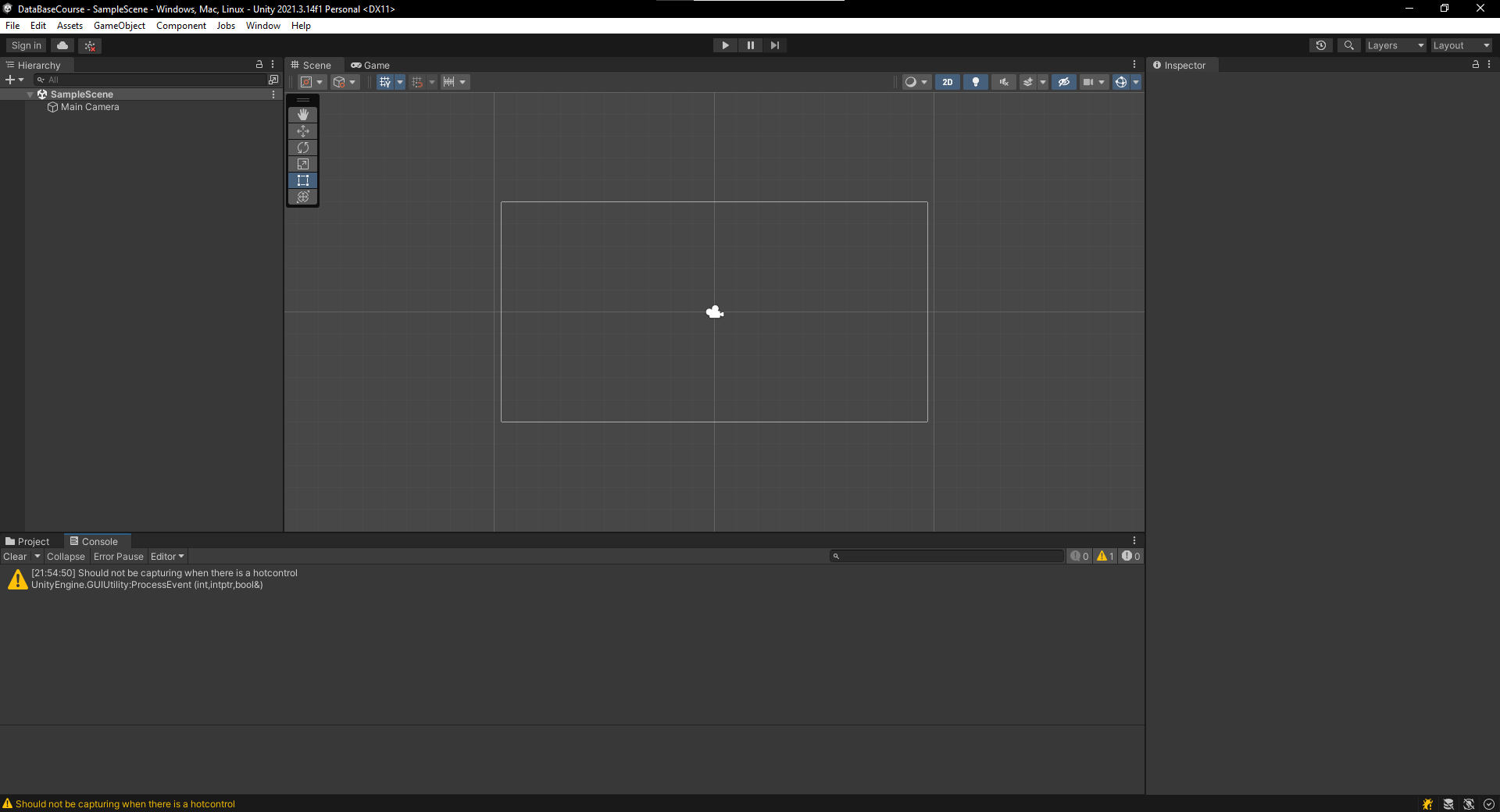
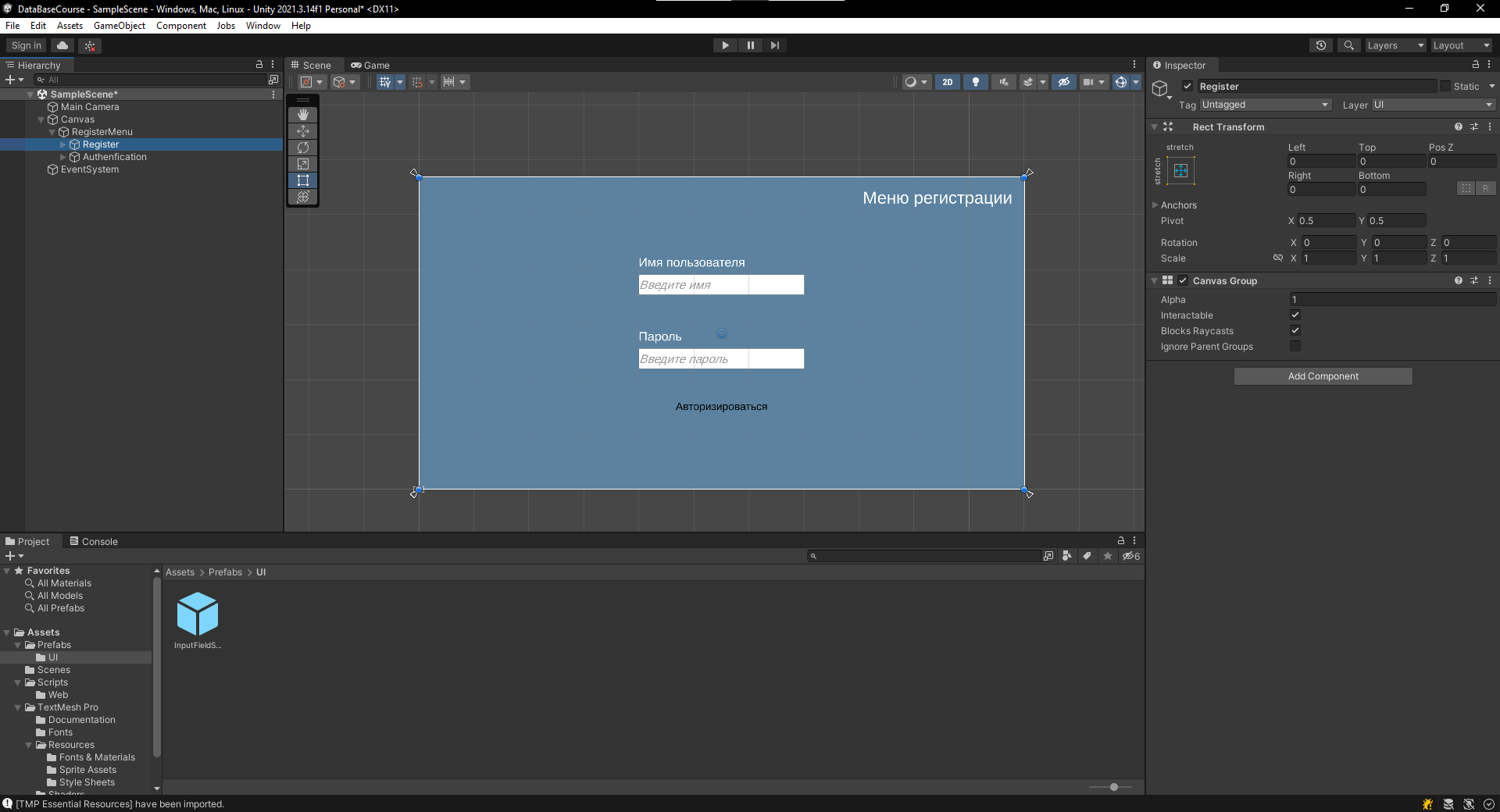
Первым делом нужно спланировать, как будет связываться приложение с базой данных.   
Я решил, что лучше всего связывать приложение будет с помощью POST запросов.

