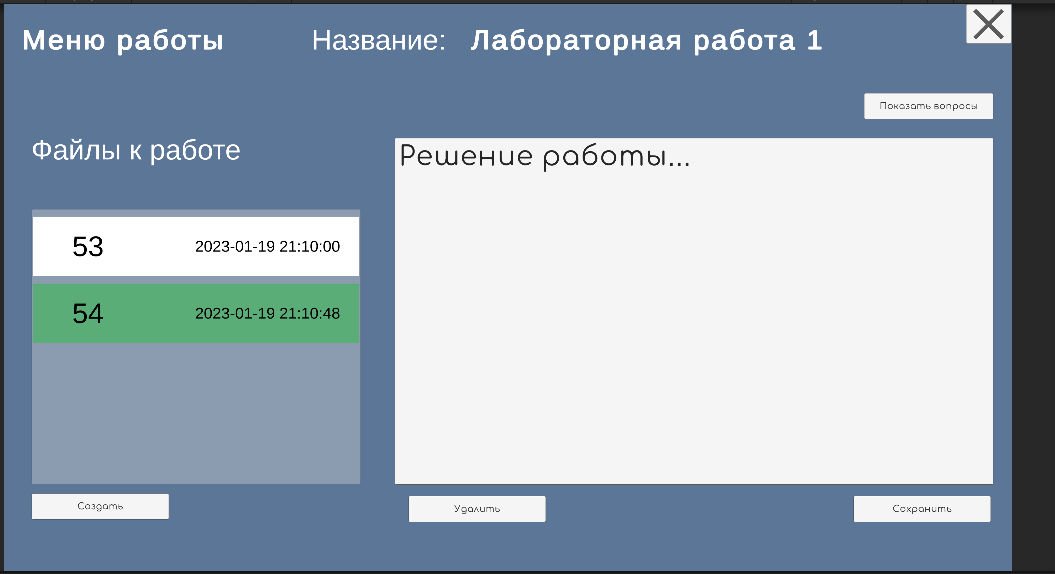


Рисунок 39 – Меню работы

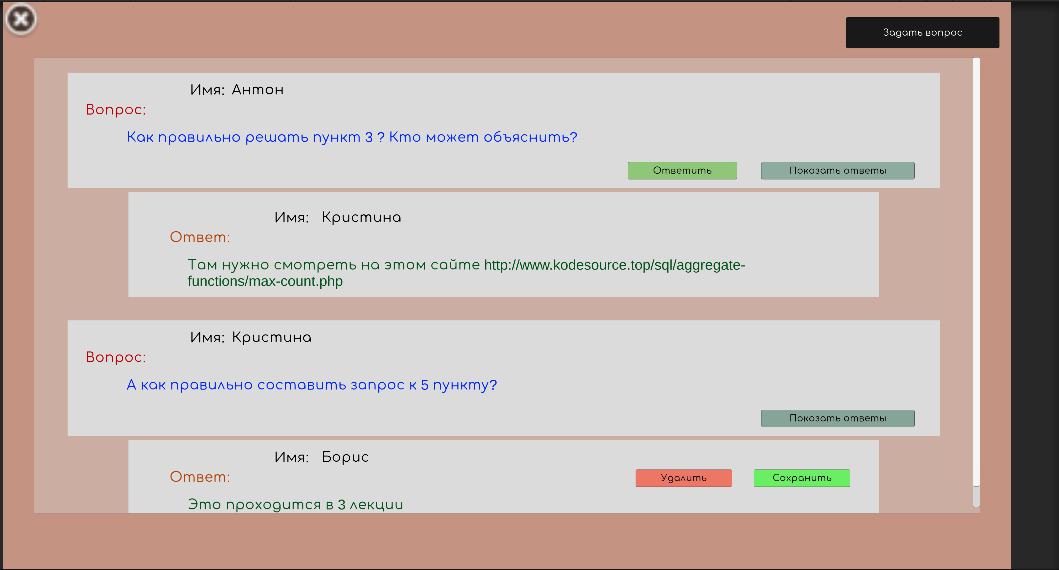


Рисунок 40 – Меню вопросов и ответов

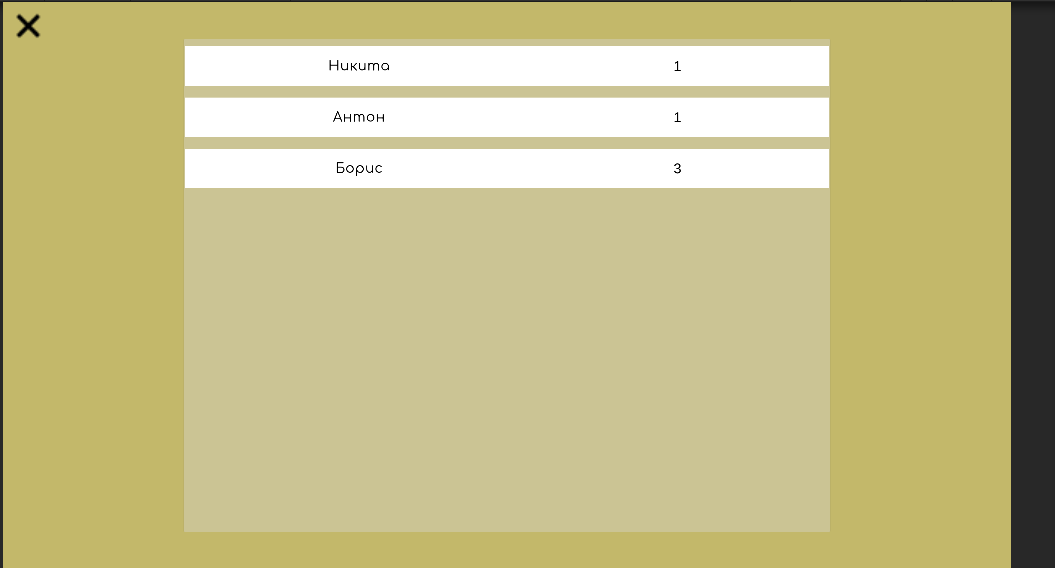


Рисунок 41 – Меню пользователей с кол-вом выложенных файлов

**Запросы на сервер:**

Регистрация и авторизация "http://localhost/Registration/Registration.php"

**Регистрация:**

|  |  |
| --- | --- |
| act | 1 |
| username | Имя пользователя |
| password | Захешированный пароль |

Ответ json:

User:

userID

Username – (“-AlreadyExist1925109251”) – Если это имя занято

Password

Raiting

/////////// Если регистрация ///////////

if($act == 1)

