**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:

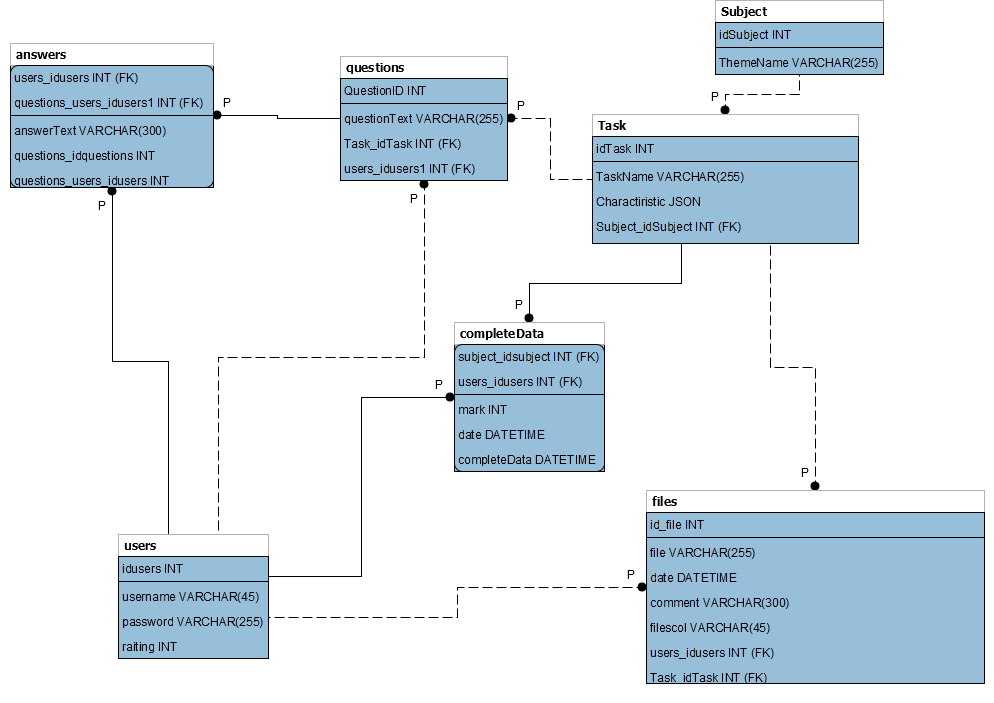


Рисунок - 1

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

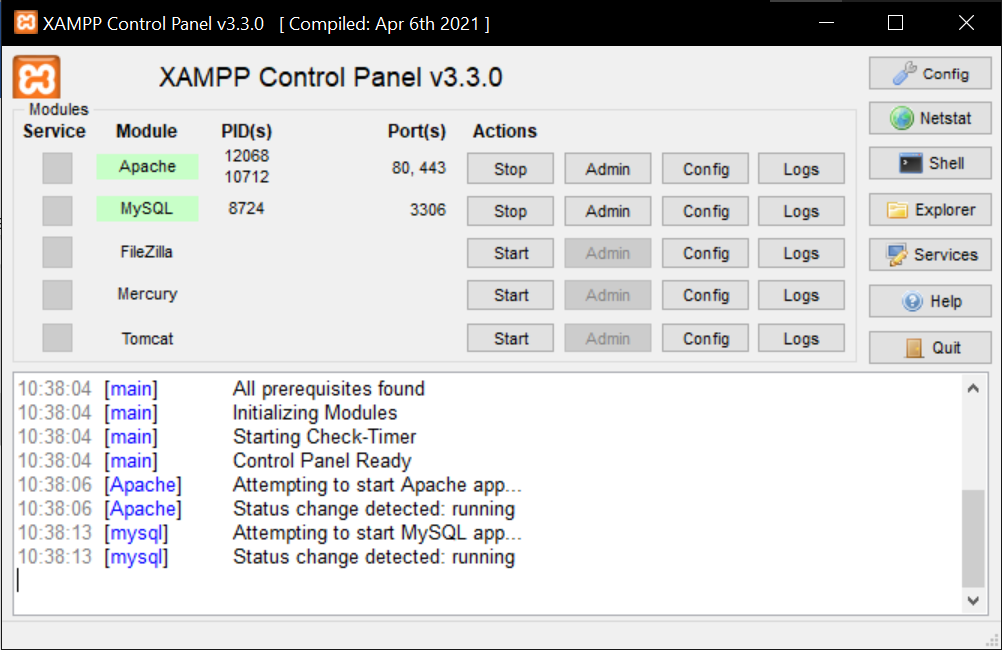


Рисунок 2 – Xammp панель

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

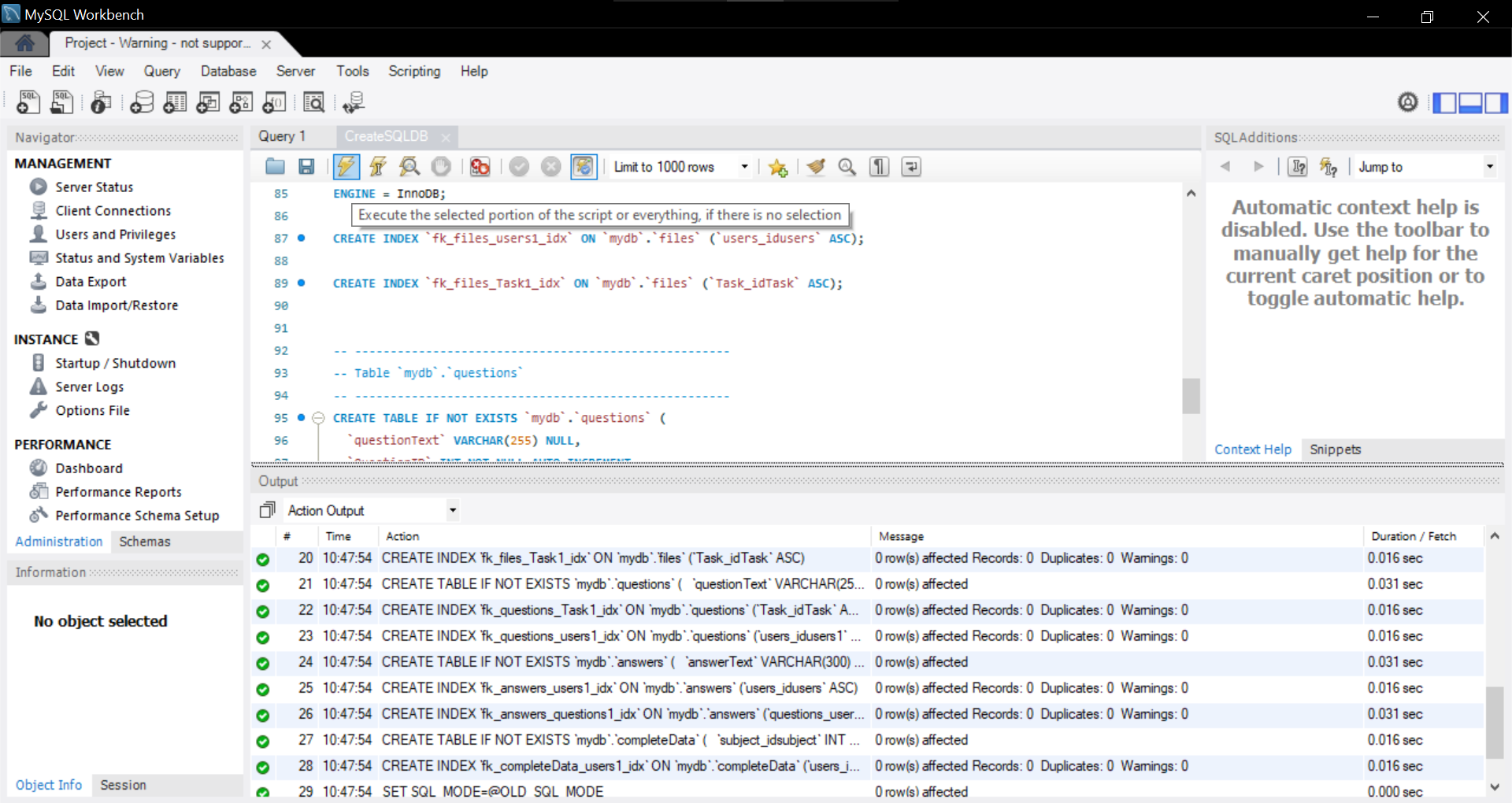


Рисунок – 3 Скрипт в MySQL Workbench

Зашли в PHPMYADMIN чтобы убедиться, что таблица создалась.

