УДК 004

Реализация методов авторизации в приложении на Unity

Фатеенков Данила Витальевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье рассмотрены подходы реализации авторизации в приложении, которое было разработано в программном обеспечении Unity. Приводится описание работы двух основных классов, предоставляющие доступ к веб-ресурсам: классы WWW и UnityWebRequest. Реализация представлена в нескольких видах с применением функций Unity API, а также функционала Google Firebase.
Ключевые слова: Unity, авторизация, Google Firebase, WWW, UnityWebRequest, PHP, C#

Implementing authorization methods in an application on Unity

Fateenkov Danila Vitalievich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the approaches to implementing authorization in an application that has been developed in the Unity software. It describes how the two main classes that provide access to web resources work: the WWW and UnityWebRequest classes. The implementation is presented in several views, using the functions of the Unity API, as well as the functionality of Google Firebase.
Keywords: Unity, authorization, Google Firebase, WWW, UnityWebRequest, PHP, C#

1. Введение

1.1 Актуальность

Многие веб-ориентированные приложения предоставляют пользователям возможность регистрации в системе с доступом к тем или иным функциям, которые могут быть недоступны гостевым пользователям. Соответственно появляется необходимость реализовать возможность авторизации в приложении.

Система пользователей и контроля ролей реализована во многих сферах деятельности человека (в играх тоже, соответственно). Одним из популярных движков для разработки игры под разные платформы является Unity, и он предоставляет необходимый функционал для настройки разработчиком соединения с удалённым сервером. Также в Unity может быть интегрирован

Google Firebase, который имеет широкий инструментарий для реализации регистрации и авторизации пользователей в приложении.

1.2 Обзор исследований

А.А. Петров и Е.А. Федотов описали средства для реализации сетевых игр, а также основные архитектуры сетевых приложений при разработке игр на Unity [1]. В.И. Васильев описал процесс подключения к сервисам Google Play и настройки приложения, разработанного на Unity [2]. А.А. Петров и Е.А. Федотов описали процесс разработки сетевых игр на Unity [3]. О.Е. Первун и Э.Р. Халилов представили процесс разработки серверной части для Android-приложения, разработанного на Unity [4].

1.3 Цель исследования

Цель – реализовать алгоритмы авторизации на удалённом сервере и рассмотреть способы авторизации с применением Google Firebase.

2. Материалы и методы

Для достижения поставленной задачи используется язык программирования C# и программное обеспечение для разработки видеоигр и приложений Unity, а также язык программирования PHP и облачная платформа Google Firebase.

3. Результаты и обсуждения

Unity API поддерживает 2 способа установления соединения с удалённым сервером для получения и передачи определённой информации: WWW и UnityWebRequest. Первый способ считается устаревшим, но в настоящее время также работает, и разработчики не редко используют его. WWW ограничен в функционале и позволяет только получать и отправлять данные на сервер, то есть выполняются только 2 HTTP-запроса: GET и POST.

WWW реализуется через создание экземпляра класса WWWForm, в которую помещаются необходимые для работы с сервером данные. После создания формы класса WWWForm, определяются необходимые для отправки данные: это могут быть числовые, строковые, логические и другого типа данные. Информация помещается в ранее созданную форму с помощью метода “AddField”, который принимает на вход 2 значения: ключ, на который будет ссылаться PHP-код и значение, соответствующее этому ключу:

```
WWWForm form = new WWWForm();
form.AddField("id",1);
form.AddField("name",Player1);
form.AddField("Score",436);
```

После заполнения формы необходимо создать экземпляр класса WWW, который на вход будет принимать 2 параметра: ссылка на удалённый сервер и форма, которую необходимо передать:

```
     WWW www = new
WWW("http://gameglobalserver.com/manager/writedata.php", form);
```

Для получения данных с сервера форма в запросе не указывается. Но указывать форму нет необходимости только в случае, если разработчик заранее знает, что получит конкретную информацию по запросу (то есть сервер не может вернуть другой ответ при успешном выполнении запроса). Также стоит учитывать, что такой запрос в качестве ответа может вернуть код страницы, что может не подойти для дальнейшей работы. Один из вариантов выхода из такой ситуации это добавление формы с одним полем, который будет ссылаться на определённую функцию в странице (а функция будет возвращать необходимую информацию).

Использовать WWW метод не рекомендуется, так как имеет ограниченный функционал и часто бывает неудобен в использовании. Заменой WWW служит UnityWebRequest.