{

$usernamePOST =$\_POST['username'];

$passwordPOST =$\_POST['password'];

// Проверка, есть ли уже такой пользователь

if(SelectUserName($usernamePOST,$link)){

$user->username = "-AlreadyExist1925109251";

echo json\_encode($user);

exit;

}

$sql = "INSERT INTO `users` (`username`, `password`, `raiting`) VALUES ('$usernamePOST', '$passwordPOST', '3')";

if (mysqli\_query($link, $sql) === TRUE)

{

$user = SelectUser($usernamePOST,$passwordPOST,$link);

echo json\_encode($user);

}

else

{

echo json\_encode($user);

}

}

**Авторизация:**

|  |  |
| --- | --- |
| act | 2 |
| username | Имя пользователя |
| password | Захешированный пароль |

Ответ json:

* Если пароль и имя пользователя верно

User:

userID

Username

Password

Raiting

- Если пароль и имя пользователя не верны

User:

userID = 0

Username – null

Password – null

Raiting - 0

/////////// Если авторизация ///////////

else if($act == 2)

{

// SELECT \* FROM users WHERE users.username = "NikitaTest" AND users.password = "123123";

$usernamePOST =$\_POST['username'];

$passwordPOST =$\_POST['password'];

$user = SelectUser($usernamePOST,$passwordPOST,$link);

echo json\_encode($user);

}

Получение и создание Предметов:

"http://localhost/Labs/Subjects.php"

**Получить все предметы**

|  |  |
| --- | --- |
| act | 11 |

Ответ json:

Массив из

subject:

subject\_id

theme\_name

////////// Если нужно вернуть все предметы ///////////

if($act == 11)

{

$sql = "SELECT \* FROM `subject`";

if ($result = mysqli\_query($link, $sql))

{

//echo mysqli\_num\_rows($result);

// Если есть такие пользователи

if (mysqli\_num\_rows($result) > 0) {

$res = array();

//$rows = mysqli\_fetch\_array($result);

while($row = mysqli\_fetch\_assoc($result)){

$newSubject = new subject();

//echo $row['idSubject'];

//echo $row['ThemeName'];

$newSubject->subject\_id = $row['idSubject'];

$newSubject->theme\_name = $row['ThemeName'];

$res[] = $newSubject;

}

$postResult = new ArrayResult();

$postResult->Items = $res;

/\*

foreach ($rows as $row)

{

$newSubject = new subject();

//echo $row['idSubject']

$newSubject->subject\_id = $row['idSubject'];

$newSubject->theme\_name = $row['ThemeName'];

$res->append($newSubject);

}

\*/

echo json\_encode($postResult);

}

// Если таких пользователей нет

else {

}

}

}

**Создать предмет**

|  |  |
| --- | --- |
| act | 12 |
| name |  |

Ответ 1 – если успешно

///////////// Если нужно создать новый предмет ////////////

if($act == 12)

{

$subject\_name = $\_POST['subject\_name'];

$sql = "SELECT \* FROM `subject` WHERE `subject`.`ThemeName` = '$subject\_name'";

if ($result = mysqli\_query($link, $sql))

{

if(mysqli\_num\_rows($result) > 0) {

echo -1;

return;

}

} else {

echo 0;

return;

}

$sql = "INSERT INTO `subject` (`ThemeName`) VALUES ('$subject\_name')";

if ($result = mysqli\_query($link, $sql))

{

echo 1;

} else {

echo 0;

}

}

Получение и создание работ: <http://localhost/Labs/Tasks.php>

Если нужно вернуть все работы по id предмета:

|  |  |
| --- | --- |
| act | 21 |
| subject | Id предмета |

Ответ: объект с Items массивом из объектов task

////////// Если нужно вернуть все работы по id предмета ///////////

if($act == 21)

{

$subject\_idPost = $\_POST['subject\_id'];

$sql = "SELECT \* FROM `task` WHERE `task`.`Subject\_idSubject` = '$subject\_idPost'";

if ($result = mysqli\_query($link, $sql))

{

//echo mysqli\_num\_rows($result);

// Если есть такие Задания есть

if (mysqli\_num\_rows($result) > 0) {

$res = array();

//$rows = mysqli\_fetch\_array($result);

while($row = mysqli\_fetch\_assoc($result)){

$newTask = new task();

//echo $row['idSubject'];

//echo $row['ThemeName'];

$newTask->task\_id = $row['idTask'];

$newTask->task\_name = $row['TaskName'];

$newTask->characteristic = $row['Charactiristic'];

$newTask->subject\_id = $row['Subject\_idSubject'];

$newTask->task\_date = $row['TaskDate'];

$res[] = $newTask;

}

$postResult = new ArrayResult();

$postResult->Items = $res;

echo json\_encode($postResult);

}

// Если таких пользователей нет

else {

}

}

}

Вернуть все работы, в которых есть файлы с id пользователя

|  |  |
| --- | --- |
| act | 23 |
| user\_id |  |

Ответ: объект с Items массивом из объектов task

///////////// вернуть все Tasks где есть файлы с id пользователя

if($act == 23)

{

//SELECT a.\* FROM `task` a INNER JOIN `files` b ON a.idTask = b.Task\_idTask AND b.users\_idusers = 103;

$user\_id = $\_POST['user\_id'];

$sql = "SELECT a.\* FROM `task` a INNER JOIN `files` b ON a.idTask = b.Task\_idTask AND b.users\_idusers = '$user\_id'";

Вернуть все файлы по id работы

|  |  |
| --- | --- |
| act | 24 |
| task\_id |  |

$task\_id = $\_POST['task\_id'];

//$sql = "SELECT a.\* FROM `task` a INNER JOIN `files` b ON a.idTask = b.Task\_idTask AND b.users\_idusers = '$user\_id'";

$sql = "SELECT \* FROM `files` WHERE `files`.`Task\_idTask` = '$task\_id'";

if ($result = mysqli\_query($link, $sql))

Создать новый файл к предмету

|  |  |
| --- | --- |
| act | 25 |
| task\_id |  |
| user\_id |  |

Ответ: объект file

//////////// Создать новый файл /////////////

if($act == 25)

{

////INSERT INTO `files` (`file`, `date`, `comment`, `users\_idusers`, `Task\_idTask`, `id\_file`) VALUES (NULL, NULL, NULL, '110', '5', NULL); SELECT LAST\_INSERT\_ID();

$task\_id = $\_POST['task\_id'];

$user\_id = $\_POST['user\_id'];