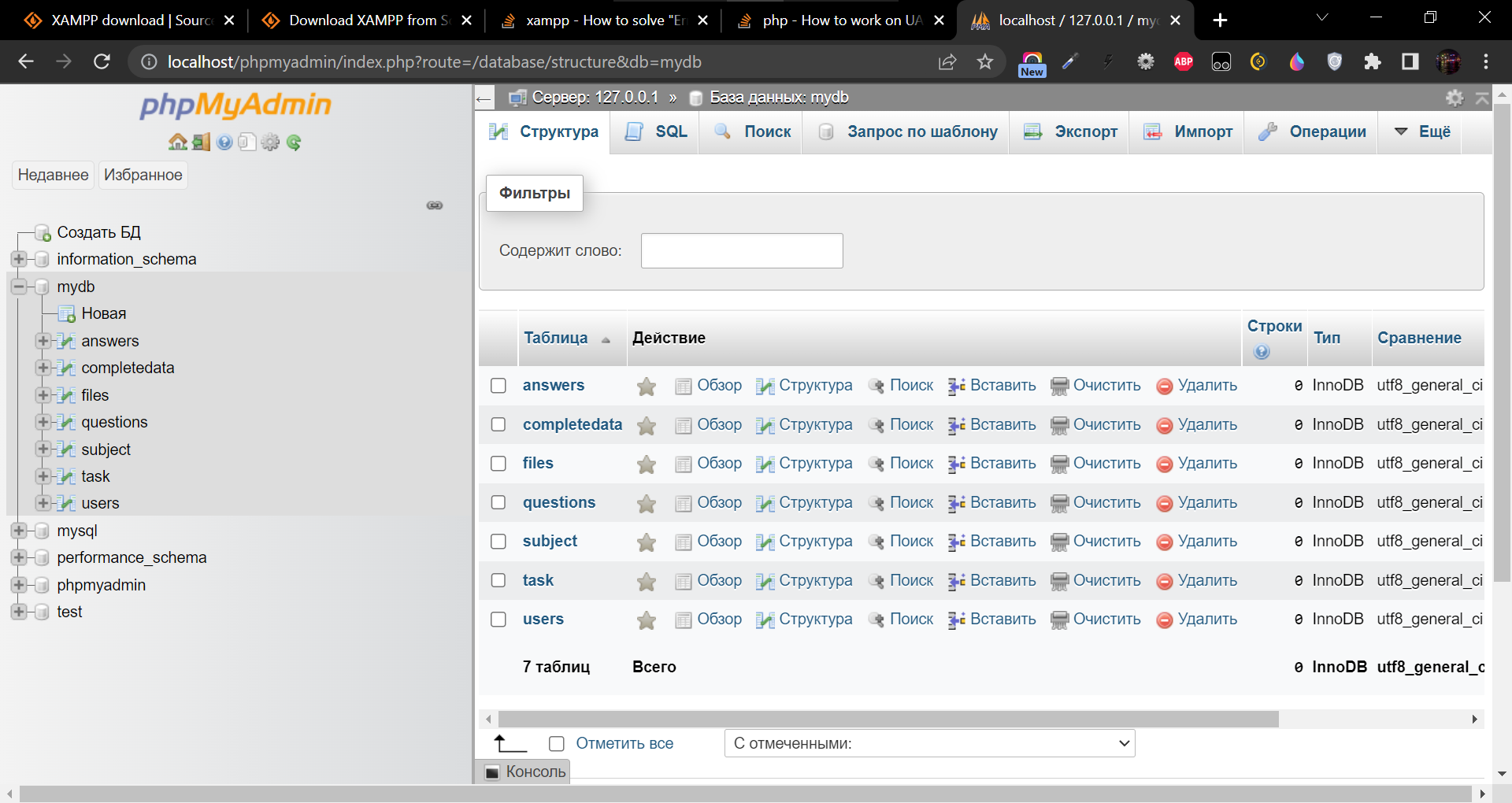


Рисунок 4 – PHPMyAdmin наша база данных

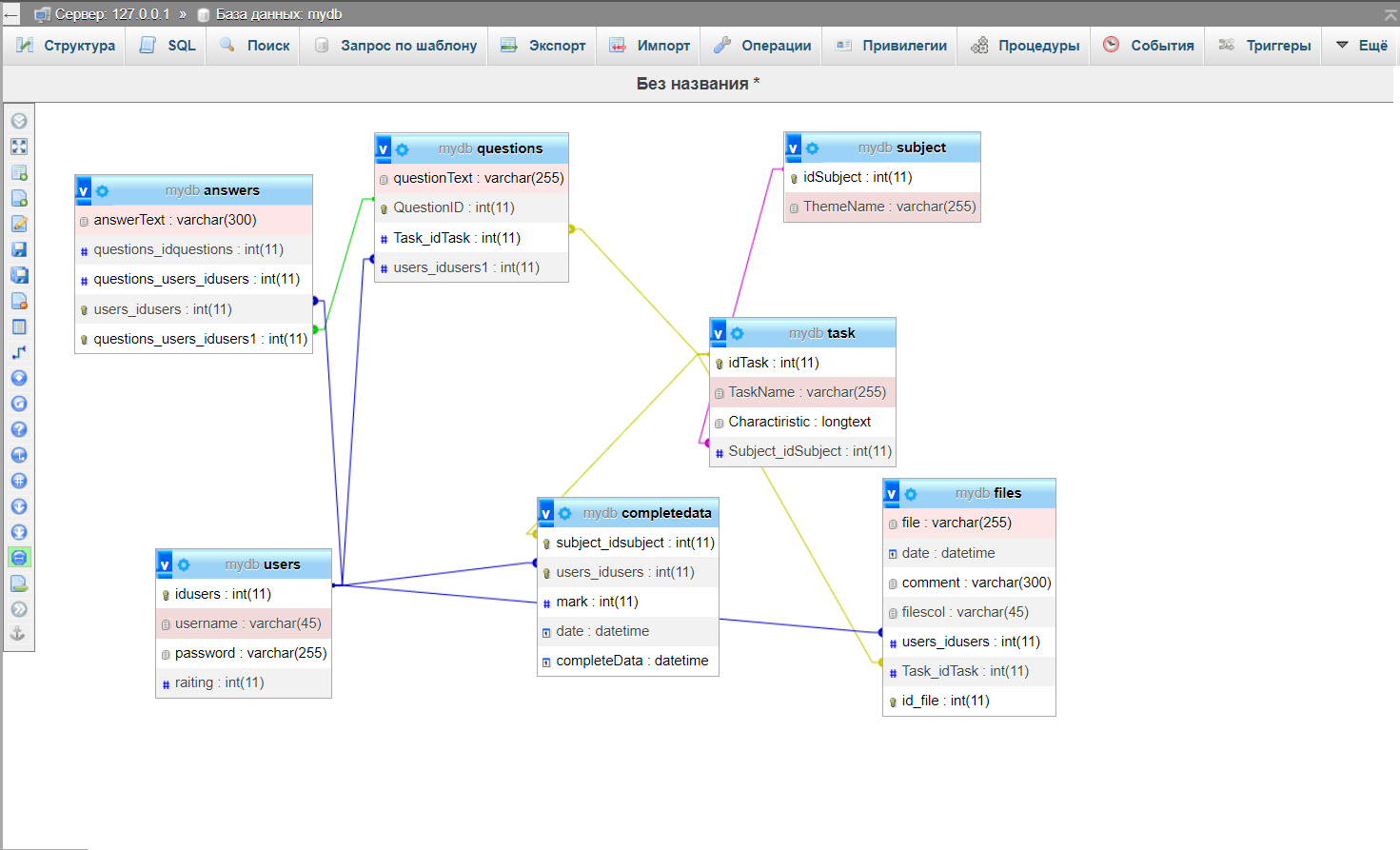


Рисунок 5 – Схема базы в PHPMyAdmin

**Соответствие нормальным формам**

**Первая нормальная форма**

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

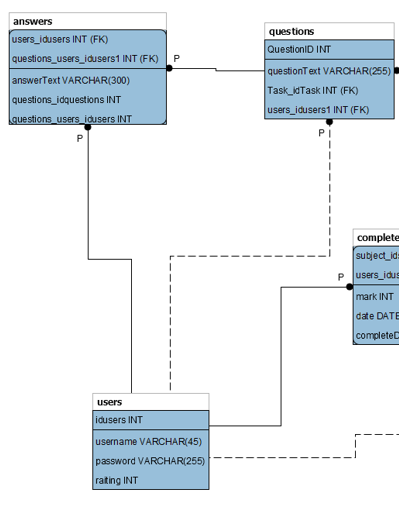
****

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).

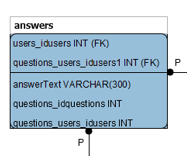
****

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).

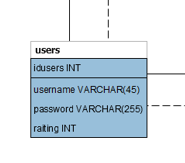
****

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.