Данный класс имеет расширенный функционал: больше двух видов запросов к серверу (GET, POST, PUT, DELETE), введены методы для отслеживания ошибок (есть 2 метода: isNetworkError – определение проблем с соединением, и isHttpError – определение проблем со стороны сервера или некорректности отправленного запроса). Для получения данных с сервера используется метод “SendWebRequest” и функция получения данных выглядит следующим образом:

```
IEnumerator GetUserData() {
     UnityWebRequest www = UnityWebRequest.Get(userDataURL);
     yield return www.SendWebRequest();
     if (www.isHttpError || www.isNetworkError) {
          Debug.Log("Something went wrong!");
          yield break;
     }

     Debug.Log(www.downloadHandler.text);
}
```

Данный скрипт отправляет запрос на удалённый сервер, где метод “Get” получает на вход в качестве параметра ссылку на удалённый сервер.

Функция для авторизации может выглядеть следующим образом:

```
private IEnumerator SendUserData()
{
          WWWForm form = new WWWForm();
          form.AddField("Login", userLogin);
          form.AddField("Password", userPassword);
          UnityWebRequest www =
     UnityWebRequest.Post(postDataURL, form);
          yield return www.SendWebRequest();
          if (www.isHttpError || www.isNetworkError) {
               Debug.Log("Something went wrong!");
```

```
                yield break;
            }
        Debug.Log(www.downloadHandler.text);
    }
```

Но данная функция только отправляет данные. На сервере необходимо проверить наличие пользователя в базе данных. Следующий код на PHP делает запрос к БД и ищет пользователя с таким же именем:

```
    $name = $_POST["Login"];
    $pass = $_POST["Password"];

    $mysqli = new mysqli("*адрес сервера*", "*логин администратора БД*", "*пароль доступа к БД*", "*название БД*");

    $result = $mysqli->query('SELECT * FROM `users` WHERE name = "' . $name . '"');
```

Данный запрос выполняет поиск записи в БД, в которой поле name соответствует отправленному пользователем запросу. Но этого недостаточно для авторизации: необходимо проверить, совпадают ли пароли:

```
    $user = mysqli_fetch_array($result, MYSQLI_NUM);
    if (count($user) == 0) {
        echo "-1";
    }
    else {
        if ($pass != $user['pass']) {
            echo "-2";
        }
        else {
            echo "0";
        }
    }
```

Данный код сначала переводит результат запроса в массив (потому что изначально он представлен в виде mysqli_result), а затем проверяет наличие пользователя, определяя длину получившегося массива. Если длина массива равна нулю, то сервер возвращает значение “-1”, что в приложении можно интерпретировать как “Такого пользователя не существует”. Если пользователь есть, но при этом пароли не совпадают, то сервер возвращает “-2”, что может означать “Неправильные данные для входа в аккаунт”. Если пользователь существует и пароль правильный, то вернётся значение 0, которое может служить триггером для перехода на другую сцену или для открытия UI-интерфейса пользователя в приложении.

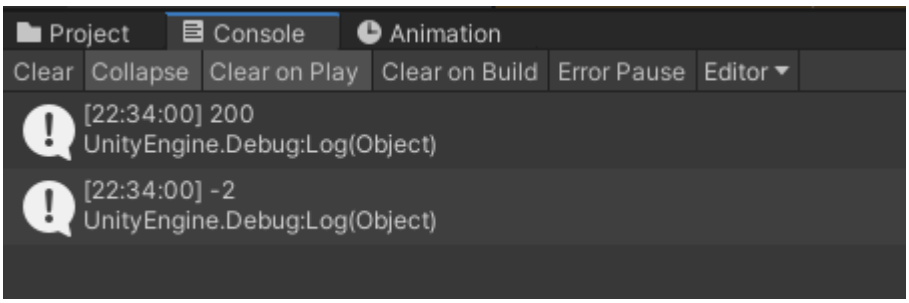

Рисунок 1. Ответ со стороны сервера при отправке данных о пользователе с неверным паролем

Это базовая реализация авторизации и имеет проблемы в безопасности. Умеющий работать с файлами Unity приложения разработчик может быстро определить, где находится файл авторизации и если файл не защищён, то может произойти утечка данных или взлом сервера.

Для Android-приложений существует более надёжный вариант хранения пользовательских данных – Google Firebase. Это платформа, которая предоставляет услуги в сфере облачных технологий. В число этих услуг входит:

1. Базы данных;
2. Модели машинного обучения;
3. Аутентификация;
4. Хранилища данных;
5. Средства для аналитики.

Это не весь пакет услуг Firebase, платформа включает в себя большое количество возможностей для разработчика. Главное условие для приложения, которое использует Firebase – наличие на устройстве сервисов Google Play.

Аутентификация является одной из важнейших функций платформы и поддерживает большое количество способов входа в аккаунт в приложении: почта, номер телефона, Google аккаунт, социальные сети, Play Games аккаунт.