$sql = "INSERT INTO `files` (`file`, `date`, `comment`, `users\_idusers`, `Task\_idTask`, `id\_file`) VALUES (NULL, NULL, NULL, '$user\_id', '$task\_id', NULL)";

if ($result = mysqli\_query($link, $sql))

{

// Если запрос удался

if($result == 1)

{

//LAST\_INSERT\_ID()

$sql = "SELECT \* FROM `files` WHERE `files`.`id\_file` = LAST\_INSERT\_ID()";

if ($result = mysqli\_query($link, $sql))

{

Сохранить файл

|  |  |
| --- | --- |
| act | 26 |
| file\_id |  |
| file |  |
| comment |  |

Ответ:

1 – выполнилось

0 – Не выполнилось

/////////////////////////// Если нужно сохранить файл ///////////////////////////

if($act == 26)

{

$file\_id = $\_POST['file\_id'];

$file\_text = $\_POST['file'];

$comment = $\_POST['comment'];

$nowFormat = date('Y-m-d H:i:s');

// UPDATE `users` SET `raiting` = '4' WHERE `users`.`idusers` = 2;

// UPDATE `files` SET `file` = '$file\_text', `date` = '2023-01-1 ', `comment` = '$comment' WHERE `files`.`id\_file` = 2;

$sql = "UPDATE `files` SET `file` = '$file\_text', `date` = '$nowFormat', `comment` = '$comment' WHERE `files`.`id\_file` = '$file\_id'";

if ($result = mysqli\_query($link, $sql))

{

echo 1;

} else {

echo 0;

}

Удалить файл

|  |  |
| --- | --- |
| act | 27 |
| file\_id |  |

/////////////////////////// Если нужно удалить файл ///////////////////////////

if($act == 27)

{

$file\_id = $\_POST['file\_id'];

$sql = "DELETE FROM `files` WHERE `files`.`id\_file` = '$file\_id'";

if ($result = mysqli\_query($link, $sql))

{

echo 1;

} else {

echo 0;

}

}

Получение и создание вопросов/ответов: <http://localhost/Labs/Questions.php>

Вернуть все вопросы по таску

|  |  |
| --- | --- |
| act | 40 |
| task\_id |  |

Ответ: объект с Items массивом из объектов questions

//////////////////////// Если нужно вернуть вопросы по таску ////////////////////////

if($act == 40)

{

$task\_id = $\_POST['task\_id'];

$sql = "SELECT \* FROM `questions`WHERE `questions`.`Task\_idTask` = '$task\_id'";

if ($result = mysqli\_query($link, $sql))

Вернуть ответы по вопросу

|  |  |
| --- | --- |
| act | 41 |
| question\_id |  |

Ответ: объект с Items массивом из объектов answers

////////// Ответы по вопросу ////////////

if($act == 41)

{

$question\_id = $\_POST['question\_id'];

$sql = "SELECT \* FROM `answers`WHERE `answers`.`question\_id` = '$question\_id'";

if ($result = mysqli\_query($link, $sql))

Вернуть имя пользователя

|  |  |
| --- | --- |
| act | 42 |
| user\_id |  |

///////////////////// Вернуть имя пользователя /////////////////////

if($act == 42)

{

$user\_id = $\_POST['user\_id'];

$sql = "SELECT `users`.`username` FROM `users`WHERE `users`.`idusers` = '$user\_id'";

if ($result = mysqli\_query($link, $sql)) {

$row = mysqli\_fetch\_assoc($result);

echo $row['username'];

} else {

echo "undefined";

}

}

Создать вопрос

|  |  |
| --- | --- |
| act | 43 |
| question\_text |  |
| user\_id |  |
| task\_id |  |

////////////////// Создать вопрос

{

$questionText = $\_POST['questionText'];

$Task\_idTask = $\_POST['Task\_idTask'];

$users\_idusers1 = $\_POST['users\_idusers1'];

$sql = "INSERT INTO `questions` (`questionText`, `QuestionID`, `Task\_idTask`, `users\_idusers1`) VALUES ('$questionText', NULL, '$Task\_idTask', '$users\_idusers1');";

if ($result = mysqli\_query($link, $sql))

{

echo 1;

} else {

echo 0;

}

Создать ответ

|  |  |
| --- | --- |
| act | 44 |
| user\_id |  |
| question\_id |  |
| answerText |  |

////////////////// Создать ответ

{

$user\_id = $\_POST['user\_id'];

$question\_id = $\_POST['question\_id'];

$answerText = $\_POST['answerText'];

$sql = "INSERT INTO `answers` (`answerText`, `user\_id`, `question\_id`) VALUES ('$answerText', '$user\_id', '$question\_id')";

if ($result = mysqli\_query($link, $sql))

{

echo 1;

} else {

echo 0;

}

Запрос на имя пользователя и кол-во выложенных файлов

|  |  |
| --- | --- |
| act | 46 |

$sql = "SELECT

`users`.`username`,

COUNT(`files`.`users\_idusers`)

FROM

`files`

JOIN `users` ON `users`.`idusers` = `files`.`users\_idusers`

GROUP BY

`files`.`users\_idusers`";

if ($result = mysqli\_query($link, $sql))

{

**Заключение**

В данной курсовой работе была спроектирована база данных приложения для студентов и разработано WEB приложение обслуживания БД. Инфологическое проектирование было выполнено при помощи CASE средства MySQLWorkbench.

На этапе логического проектирования была выполнена проверка на соответствие нормальным формам. База данных была создана с использованием СУБД MySQL на локальном сервере (пакет XAMPP: Apache/MySQL/PHP и phpMyAdmin).

Были реализованы меню через которые студенты смогут взаимодействовать с базой данных. Приложение имеет возможность создания, удаления, изменения различных строк в базе. Также были созданы запросы на сервер с приложения и ответ с сервера. Все запросы, которые реализованы в приложении были проверены и работают правильно. Все запросы, которые были разработаны имеют документацию, которая написана выше.

Техническое задание на проектирование выполнено в полном объёме.

**Список используемой литературы**

1. Люк Веллинг, Лора Томсон. Разработка веб-приложений с помощью PHP и MySQL. – М.: Вильямс, 2017. – 848 с.
2. А.В. Маркин, С.С. Шкарин. Основы Web-программирования на PHP. – М.: Диалог-МИФИ, 2012. – 256 с.
3. Кристиан Уэнц. PHP и MySQL. Карманный справочник. – М.: Вильямс, 2015. – 256 с.
4. Е.В. Поляков. PHP на примерах. – М.: Наука и техника, 2017.–256с.
5. Денис Колисниченко. PHP и MySQL. Разработка Web-приложений. – СПб.: БХВ-Петербург, 2017. – 640 с.
6. <http://www.kodesource.top/sql/aggregate-functions/max-count.php> (дата обращения: 19.01.2023).
7. <https://metanit.com/sql/mysql/4.8.php> (дата обращения: 18.01.2023).