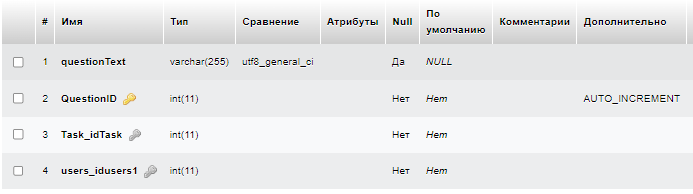
****

Рисунок 12 – Типы данных в таблице questions

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

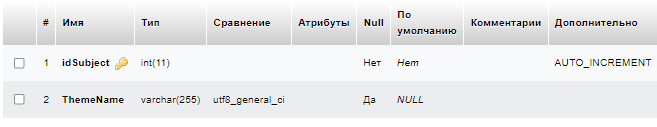
****

Рисунок 13 – Типы данных в таблице subject

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

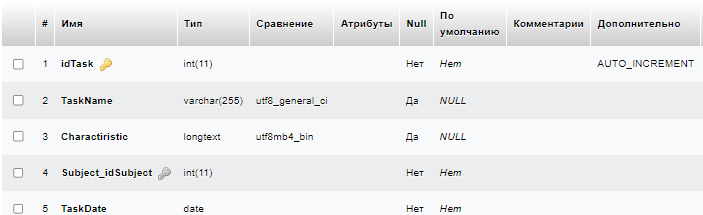
****

Рисунок 14 – Типы данных в таблице task

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

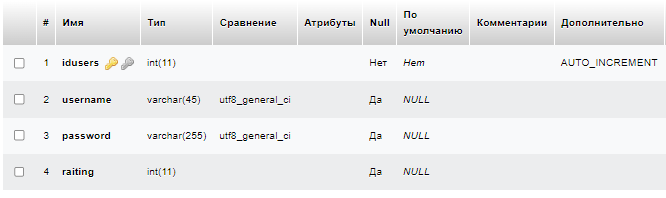
****

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 17

****

Рисунок 17 – 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 18

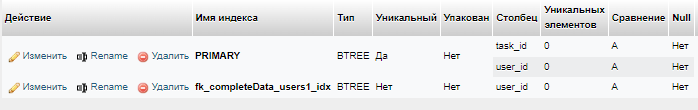
****

Рисунок 18 – 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 19

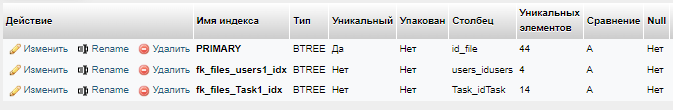
****

Рисунок 19 – 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 20

****

Рисунок 20 – 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 рисунке21

****

Рисунок 21 – Индексы в таблице subject

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

1. Индексы в таблице task изображены на рисунке 22

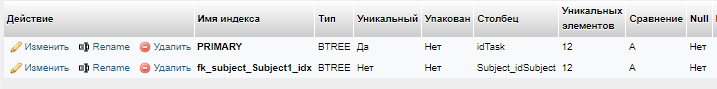
****

Рисунок 22 – 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 23

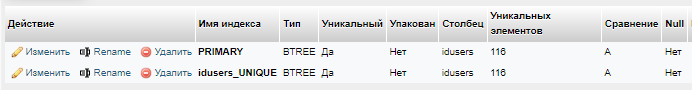
****

Рисунок 23 – Индексы в таблице users

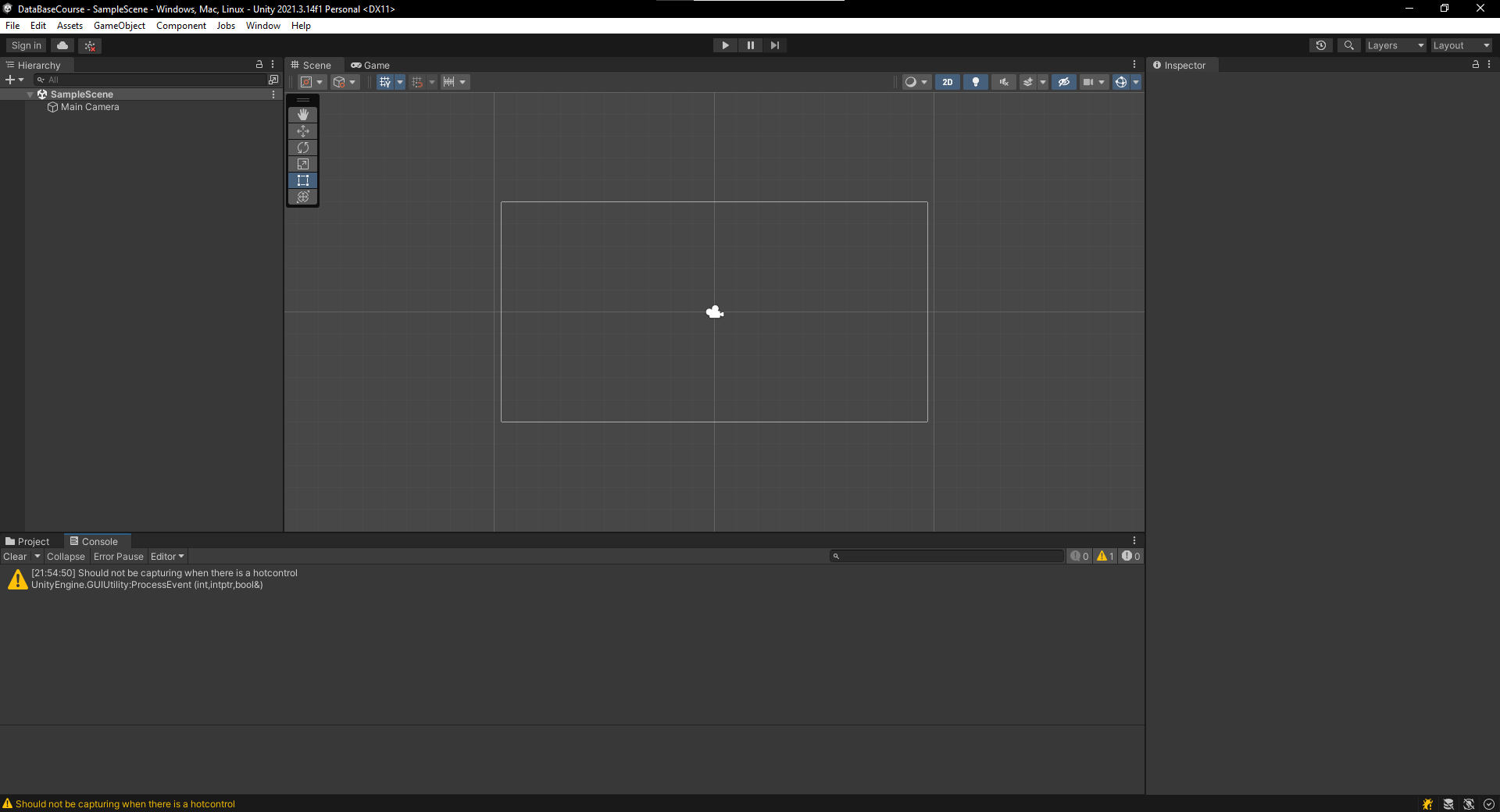
**PRIMARY** – первичный ключ столбец id users;