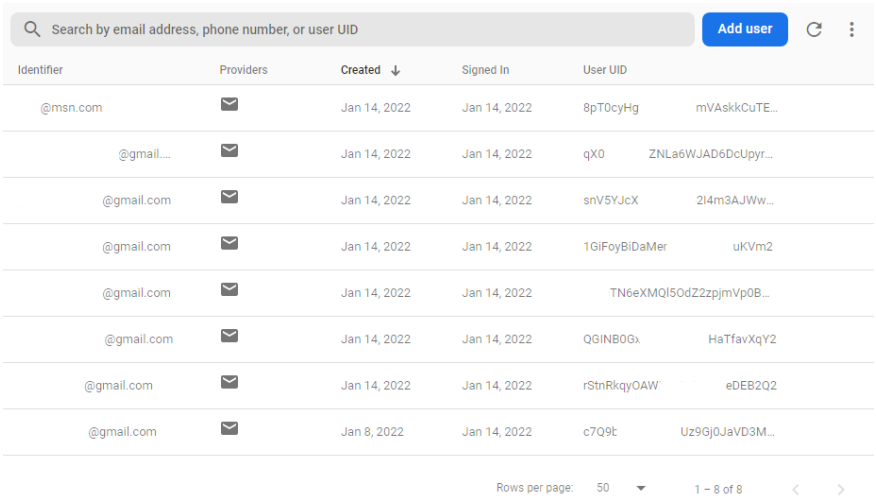

Рисунок 2. Окно пользователей в аутентификации Google Firebase

В первую очередь проверяются зависимости в приложении и наличие сервисов Google Play (API платформы Firebase предоставляет такой функционал).

Перед началом процесса аутентификации необходимо инициализировать экземпляр (DefaultInstance):

```
Firebase.Auth.FirebaseAuth auth =
Firebase.Auth.FirebaseAuth.DefaultInstance;
```

Переменная “auth” типа данных “Firebase.Auth.FirebaseAuth” предназначена для установки соединения с платформой Firebase. Также часто необходимо определить пользователя в приложении:

```
Firebase.Auth.FirebaseUser user;
```

Для авторизации с использованием почты используется метод “SignInWithEmalAndPasswordAsync”. Данный метод принимает на вход 2 параметра: адрес почты и пароль от аккаунта. При успешном входе, создаётся новый пользователь, которому присваивается результат выполнения метода:

```
auth.SignInWithEmailAndPasswordAsync(email,
pass).ContinueWith(q => {
        if (q.IsFaulted) {
            Debug.Log(“Something went wrong! ” + q.Exception);
        }
        else {
            Firebase.Auth.FirebaseUser user = q.Result;
            Debug.Log(“Signed in!”);
        }
});
```

Для регистрации нового пользователя используется метод “CreateUserWithEmailAndPasswordAsync”, который имеет схожие свойства и реализацию с методом “SignInWithEmalAndPasswordAsync”. При попытке войти в несуществующий аккаунт, сервер вернёт сообщение об ошибке, оповещая о том, что пользователя н существует, либо он был удалён (см. рис. 3).

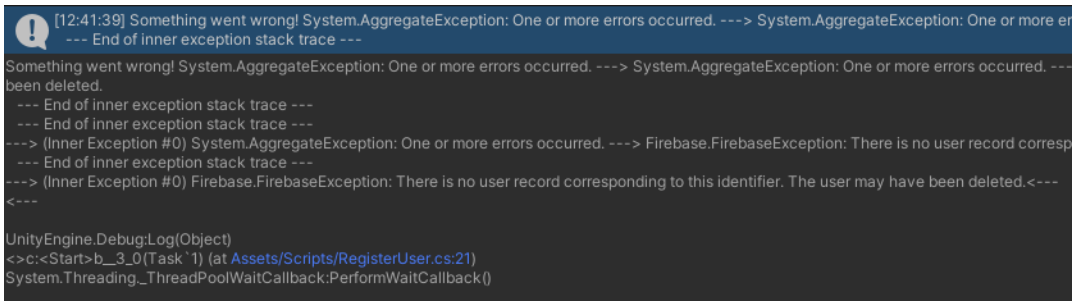

Рисунок 3. Результат попытки войти в несуществующий аккаунт

Только что созданный аккаунт будет отображён в таблице пользователей, но у него будет пустое поле “Signed in” до того момента, пока пользователь не войдёт в аккаунт (см. рис. 4).