**Приложение А**

Генерация SQL кода (фрагмент)

-- MySQL Script generated by MySQL Workbench

-- Fri Nov 11 10:05:11 2022

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`users`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`users` (

`idusers` INT NULL AUTO\_INCREMENT,

`username` VARCHAR(45) NULL,

`password` VARCHAR(255) NULL,

`raiting` INT NULL,

PRIMARY KEY (`idusers`))

ENGINE = InnoDB;

CREATE UNIQUE INDEX `idusers\_UNIQUE` ON `mydb`.`users` (`idusers` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `mydb`.`Subject`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Subject` (

`idSubject` INT NOT NULL AUTO\_INCREMENT,

`ThemeName` VARCHAR(255) NULL,

PRIMARY KEY (`idSubject`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Task`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Task` (

`idTask` INT NULL AUTO\_INCREMENT,

`TaskName` VARCHAR(255) NULL,

`Charactiristic` JSON NULL,

`Subject\_idSubject` INT NOT NULL,

PRIMARY KEY (`idTask`),

CONSTRAINT `fk\_subject\_Subject1`

FOREIGN KEY (`Subject\_idSubject`)

REFERENCES `mydb`.`Subject` (`idSubject`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `fk\_subject\_Subject1\_idx` ON `mydb`.`Task` (`Subject\_idSubject` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `mydb`.`files`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`files` (

`file` VARCHAR(255) NULL,

`date` DATETIME NULL,

`comment` VARCHAR(300) NULL,

`filescol` VARCHAR(45) NULL,

`users\_idusers` INT NOT NULL,

`Task\_idTask` INT NOT NULL,

`id\_file` INT NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (`id\_file`),

CONSTRAINT `fk\_files\_users1`

FOREIGN KEY (`users\_idusers`)

REFERENCES `mydb`.`users` (`idusers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_files\_Task1`

FOREIGN KEY (`Task\_idTask`)

REFERENCES `mydb`.`Task` (`idTask`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `fk\_files\_users1\_idx` ON `mydb`.`files` (`users\_idusers` ASC) VISIBLE;

CREATE INDEX `fk\_files\_Task1\_idx` ON `mydb`.`files` (`Task\_idTask` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `mydb`.`questions`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`questions` (

`questionText` VARCHAR(255) NULL,

`QuestionID` INT NOT NULL AUTO\_INCREMENT,

`Task\_idTask` INT NOT NULL,

`users\_idusers1` INT NOT NULL,

PRIMARY KEY (`QuestionID`),

CONSTRAINT `fk\_questions\_Task1`

FOREIGN KEY (`Task\_idTask`)

REFERENCES `mydb`.`Task` (`idTask`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_questions\_users1`

FOREIGN KEY (`users\_idusers1`)

REFERENCES `mydb`.`users` (`idusers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `fk\_questions\_Task1\_idx` ON `mydb`.`questions` (`Task\_idTask` ASC) VISIBLE;

CREATE INDEX `fk\_questions\_users1\_idx` ON `mydb`.`questions` (`users\_idusers1` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `mydb`.`answers`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`answers` (

`answerText` VARCHAR(300) NULL,

`questions\_idquestions` INT NOT NULL,

`questions\_users\_idusers` INT NOT NULL,

`users\_idusers` INT NOT NULL,

`questions\_users\_idusers1` INT NOT NULL,

PRIMARY KEY (`users\_idusers`, `questions\_users\_idusers1`),

CONSTRAINT `fk\_answers\_users1`

FOREIGN KEY (`users\_idusers`)

REFERENCES `mydb`.`users` (`idusers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_answers\_questions1`

FOREIGN KEY (`questions\_users\_idusers1`)

REFERENCES `mydb`.`questions` (`QuestionID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `fk\_answers\_users1\_idx` ON `mydb`.`answers` (`users\_idusers` ASC) VISIBLE;

CREATE INDEX `fk\_answers\_questions1\_idx` ON `mydb`.`answers` (`questions\_users\_idusers1` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `mydb`.`completeData`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`completeData` (

`subject\_idsubject` INT NOT NULL,

`users\_idusers` INT NOT NULL,

`mark` INT NULL,

`date` DATETIME NULL,

`completeData` DATETIME NULL,

PRIMARY KEY (`subject\_idsubject`, `users\_idusers`),

CONSTRAINT `fk\_completeData\_subject1`

FOREIGN KEY (`subject\_idsubject`)

REFERENCES `mydb`.`Task` (`idTask`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_completeData\_users1`

FOREIGN KEY (`users\_idusers`)

REFERENCES `mydb`.`users` (`idusers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `fk\_completeData\_users1\_idx` ON `mydb`.`completeData` (`users\_idusers` ASC) VISIBLE;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

**Приложение Б**

Программные модули

1. Листинг файла DataBaseConnection.php:

<?php

$hostname = "localhost";

$username = "root";

$password = "";

$databaseName = "mydb";

$link = mysqli\_connect($hostname, $username, $password, $databaseName);

if ($link == false){

// Ошибка: Невозможно подключиться к MySQL

echo("0");

exit;

}

//Соединение установлено успешно

?>

1. Листинг файла Tasks.php:

<?php

include '../DataBaseConnection/DataBaseConnection.php';

class task

{

public $task\_id;

public $task\_name;

public $characteristic;

public $subject\_id;

public $task\_date;

}

class file {

public $file;

public $date;

public $comment;

public $users\_idusers;

public $Task\_idTask;

public $id\_file;

}

class ArrayResult{

public $Items;

}

$act = $\_POST['act'];

////////// Если нужно вернуть все работы по id предмета ///////////

if($act == 21)

{

$subject\_idPost = $\_POST['subject\_id'];

$sql = "SELECT \* FROM `task` WHERE `task`.`Subject\_idSubject` = '$subject\_idPost'";

if ($result = mysqli\_query($link, $sql))

{

//echo mysqli\_num\_rows($result);

// Если есть такие Задания есть

if (mysqli\_num\_rows($result) > 0) {

$res = array();

//$rows = mysqli\_fetch\_array($result);

while($row = mysqli\_fetch\_assoc($result)){

$newTask = new task();

//echo $row['idSubject'];

//echo $row['ThemeName'];

$newTask->task\_id = $row['idTask'];