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

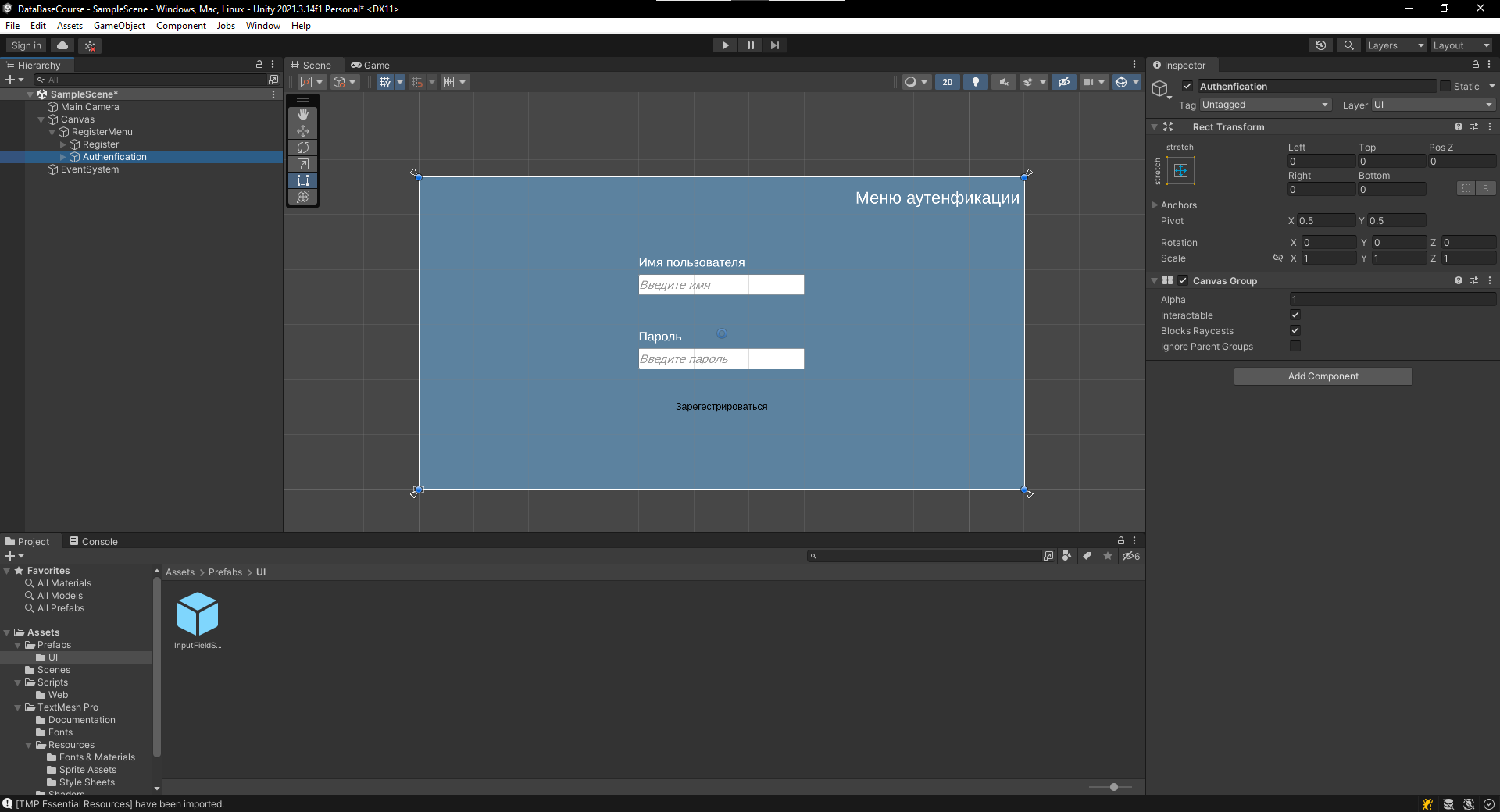
Далее создадим новый проект на Unity и сверстаем UI меню для регистрации или входа пользователя.