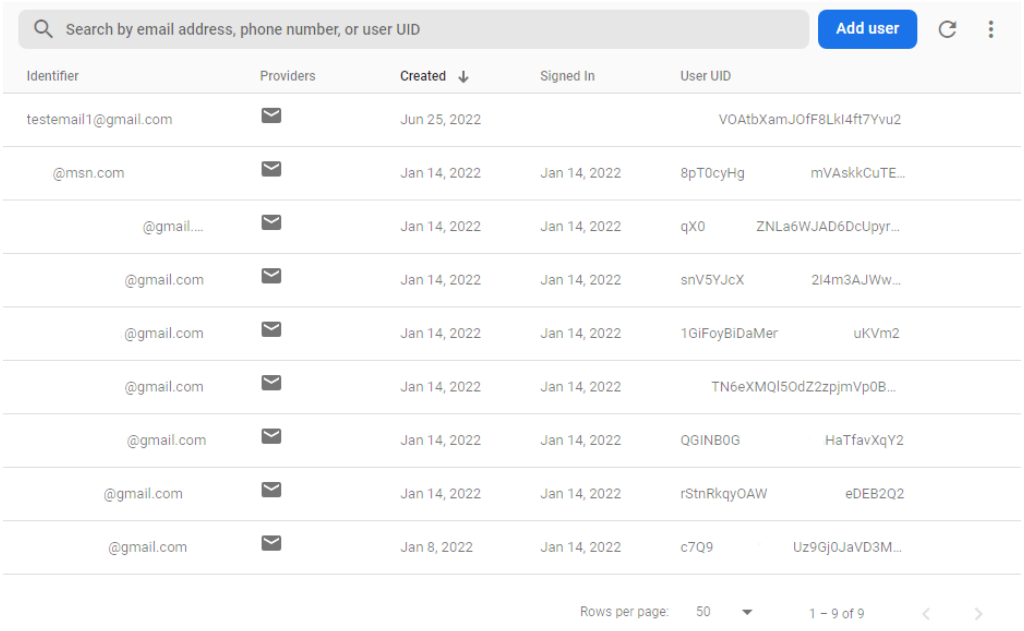

Рисунок 4. Таблица пользователей после создания нового в приложении

Также API платформы Firebase позволяет получать данные о пользователях с сервера. Для этого существуют следующие методы:

1. DisplayName – имя пользователя (может отсутствовать);
2. Email – электронная почта пользователя;
3. PhotoUrl – ссылка на изображения профиля пользователя;
4. UserId – идентификатор пользователя в Auth системе;
5. ProviderId – идентификатор провайдера (того, через кого происходит подключение к Auth);
6. ProviderData – информация о провайдере.

Также есть и другие методы для получения информации о пользователе.

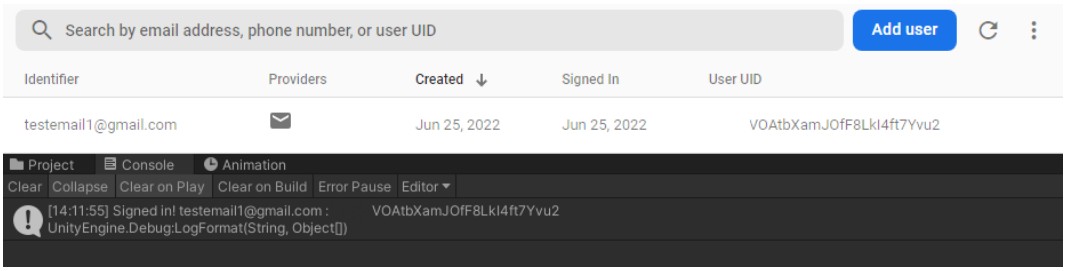

Рисунок 5. Результат успешной авторизации в Firebase Auth

Таким образом, в статье были рассмотрены подходы к реализации авторизации в приложении, которое разрабатывается с применением программного обеспечения Unity. Были рассмотрены основные методы и функции классов WWW и UnityWebRequest, которые относятся к библиотеке

Networking. Также были рассмотрены основные методы реализации регистрации и авторизации пользователя с использованием технологий Google Firebase.

Библиографический список

1. Петров А.А., Федотов Е.А. Разработка простых сетевых игр с использованием игрового движка Unity // XII международный молодёжный форум "Образование. Наука. Производство". Белгород: Белгородский государственный технологический университет им. В.Г. Шухова, 2020. С. 1961-1968.
2. Васильев В.И. Настройка Google Play Services для проекта Unity // Интеграция наук. 2019. № 2 (25). С. 79-84.
3. Петров А.А., Федотов Е.А. Разработка простых сетевых игр с использованием игрового движка Unity // XII международный молодёжный форум "Образование. Наука. Производство". Белгород: Белгородский государственный технологический университет им. В.Г. Шухова, 2020. С. 1961-1968.
4. Первун О.Е., Халилов Э.Р. Разработка серверной части игрового приложения для платформы Android с использованием Unity // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. Крым: Крымский инженерно-педагогический университет, 2018. С. 26-33.
5. Unity - Manual: Unity User Manual 2020.3 (LTS) URL: https://docs.unity3d.com/Manual/index.html.
6. Firebase URL: https://firebase.google.com.