$newTask->task\_name = $row['TaskName'];

$newTask->characteristic = $row['Charactiristic'];

$newTask->subject\_id = $row['Subject\_idSubject'];

$newTask->task\_date = $row['TaskDate'];

$res[] = $newTask;

}

$postResult = new ArrayResult();

$postResult->Items = $res;

echo json\_encode($postResult);

}

// Если таких пользователей нет

else {

}

}

}

else ///////////// Если нужно создать новую лабораторную работу ////////////

if($act == 22)

{

$task\_name = $\_POST['task\_name'];

$Subject\_idSubject = $\_POST['subject\_id'];

$nowFormat = date('Y-m-d H:i:s');

$sql = "INSERT INTO `task` (`TaskName`, `TaskDate`, `Subject\_idSubject`) VALUES ('$task\_name','$nowFormat','$Subject\_idSubject')";

if ($result = mysqli\_query($link, $sql))

{

echo 1;

} else {

echo 0;

}

}

else ///////////// вернуть все Tasks где есть файлы с id пользователя

if($act == 23)

{

//SELECT a.\* FROM `task` a INNER JOIN `files` b ON a.idTask = b.Task\_idTask AND b.users\_idusers = 103;

$user\_id = $\_POST['user\_id'];

$sql = "SELECT a.\* FROM `task` a INNER JOIN `files` b ON a.idTask = b.Task\_idTask AND b.users\_idusers = '$user\_id'";

if ($result = mysqli\_query($link, $sql))

{

//echo mysqli\_num\_rows($result);

//echo mysqli\_num\_rows($result);

// Если есть такие Задания есть

if (mysqli\_num\_rows($result) > 0) {

$res = array();

//$rows = mysqli\_fetch\_array($result);

while($row = mysqli\_fetch\_assoc($result)){

$newTask = new task();

$newTask->task\_id = $row['idTask'];

$newTask->task\_name = $row['TaskName'];

$newTask->characteristic = $row['Charactiristic'];

$newTask->subject\_id = $row['Subject\_idSubject'];

$newTask->task\_date = $row['TaskDate'];

$res[] = $newTask;

}

$postResult = new ArrayResult();

$postResult->Items = $res;

echo json\_encode($postResult);

}

// Если таких пользователей нет

else {

echo null;

}

}

} else //////////// Вернуть все файлы по id работы /////////////

if($act == 24)

{

$task\_id = $\_POST['task\_id'];

//$sql = "SELECT a.\* FROM `task` a INNER JOIN `files` b ON a.idTask = b.Task\_idTask AND b.users\_idusers = '$user\_id'";

$sql = "SELECT \* FROM `files` WHERE `files`.`Task\_idTask` = '$task\_id'";

if ($result = mysqli\_query($link, $sql))

{

if (mysqli\_num\_rows($result) > 0) {

$res = array();

while($row = mysqli\_fetch\_assoc($result)){

$newFile = new file();

$newFile->file = $row['file'];

$newFile->date = $row['date'];

$newFile->comment = $row['comment'];

$newFile->users\_idusers = $row['users\_idusers'];

$newFile->Task\_idTask = $row['Task\_idTask'];

$newFile->id\_file = $row['id\_file'];

$res[] = $newFile;

}

$postResult = new ArrayResult();

$postResult->Items = $res;

echo json\_encode($postResult);

}

// Если таких пользователей нет

else {

echo null;

}

}

} else //////////// Создать новый файл /////////////

if($act == 25)

{

////INSERT INTO `files` (`file`, `date`, `comment`, `users\_idusers`, `Task\_idTask`, `id\_file`) VALUES (NULL, NULL, NULL, '110', '5', NULL); SELECT LAST\_INSERT\_ID();

$task\_id = $\_POST['task\_id'];

$user\_id = $\_POST['user\_id'];

$sql = "INSERT INTO `files` (`file`, `date`, `comment`, `users\_idusers`, `Task\_idTask`, `id\_file`) VALUES (NULL, NULL, NULL, '$user\_id', '$task\_id', NULL)";

if ($result = mysqli\_query($link, $sql))

{

// Если запрос удался

if($result == 1)

{

//LAST\_INSERT\_ID()

$sql = "SELECT \* FROM `files` WHERE `files`.`id\_file` = LAST\_INSERT\_ID()";

if ($result = mysqli\_query($link, $sql))

{

if (mysqli\_num\_rows($result) > 0) {

$res = array();

while($row = mysqli\_fetch\_assoc($result)){

$newFile = new file();

$newFile->file = $row['file'];

$newFile->date = $row['date'];

$newFile->comment = $row['comment'];

$newFile->users\_idusers = $row['users\_idusers'];

$newFile->Task\_idTask = $row['Task\_idTask'];

$newFile->id\_file = $row['id\_file'];

$res[] = $newFile;

}

$postResult = new ArrayResult();

$postResult->Items = $res;

echo json\_encode($postResult);

}

// Если таких пользователей нет

else {

echo null;

}

}

}

}

} else /////////////////////////// Если нужно сохранить файл ///////////////////////////

if($act == 26)

{

$file\_id = $\_POST['file\_id'];

$file\_text = $\_POST['file'];

$comment = $\_POST['comment'];

$nowFormat = date('Y-m-d H:i:s');

// UPDATE `users` SET `raiting` = '4' WHERE `users`.`idusers` = 2;

// UPDATE `files` SET `file` = '$file\_text', `date` = '2023-01-1 ', `comment` = '$comment' WHERE `files`.`id\_file` = 2;

$sql = "UPDATE `files` SET `file` = '$file\_text', `date` = '$nowFormat', `comment` = '$comment' WHERE `files`.`id\_file` = '$file\_id'";

if ($result = mysqli\_query($link, $sql))

{

echo 1;

} else {

echo 0;

}

}

else /////////////////////////// Если нужно удалить файл ///////////////////////////

if($act == 27)

{

$file\_id = $\_POST['file\_id'];

$sql = "DELETE FROM `files` WHERE `files`.`id\_file` = '$file\_id'";

if ($result = mysqli\_query($link, $sql))

{

echo 1;

} else {

echo 0;

}

}

?>

1. Листинг файла Questions.php:

<?php

include '../DataBaseConnection/DataBaseConnection.php';

class Question

{

public $question\_text;

public $question\_id;

public $task\_id;

public $user\_id;

}

class Answer

{

public $answerText;

public $question\_id;

public $user\_id;

}

class ArrayResult

{

public $Items;

}

$act = $\_POST['act'];

//////////////////////// Если нужно вернуть вопросы по таску ////////////////////////

if($act == 40)