**idusers\_UNIQUE –** уникальныйидентификатор;

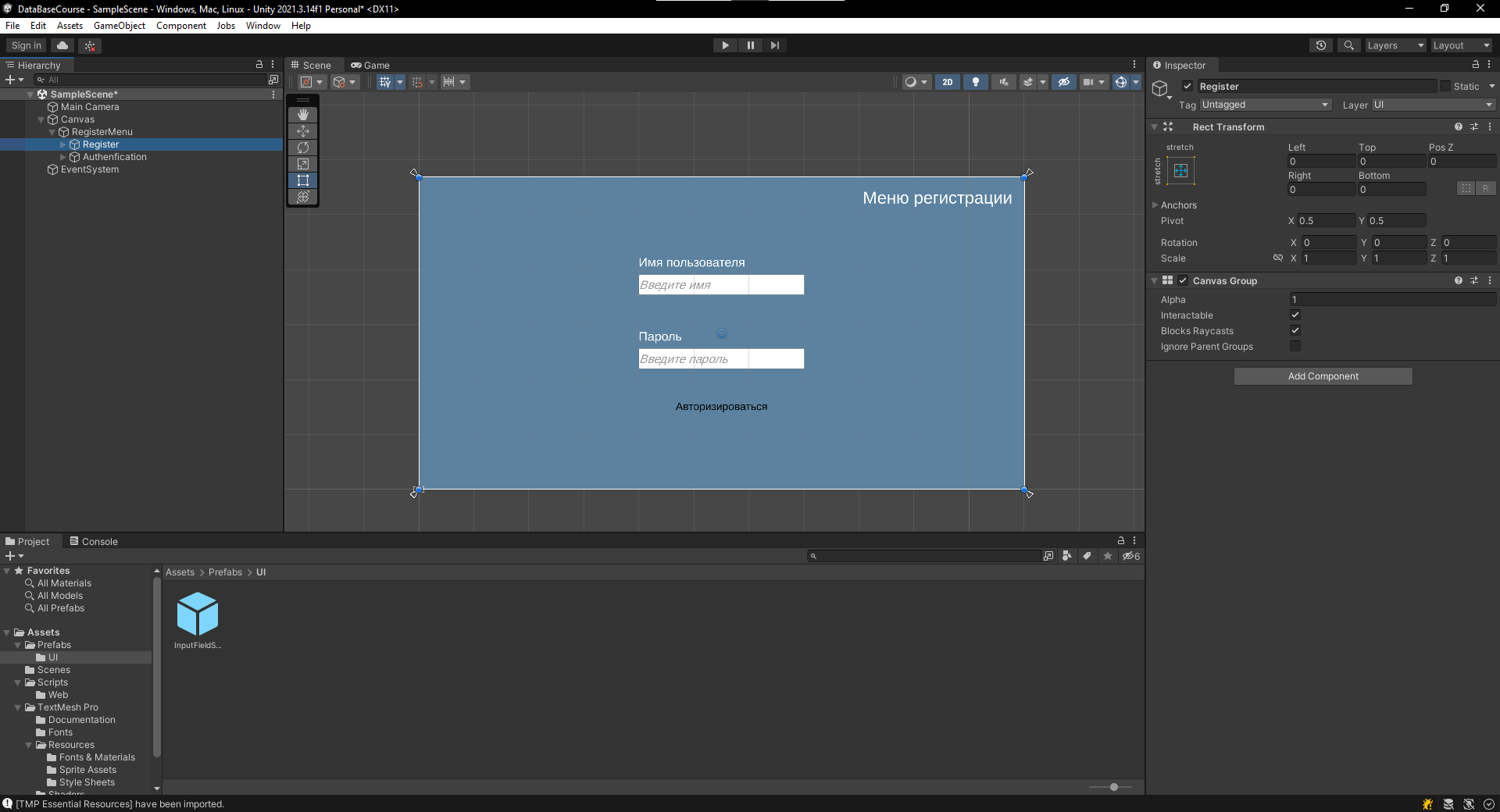
Первым делом нужно спланировать, как будет связываться приложение с базой данных.   
Я решил, что лучше всего связывать приложение будет с помощью POST запросов.

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

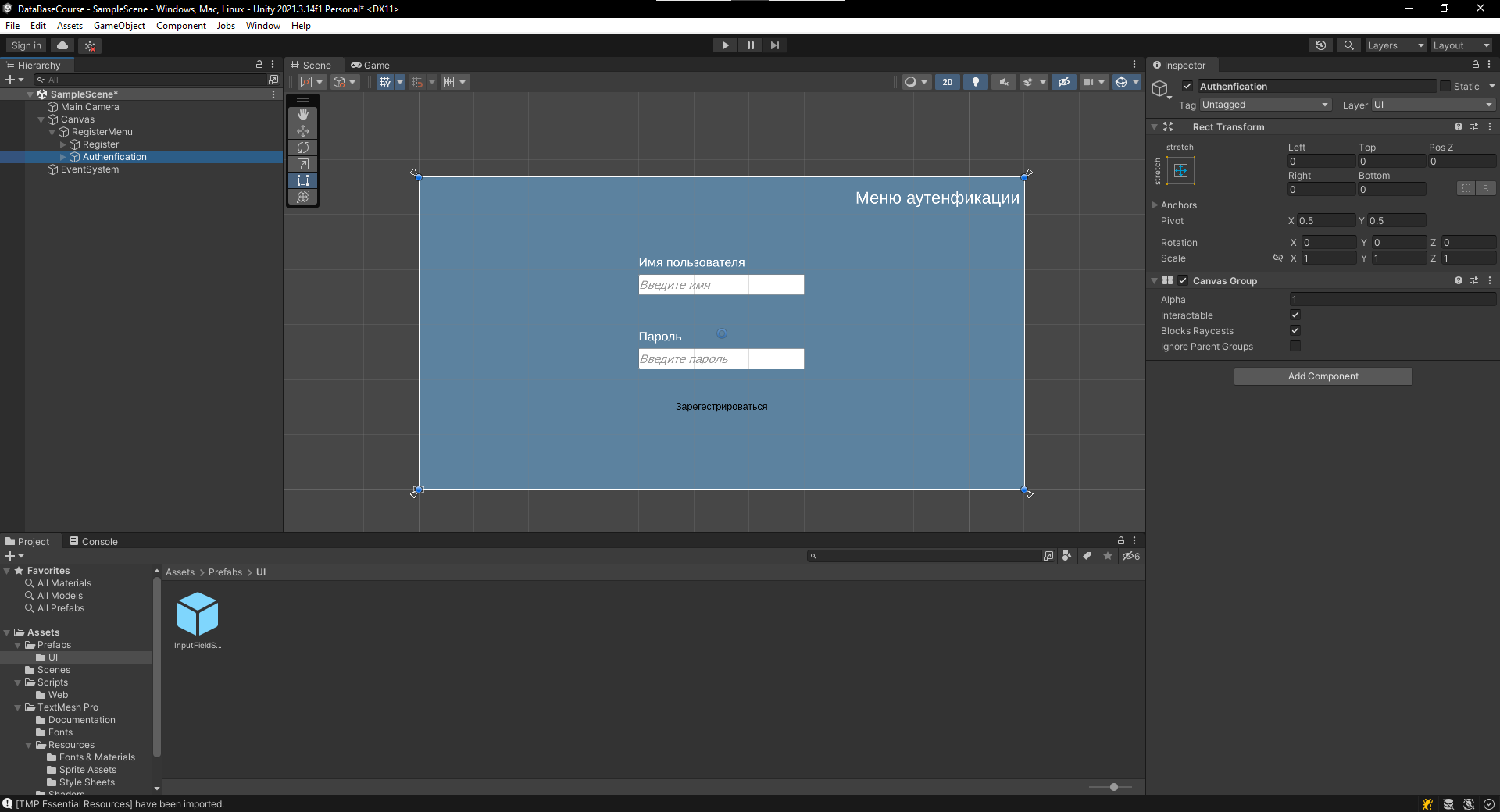
Далее создадим новый проект на Unity и сверстаем UI меню для регистрации или входа пользователя.