Новый проект

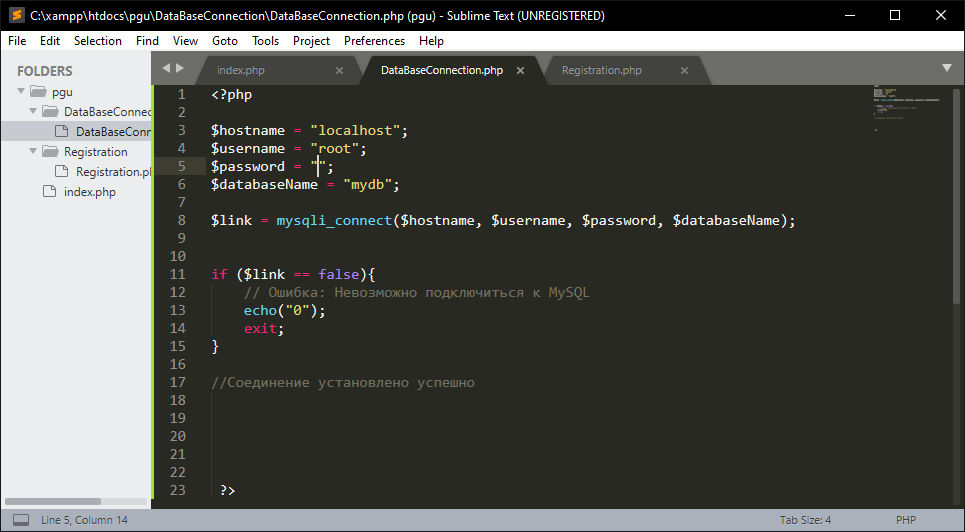


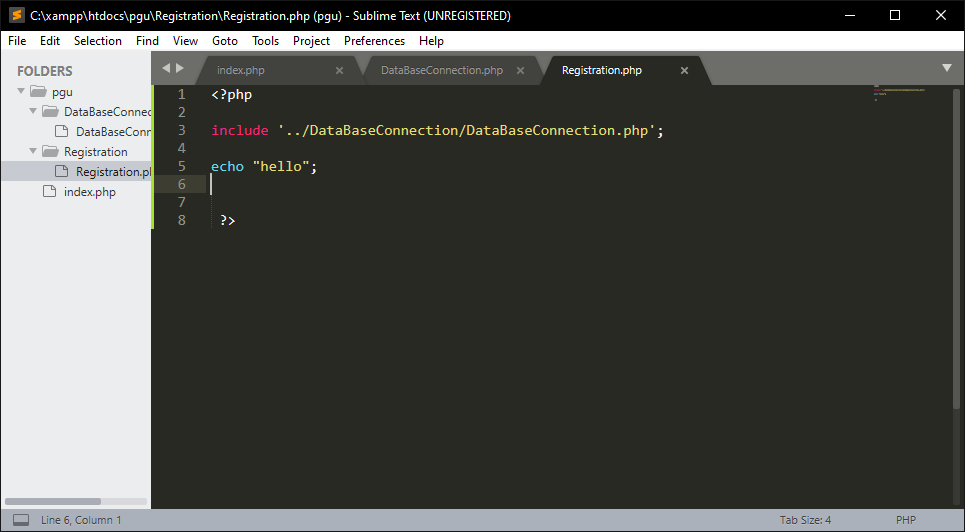
Меню регистрации



Меню авторизации

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



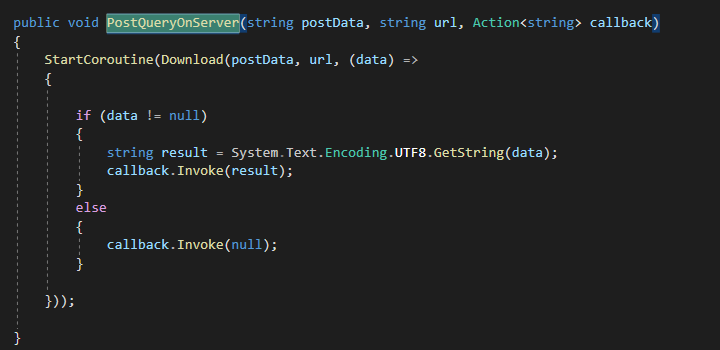


С приложения мы обращаемся по адресу "http://localhost/Registration/Registration.php"   
По этой ссылке выполняется скрипт написанный выше, а он вызывает подскрипт подключения к базе данных, после чего возвращает ответ “hello”