{

$task\_id = $\_POST['task\_id'];

$sql = "SELECT \* FROM `questions`WHERE `questions`.`Task\_idTask` = '$task\_id'";

if ($result = mysqli\_query($link, $sql))

{

if (mysqli\_num\_rows($result) > 0) {

$res = array();

while($row = mysqli\_fetch\_assoc($result)){

$newQuestion = new Question();

$newQuestion->question\_text = $row['questionText'];

$newQuestion->question\_id = $row['QuestionID'];

$newQuestion->task\_id = $row['Task\_idTask'];

$newQuestion->user\_id = $row['users\_idusers1'];

$res[] = $newQuestion;

}

$postResult = new ArrayResult();

$postResult->Items = $res;

echo json\_encode($postResult);

}

// Если таких пользователей нет

else {

}

}

}

else ////////// Ответы по вопросу ////////////

if($act == 41)

{

$question\_id = $\_POST['question\_id'];

$sql = "SELECT \* FROM `answers`WHERE `answers`.`question\_id` = '$question\_id'";

if ($result = mysqli\_query($link, $sql))

{

if (mysqli\_num\_rows($result) > 0) {

$res = array();

while($row = mysqli\_fetch\_assoc($result)){

$newAnswer = new Answer();

$newAnswer->answerText = $row['answerText'];

$newAnswer->question\_id = $row['question\_id'];

$newAnswer->user\_id = $row['user\_id'];

$res[] = $newAnswer;

}

$postResult = new ArrayResult();

$postResult->Items = $res;

echo json\_encode($postResult);

}

// Если таких пользователей нет

else {

echo null;

}

}

}

else ///////////////////// Вернуть имя пользователя /////////////////////

if($act == 42)

{

$user\_id = $\_POST['user\_id'];

$sql = "SELECT `users`.`username` FROM `users`WHERE `users`.`idusers` = '$user\_id'";

if ($result = mysqli\_query($link, $sql)) {

$row = mysqli\_fetch\_assoc($result);

echo $row['username'];

} else {

echo "undefined";

}

}

else

if($act == 43) ////////////////// Создать вопрос

{

$questionText = $\_POST['questionText'];

$Task\_idTask = $\_POST['Task\_idTask'];

$users\_idusers1 = $\_POST['users\_idusers1'];

$sql = "INSERT INTO `questions` (`questionText`, `QuestionID`, `Task\_idTask`, `users\_idusers1`) VALUES ('$questionText', NULL, '$Task\_idTask', '$users\_idusers1');";

if ($result = mysqli\_query($link, $sql))

{

echo 1;

} else {

echo 0;

}

}

else

if($act == 44) ////////////////// Создать ответ

{

$user\_id = $\_POST['user\_id'];

$question\_id = $\_POST['question\_id'];

$answerText = $\_POST['answerText'];

$sql = "INSERT INTO `answers` (`answerText`, `user\_id`, `question\_id`) VALUES ('$answerText', '$user\_id', '$question\_id')";

if ($result = mysqli\_query($link, $sql))

{

echo 1;

} else {

echo 0;

}

}

else

if($act == 45) ////////////////// Удалить ответ

{

$user\_id = $\_POST['user\_id'];

$question\_id = $\_POST['question\_id'];

$sql = "DELETE FROM `answers` WHERE `answers`.`user\_id` = '$user\_id' AND `answers`.`question\_id` = '$question\_id'";

if ($result = mysqli\_query($link, $sql))

{

echo 1;

} else {

echo 0;

}

}

else

if($act == 46)

{

$sql = "SELECT

`users`.`username`,

COUNT(`files`.`users\_idusers`)

FROM

`files`

JOIN `users` ON `users`.`idusers` = `files`.`users\_idusers`

GROUP BY

`files`.`users\_idusers`";

if ($result = mysqli\_query($link, $sql))

{

if (mysqli\_num\_rows($result) > 0) {

//$res = array();

$rows = mysqli\_fetch\_all($result);

foreach ($rows as $row) {

$newAnswer = new BigRequestClass();

$newAnswer->username = $row[0];

$newAnswer->countOFFiles = $row[1];

$res[] = $newAnswer;

}

$postResult = new ArrayResult();

$postResult->Items = $res;

echo json\_encode($postResult);

}

// Если таких пользователей нет

else {

echo null;

}

}

}

class BigRequestClass {

public $username;

public $countOFFiles;

}

/\*

SELECT

`users`.`username`,

COUNT(`files`.`users\_idusers`)

FROM

`files`

JOIN `users` ON `users`.`idusers` = `files`.`users\_idusers`

GROUP BY

`files`.`users\_idusers`

"{\

"Items\":[

{\"username\":\"\\u043a\",

\"countOFFiles\":\"41\"},

{\"username\":\"NikitaTest\",\"countOFFiles\":\"3\"}]}"

\*/

?>

1. Листинг файла Subjects.php:

<?php

include '../DataBaseConnection/DataBaseConnection.php';

class subject

{

public $subject\_id;

public $theme\_name;

}

class ArrayResult{

public $Items;

}

$act = $\_POST['act'];

////////// Если нужно вернуть все предметы ///////////

if($act == 11)

{

$sql = "SELECT \* FROM `subject`";

if ($result = mysqli\_query($link, $sql))

{

//echo mysqli\_num\_rows($result);

// Если есть такие пользователи

if (mysqli\_num\_rows($result) > 0) {

$res = array();

//$rows = mysqli\_fetch\_array($result);

while($row = mysqli\_fetch\_assoc($result)){

$newSubject = new subject();

//echo $row['idSubject'];

//echo $row['ThemeName'];

$newSubject->subject\_id = $row['idSubject'];

$newSubject->theme\_name = $row['ThemeName'];

$res[] = $newSubject;

}

$postResult = new ArrayResult();

$postResult->Items = $res;

/\*

foreach ($rows as $row)

{

$newSubject = new subject();

//echo $row['idSubject']

$newSubject->subject\_id = $row['idSubject'];

$newSubject->theme\_name = $row['ThemeName'];

$res->append($newSubject);

}

\*/

echo json\_encode($postResult);

}

// Если таких пользователей нет

else {

}

}

}

else ///////////// Если нужно создать новый предмет ////////////

if($act == 12)

{

$subject\_name = $\_POST['subject\_name'];

$sql = "SELECT \* FROM `subject` WHERE `subject`.`ThemeName` = '$subject\_name'";

if ($result = mysqli\_query($link, $sql))

{

if(mysqli\_num\_rows($result) > 0) {

echo -1;

return;