Новый проект

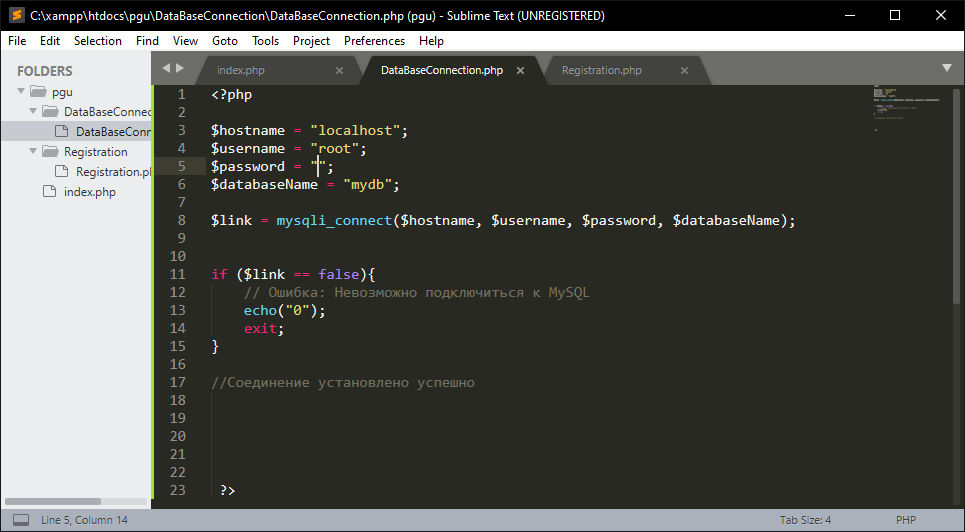


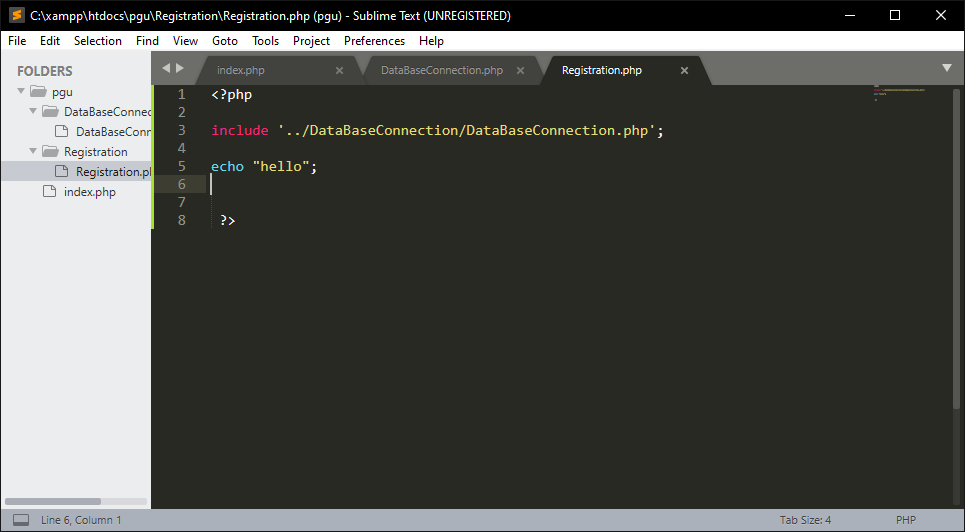
Меню регистрации



Меню авторизации

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



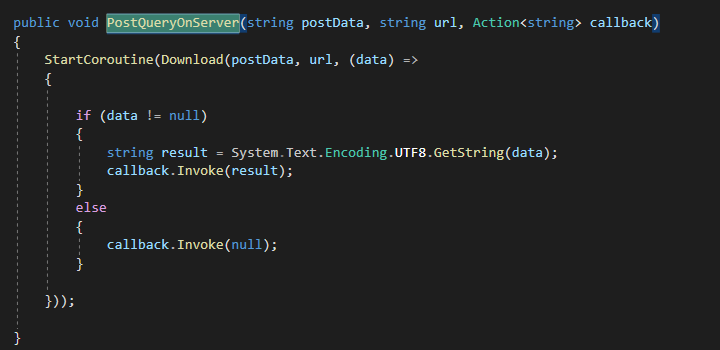


С приложения мы обращаемся по адресу "http://localhost/Registration/Registration.php"   
По этой ссылке выполняется скрипт написанный выше, а он вызывает подскрипт подключения к базе данных, после чего возвращает ответ “hello”

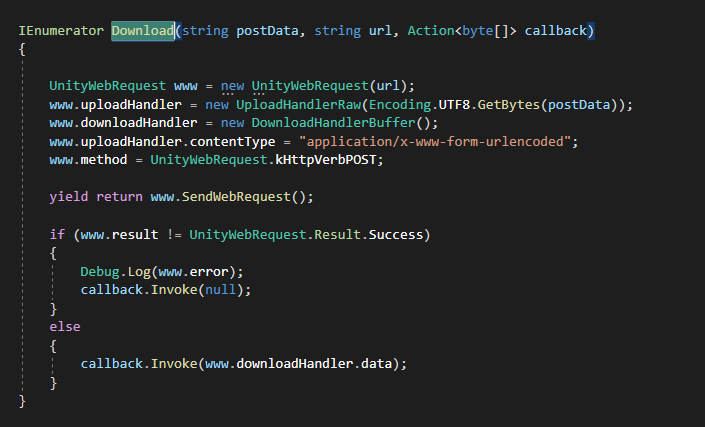
В приложении скрипт работает так, что он запрашивает ответ по ссылке:



Static метод для вызова с верхнего уровня

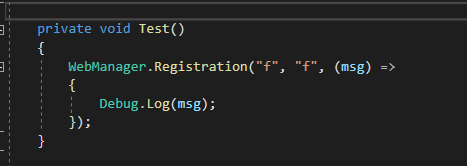


Метод вызова корутины



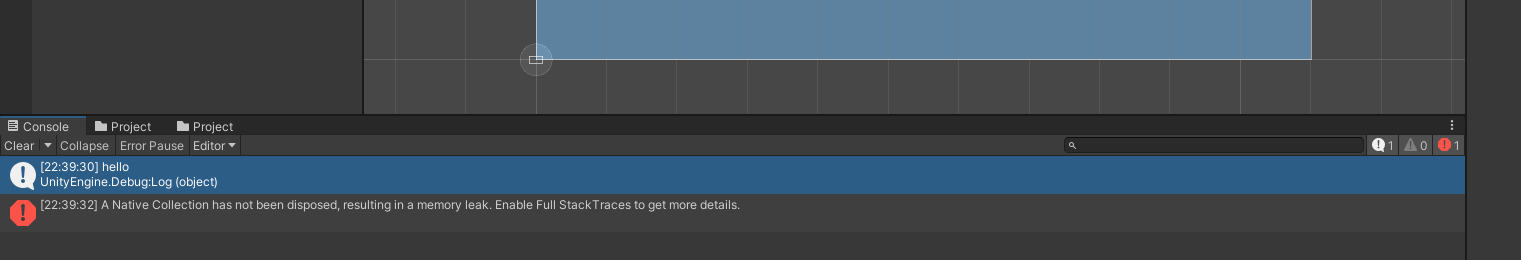
Корутина с запросом на сервер

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.