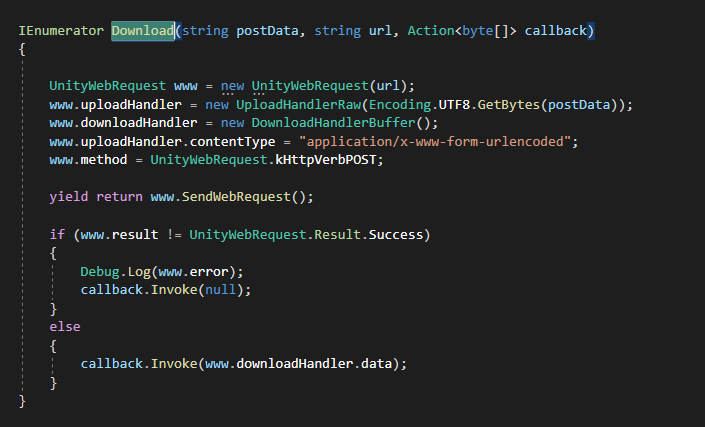
В приложении скрипт работает так, что он запрашивает ответ по ссылке:



Static метод для вызова с верхнего уровня

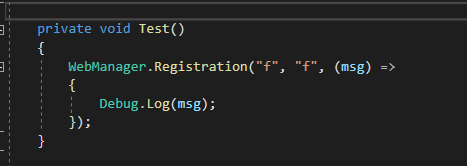


Метод вызова корутины



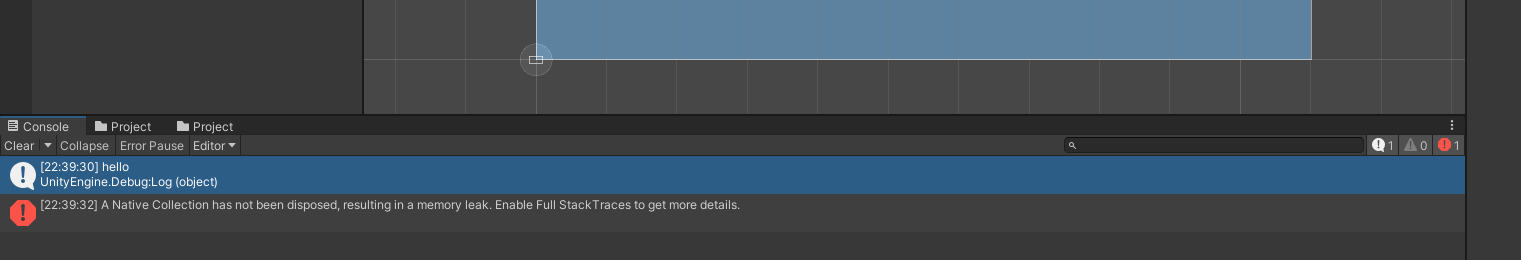
Корутина с запросом на сервер

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:

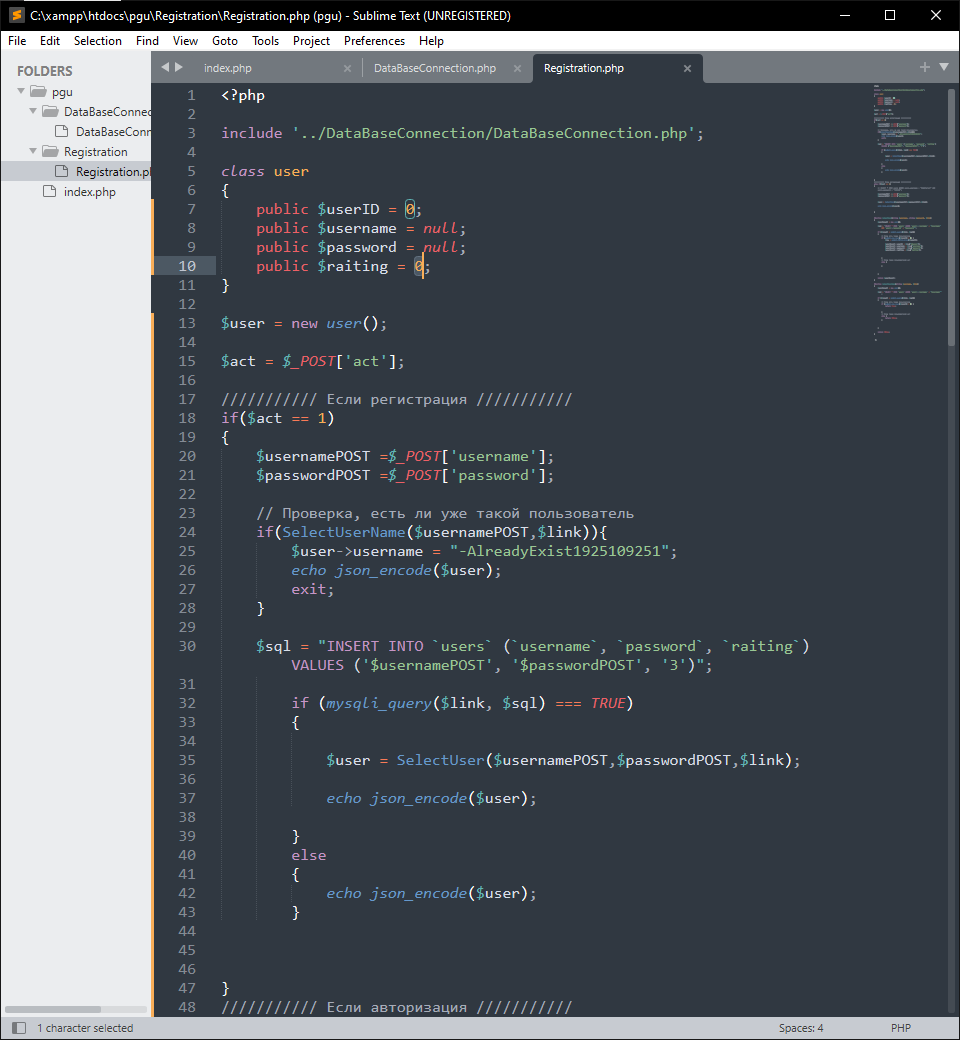


Тестовый ответ от сервера

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:

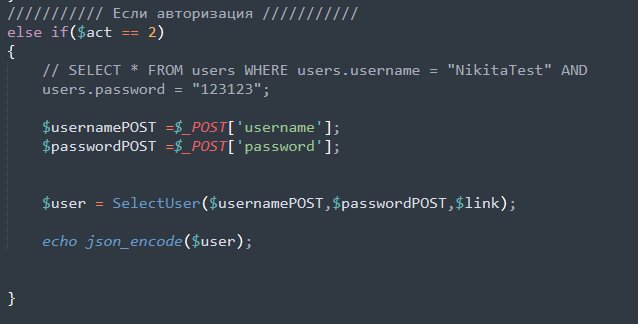


Скрипт регистрации на php

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.