}

} else {

echo 0;

return;

}

$sql = "INSERT INTO `subject` (`ThemeName`) VALUES ('$subject\_name')";

if ($result = mysqli\_query($link, $sql))

{

echo 1;

} else {

echo 0;

}

}

?>

1. Листинг файла Registration.php:

<?php

include '../DataBaseConnection/DataBaseConnection.php';

class user

{

public $userID = 0;

public $username = null;

public $password = null;

public $raiting = 0;

}

$user = new user();

$act = $\_POST['act'];

/////////// Если регистрация ///////////

if($act == 1)

{

$usernamePOST =$\_POST['username'];

$passwordPOST =$\_POST['password'];

// Проверка, есть ли уже такой пользователь

if(SelectUserName($usernamePOST,$link)){

$user->username = "-AlreadyExist1925109251";

echo json\_encode($user);

exit;

}

$sql = "INSERT INTO `users` (`username`, `password`, `raiting`) VALUES ('$usernamePOST', '$passwordPOST', '3')";

if (mysqli\_query($link, $sql) === TRUE)

{

$user = SelectUser($usernamePOST,$passwordPOST,$link);

echo json\_encode($user);

}

else

{

echo json\_encode($user);

}

}

/////////// Если авторизация ///////////

else if($act == 2)

{

// SELECT \* FROM users WHERE users.username = "NikitaTest" AND users.password = "123123";

$usernamePOST =$\_POST['username'];

$passwordPOST =$\_POST['password'];

$user = SelectUser($usernamePOST,$passwordPOST,$link);

echo json\_encode($user);

}

function SelectUser(string $username, string $password, $link)

{

$userResult = new user();

$sql = "SELECT \* FROM `users` WHERE `users`.`username` = '$username' AND `users`.`password` = '$password'";

if ($result = mysqli\_query($link, $sql))

{

// Если есть такие пользователи

if (mysqli\_num\_rows($result) > 0) {

$row = mysqli\_fetch\_array($result);

$userResult->userID = $row['idusers'];

$userResult->username = $row['username'];

$userResult->password = $row['password'];

$userResult->raiting = $row['raiting'];

}

// Если таких пользователей нет

else {

}

}

return $userResult;

}

function SelectUserName(string $username, $link)

{

$userResult = new user();

$sql = "SELECT \* FROM `users` WHERE `users`.`username` = '$username'";

if ($result = mysqli\_query($link, $sql))

{

// Если есть такие пользователи

if (mysqli\_num\_rows($result) > 0) {

return true;

}

// Если таких пользователей нет

else {

return false;

}

}

return false;

}

?>

1. Листинг файла WebManager.CS

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class WebManager : MonoBehaviour

{

public static WebManager instance;

[SerializeField] private WebSender webSender;

private void Awake()

{

if (instance != null) Destroy(instance);

instance = this;

}

#region registration

//////////////////////////////////////

///// R E G I S T R A T I O N /////

//////////////////////////////////////

private const string REGISTRATION\_ACT = "1";

private const string REGISTRATION\_URL = "http://localhost/Registration/Registration.php";

public static void Registration(string name, string pass, Action<string> callback)

{

string post = $"act={REGISTRATION\_ACT}&username={name}&password={pass}";

instance.webSender.PostQueryOnServer(post, REGISTRATION\_URL, callback);

}

////////////////////////////////////////

///// AUTHENTICATION ////

////////////////////////////////////////

private const string AUTHENTICATION\_ACT = "2";

private const string AUTHENTICATION\_URL = "http://localhost/Registration/Registration.php";

public static void Authentication(string name, string pass, Action<string> callback)

{

string post = $"act={AUTHENTICATION\_ACT}&username={name}&password={pass}";

instance.webSender.PostQueryOnServer(post, AUTHENTICATION\_URL, callback);

}

#endregion

#region subjects

////////////////////////////////////////

///// S U B J E C T S ////

////////////////////////////////////////

private const string SUBJECTS\_URL = "http://localhost/Labs/Subjects.php";

private const string GET\_SUBJECTS\_ACT = "11";

public static void GetAllSubjects(Action<string> callback)

{

string post = $"act={GET\_SUBJECTS\_ACT}";

instance.webSender.PostQueryOnServer(post, SUBJECTS\_URL, callback);

}

private const string CREATE\_SUBJECTS\_ACT = "12";

public static void CreateNewSubject(string subject\_name, Action<string> callback)

{

//$subject\_name={subject\_name}

string post = $"act={CREATE\_SUBJECTS\_ACT}&subject\_name={subject\_name}";

instance.webSender.PostQueryOnServer(post, SUBJECTS\_URL, callback);

}

#endregion

#region Tasks

////////////////////////////////////////

///// T A S K S ////

////////////////////////////////////////

private const string TASK\_URL = "http://localhost/Labs/Tasks.php";

private const string GET\_TASKS\_ACT = "21";

public static void GetTasksBySubject(int subject\_id, Action<string> callback)

{

string post = $"act={GET\_TASKS\_ACT}&subject\_id={subject\_id}";

instance.webSender.PostQueryOnServer(post, TASK\_URL, callback);

}

private const string GET\_TASKS\_USER\_ID\_ACT = "23";

public static void GetTasksByUserID(int user\_id, Action<string> callback)

{

string post = $"act={GET\_TASKS\_USER\_ID\_ACT}&user\_id={user\_id}";

instance.webSender.PostQueryOnServer(post, TASK\_URL, callback);

}

private const string CREATE\_TASK\_ACT = "22";

public static void CreateNewTask(string task\_name, int subject\_id, Action<string> callback)

{

//$subject\_name={subject\_name}

string post = $"act={CREATE\_TASK\_ACT}&task\_name={task\_name}&subject\_id={subject\_id}";

instance.webSender.PostQueryOnServer(post, TASK\_URL, callback);

}

#endregion

#region Files

private const string GET\_FILES\_BY\_TASK\_ACT = "24";

public static void GetFilesByTaskID(int task\_id, Action<string> callback)

{

string post = $"act={GET\_FILES\_BY\_TASK\_ACT}&task\_id={task\_id}";

instance.webSender.PostQueryOnServer(post, TASK\_URL, callback);

}

private const string CREATE\_FILE\_ACT = "25";

public static void CreateNewFile(int task\_id, int user\_id, Action<string> callback)

{

string post = $"act={CREATE\_FILE\_ACT}&task\_id={task\_id}&user\_id={user\_id}";

instance.webSender.PostQueryOnServer(post, TASK\_URL, callback);