Тестовый метод

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

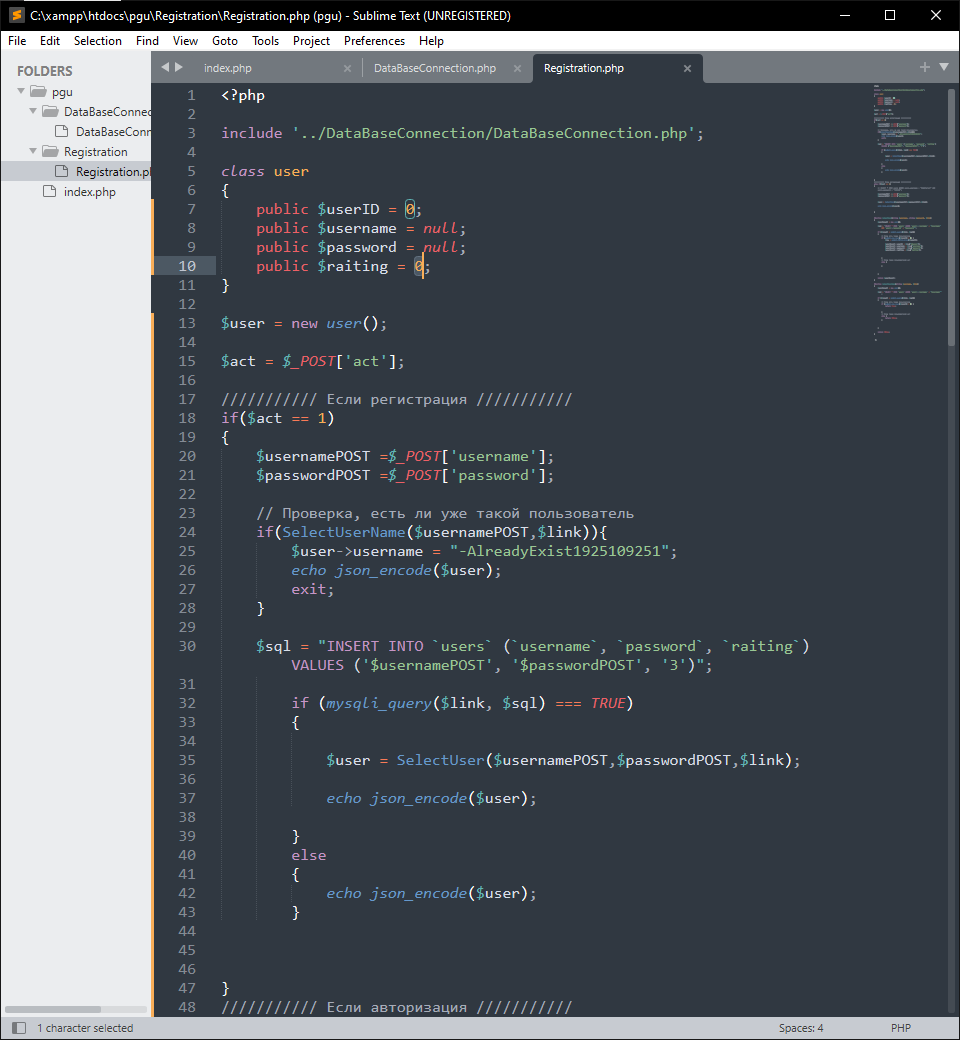


Тестовый ответ от сервера

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:

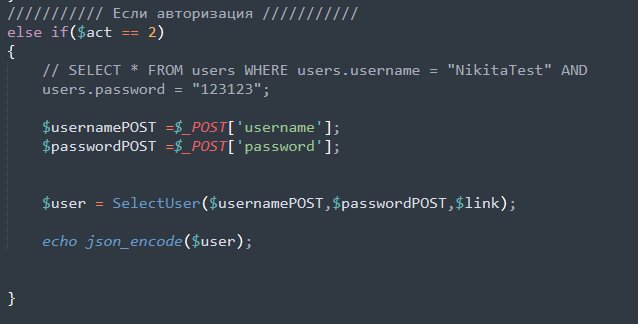


Скрипт регистрации на php

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.

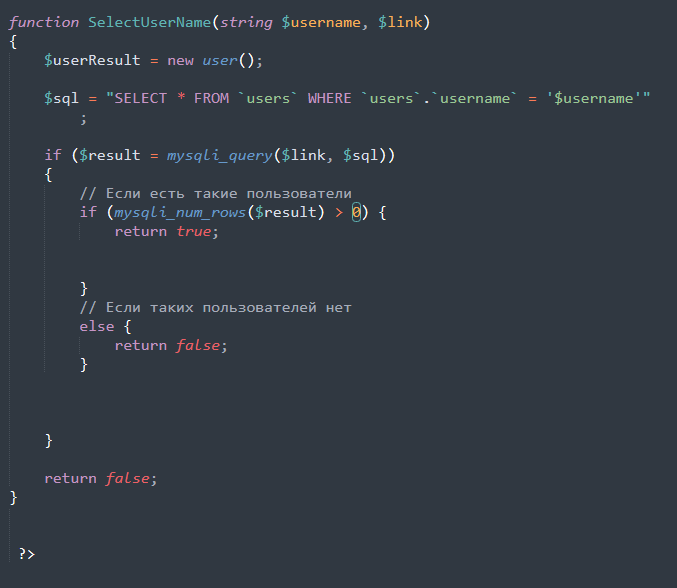
Если act = 2, то это авторизация и нужно просто вернуть все данные о пользователе