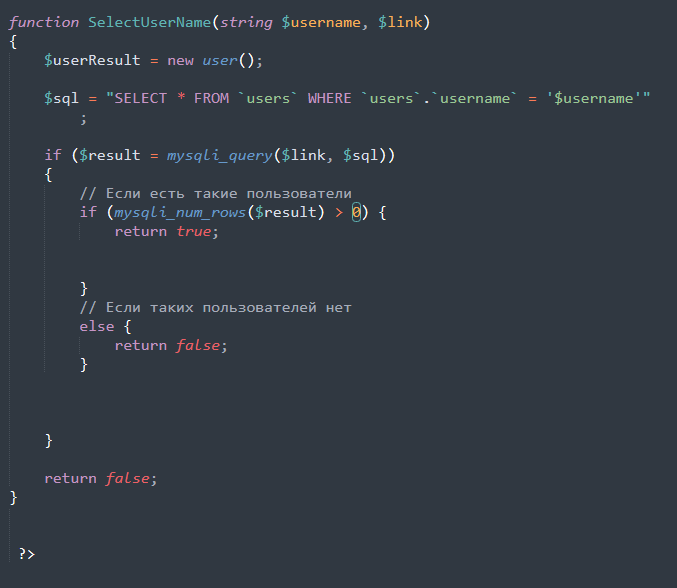
Если act = 2, то это авторизация и нужно просто вернуть все данные о пользователе



Скрипт php при авторизации

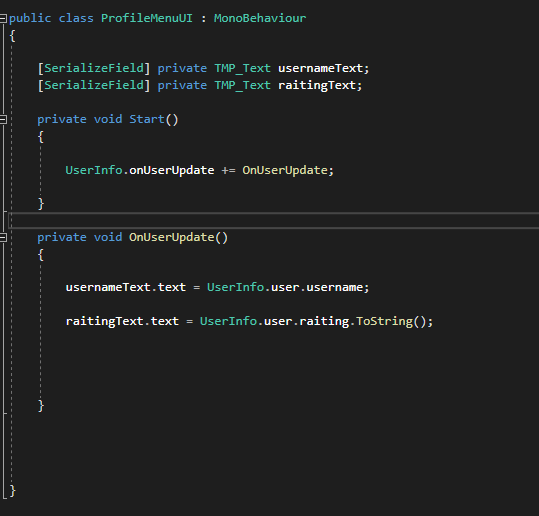
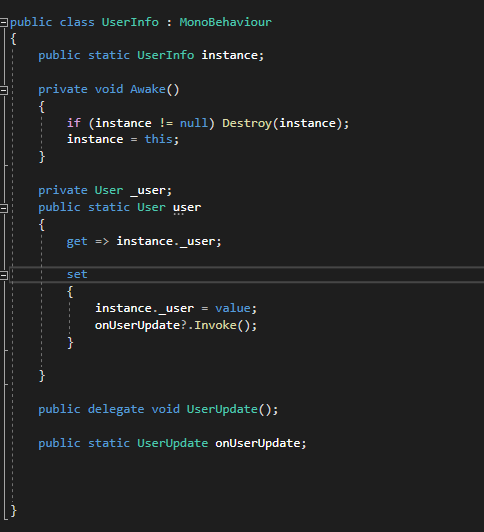


Функция выборки пользователя из базы данных



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.



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 |

Ответ 1 – если успешно

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

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

Ответ: объект с Items массивом из объектов task

{

}