}

private const string SAVE\_FILE\_ACT = "26";

public static void SaveFile(int file\_id, string file\_text, string comment ,Action<string> callback)

{

string post = $"act={SAVE\_FILE\_ACT}&file\_id={file\_id}&file={file\_text}&comment={comment}";

instance.webSender.PostQueryOnServer(post, TASK\_URL, callback);

}

private const string DELETE\_FILE\_ACT = "27";

public static void DeleteFile(int file\_id, Action<string> callback)

{

string post = $"act={DELETE\_FILE\_ACT}&file\_id={file\_id}";

instance.webSender.PostQueryOnServer(post, TASK\_URL, callback);

}

#endregion

#region Answers & Questions

private const string QUESTIONS\_URL = "http://localhost/Labs/Questions.php";

private const string QUESTIONS\_BY\_TASK\_ACT = "40";

public static void GetQuestionsByTaskID(int task\_id, Action<string> callback)

{

string post = $"act={QUESTIONS\_BY\_TASK\_ACT}&task\_id={task\_id}";

instance.webSender.PostQueryOnServer(post, QUESTIONS\_URL, callback);

}

private const string ANSWERS\_BY\_QUESTION\_ACT= "41";

public static void GetAnswersByQuestionID(int question\_id, Action<string> callback)

{

string post = $"act={ANSWERS\_BY\_QUESTION\_ACT}&question\_id={question\_id}";

instance.webSender.PostQueryOnServer(post, QUESTIONS\_URL, callback);

}

private const string CREATE\_QUESTION\_ACT = "43";

public static void CreateQuestion(string questionText, int Task\_idTask , int users\_idusers1, Action<string> callback)

{

string post = $"act={CREATE\_QUESTION\_ACT}&questionText={questionText}&Task\_idTask={Task\_idTask}&users\_idusers1={users\_idusers1}";

instance.webSender.PostQueryOnServer(post, QUESTIONS\_URL, callback);

}

private const string CREATE\_ANSWER\_ACT = "44";

public static void CreateAnswer(string answerText, int question\_id, int user\_id, Action<string> callback)

{

string post = $"act={CREATE\_ANSWER\_ACT}&answerText={answerText}&question\_id={question\_id}&user\_id={user\_id}";

instance.webSender.PostQueryOnServer(post, QUESTIONS\_URL, callback);

}

private const string DELETE\_ANSWER\_ACT = "45";

public static void DeleteAnswer(int question\_id, int user\_id, Action<string> callback)

{

string post = $"act={DELETE\_ANSWER\_ACT}&user\_id={user\_id}&question\_id={question\_id}";

instance.webSender.PostQueryOnServer(post, QUESTIONS\_URL, callback);

}

#endregion

private const string GET\_BEST\_USERS\_ACT = "46";

public static void GetBestUsers(Action<string> callback)

{

string post = $"act={GET\_BEST\_USERS\_ACT}";

instance.webSender.PostQueryOnServer(post, QUESTIONS\_URL, callback);

}

#region USERS

private const string USERNAME\_BY\_QUESTION\_ACT = "42";

public static void GetUsernameByUserID(int user\_id, Action<string> callback)

{

string post = $"act={USERNAME\_BY\_QUESTION\_ACT}&user\_id={user\_id}";

instance.webSender.PostQueryOnServer(post, QUESTIONS\_URL, callback);

}

#endregion

}

1. Листинг файла WebSender.CS

using System;

using System.Collections;

using System.Collections.Generic;

using System.IO;

using System.Net;

using System.Text;

using UnityEngine;

using UnityEngine.Networking;

public class WebSender : MonoBehaviour

{

[SerializeField] private string[] urlsToCheckConnection;

[SerializeField] private string serverURL = "http://localhost/";

public void PostQueryOnServer(string postData, string url, Action<string> callback)

{

StartCoroutine(Download(postData, url, (data) =>

{

if (data != null)

{

string result = System.Text.Encoding.UTF8.GetString(data);

//Debug.Log(result);

callback.Invoke(result);

}

else

{

callback.Invoke(null);

}

}));

}

public void GetFile(string url, Action<byte[]> callback)

{

StartCoroutine(DownloadFile(url, (data) =>

{

if (data != null)

{

//string result = System.Text.Encoding.UTF8.GetString(data);

callback.Invoke(data);

}

else

{

callback.Invoke(null);

}

}));

}

IEnumerator Download(string postData, string url, Action<byte[]> callback)

{

UnityWebRequest www = new UnityWebRequest(url);

www.uploadHandler = new UploadHandlerRaw(Encoding.UTF8.GetBytes(postData));

www.downloadHandler = new DownloadHandlerBuffer();

www.uploadHandler.contentType = "application/x-www-form-urlencoded";

www.method = UnityWebRequest.kHttpVerbPOST;

yield return www.SendWebRequest();

if (www.result != UnityWebRequest.Result.Success)

{

//Debug.Log(www.error);

callback.Invoke(null);

}

else

{

callback.Invoke(www.downloadHandler.data);

}

www.Dispose();

}

IEnumerator DownloadFile(string url, Action<byte[]> callback)

{

UnityWebRequest www = new UnityWebRequest(url);

www.uploadHandler = new UploadHandlerRaw(Encoding.UTF8.GetBytes(url));

www.downloadHandler = new DownloadHandlerBuffer();

www.uploadHandler.contentType = "application/x-www-form-urlencoded";

www.method = UnityWebRequest.kHttpVerbGET;

yield return www.SendWebRequest();

if (www.result != UnityWebRequest.Result.Success)

{

//Debug.Log(www.error);

callback.Invoke(null);

}

else

{

callback.Invoke(www.downloadHandler.data);

}

www.Dispose();

}

public void CheckInternetConnection(Action<ConnectionResult> callback)

{

StartCoroutine(CheckInternetConnectionCoroutine(callback));

}

IEnumerator CheckInternetConnectionCoroutine(Action<ConnectionResult> callback)

{

{

UnityWebRequest www = UnityWebRequest.Get(serverURL);

yield return www.SendWebRequest();

if (www.result == UnityWebRequest.Result.Success)

{

callback.Invoke(ConnectionResult.success);

yield break;

}

}

foreach (var url in urlsToCheckConnection)

{

UnityWebRequest www = UnityWebRequest.Get(url);

yield return www.SendWebRequest();

if (www.result == UnityWebRequest.Result.Success)

{

callback.Invoke(ConnectionResult.ServerCrushed);

yield break;

}

}

callback.Invoke(ConnectionResult.noInternet);

}

public enum ConnectionResult

{

noInternet,

ServerCrushed,

success,

}

}