Скрипт php при авторизации

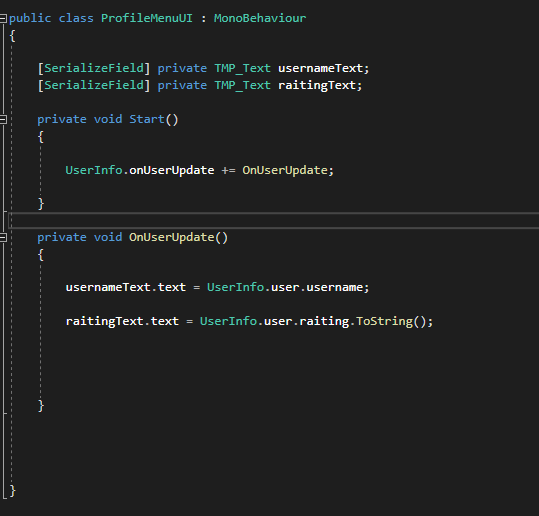
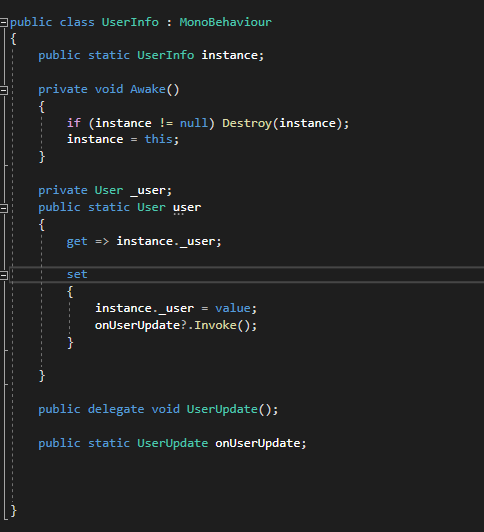


Функция выборки пользователя из базы данных



Функция проверки существования пользователя с таким именем на php

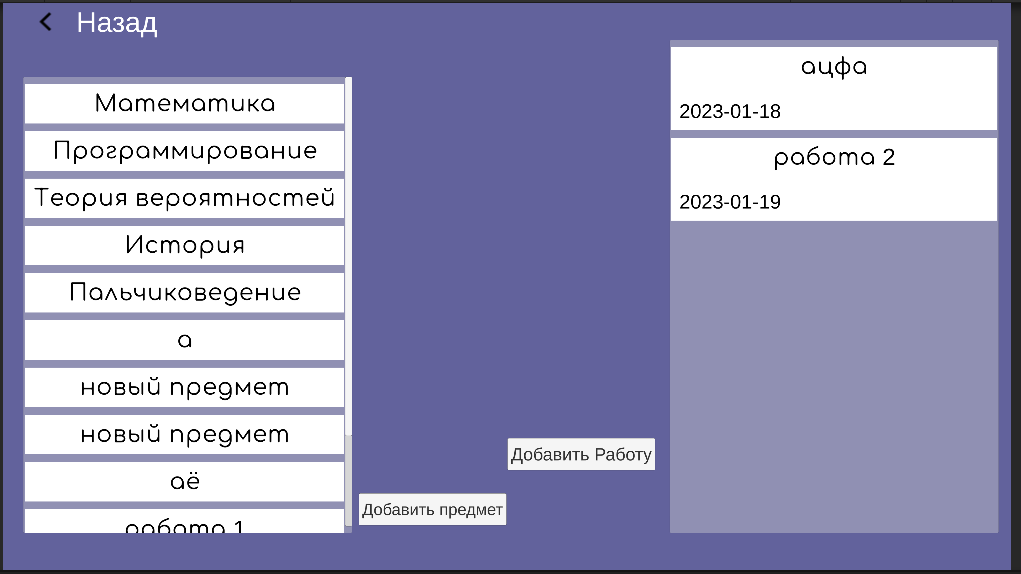
Далее создал окно профиля, через которое можно будет создавать новые лабораторные работы, отвечать на вопросы и искать ответы от других пользователей. В Профиле отображается имя пользователя и его рейтинг – они обновляются с помощью static делегата, который вызывается при присвоении нового пользователя.



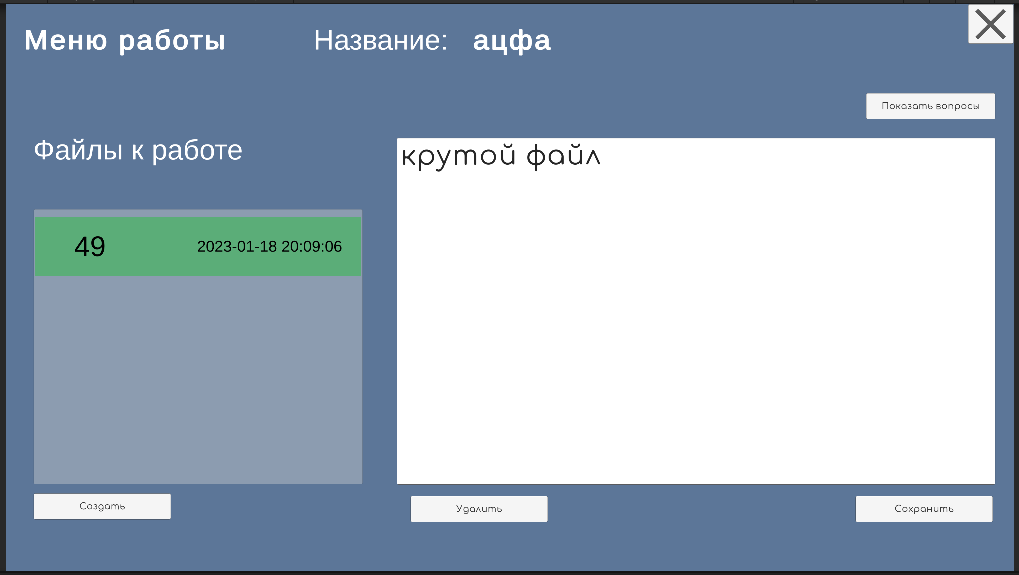
Далее все меню были созданы по оному образу:



Меню профиля



Меню лабораторных работ



Меню работы



Меню вопросов и ответов

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

**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

Получение и создание Предметов:

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

**Получить все предметы**

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

Ответ json:

Массив из

subject:

subject\_id

theme\_name

**Создать предмет**

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

Ответ 1 – если успешно

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

Вернуть все работы, в которых есть файлы с id пользователя

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

Ответ: объект с Items массивом из объектов task

Вернуть все файлы по id работы

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

Создать новый файл к предмету

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

Ответ: объект file

Сохранить файл

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

Ответ:

1 – выполнилось

0 – Не выполнилось

Удалить файл

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

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

Вернуть ответы по вопросу

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

Ответ: объект с Items массивом из объектов answers

Вернуть имя пользователя

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

Создать вопрос

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

Создать ответ

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