



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ

Робототехника и комплексная автоматизация (РК)

КАФЕДРА

Системы автоматизированного проектирования (РК6)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

*«Интеграция сетевых компонентов в шаблон
многопользовательской игры на Unreal Engine 4»*

Студент РК6-73Б

(Подпись, дата)

Боженко Д.В.

И.О. Фамилия

Руководитель курсового проекта

(Подпись, дата)

Витюков Ф.А.

И.О. Фамилия

2022 г.

**Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ
Заведующий кафедрой РК6
А.П. Карпенко

« ____ » _____ 20 ____ г.

**ЗАДАНИЕ
на выполнение курсового проекта**

по дисциплине _____ Модели и методы анализа проектных решений

Студент группы _____ РК6-73Б

Боженко Дмитрий Владимирович
(Фамилия, имя, отчество)

Тема курсового проекта Интеграция сетевых компонентов в шаблон многопользовательской игры на Unreal Engine 4

Направленность КП (учебный, исследовательский, практический, производственный, др.)
учебный

Источник тематики (кафедра, предприятие, НИР) _____ кафедра

График выполнения проекта: 25% к 5 нед., 50% к 11 нед., 75% к 14 нед., 100% к 16 нед.

Задание. Провести анализ существующих подсистем, предоставляющих ряд кроссплатформенных функций современных многопользовательских игр. Выбрать существующую подсистему, и изучить принципы работы с авторизацией в рамках данной подсистемы в движке Unreal Engine 4. Интегрировать авторизацию пользователя в игровую сессию в существующий шаблон многопользовательской игры на уровне приложения. Изучить дополнительные возможности подсистемы для дальнейшего улучшения функционала проекта.

Оформление курсового проекта:

Расчетно-пояснительная записка на 30 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

3 графических листа _____

Дата выдачи задания «10» сентября 2022 г.

Руководитель курсовой работы

(Подпись, дата)

Витюков Ф.А.
И.О. Фамилия

Студент

(Подпись, дата)

Боженко Д.В.
И.О. Фамилия

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

АННОТАЦИЯ

В данной работе рассмотрены основные виды Online Subsystem (OSS) в движке Unreal Engine. Описаны основные доступные виды интерфейсов, предоставляемые Epic Online Subsystem. Описаны все доступные виды авторизации в Epic Online Subsystem. Реализованы два вида авторизации и виджеты для навигации пользователя среди типов авторизации и введении данных от учетной записи пользователя Epic Games.

В расчетно-пояснительной записке 30 листов, 13 рисунков, 3 графических листа, 6 листингов.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	5
ВВЕДЕНИЕ	7
1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ	8
1.1. Интерфейсы EOS, интегрируемые в проект	8
1.1.1 Auth Interface	9
1.1.2. Leaderboards Interface	12
1.1.3. Lobbies Interface	14
2. ПРАКТИЧЕСКАЯ ЧАСТЬ	16
2.1. Создание клиентов и политик для авторизации пользователя	16
2.2. Авторизация пользователя с помощью данных аккаунта Epic Games	19
2.3. Авторизация пользователя с помощью лаунчера Epic Games	27
ЗАКЛЮЧЕНИЕ	28
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	29

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ЯП — язык программирования.

Многопользовательская игра — режим компьютерной игры, в котором играет более одного пользователя по сети Интернет.

Движок — это программный фреймворк, предназначенный в первую очередь для разработки видеоигр и обычно включающий соответствующие библиотеки и программы поддержки.

UE 4 — движок Unreal Engine 4.

Клиент — машина, которая получает информацию об игровом мире через сервер и на которой происходит отрисовка игрового процесса.

Сервер — мощная вычислительная машина, через которую происходит обмен информацией об игровом мире без отрисовки графики и воспроизведения звуков.

Удаленный игрок — игрок, который находится на другой машине в пределах одной игровой сети.

Локальный игрок — игрок, который находится на локальной машине.

LAN — локальная вычислительная сеть, где все участники находятся, как правило, в пределах одной ограниченной территории.

AActor — один из основных классов в UE 4, являющийся базовым для всех остальных классов, представленных в игровом мире.

OnlineSubsystem (далее OSS) — кроссплатформенная система, позволяющая использовать современные возможности многопользовательских игр.

Epic Online Services (далее EOS) — OSS, предоставляемая компанией Epic Games.

Epic Account Services (далее EAS) — плагин, являющийся частью EOS, предоставляющий доступ к интерфейсу авторизации и многим другим интерфейсам.

Epic Game Services — плагин, являющийся частью EOS, предоставляющий доступ к интерфейсам, связанным с игровым процессом.

DevPortal — Интернет ресурс Epic Games, предназначенный для создания и редактирования настроек приложения, использующего EOS.

УЗ — Учетная запись.

ВВЕДЕНИЕ

Рынок видеоигр стремительно развивается с каждым годом. На сегодняшний день рынок игр во всем мире является одним из самых больших сегментом мирового рынка цифрового контента, ежегодно генерируя многомиллиардные доходы и привлекая огромную аудиторию. Наибольшая доля в структуре российского рынка приходится на сегмент онлайн-игр. По данным *Mail.ru Group*, в 2019 году его объем увеличился на 9% и составил 56,7 млрд рублей (около \$1 млрд).

Среди всех жанров игр на данный момент самыми популярными являются *ММО* (массовые многопользовательские онлайн игры), которые использует *Real-Time Multiplayer*. *Real-Time Multiplayer* — это тот режим игры, в котором каждый пользователь получает и отправляет данные об игровом мире с выделенного игрового сервера несколько десятков раз за отведенный промежуток времени. Данное понятие называется тикрейт, т. е. какое количество запросов сервер может обрабатывать за установленные промежуток времени.

Целью данной практической работой является изучение одной из доступных подсистем для движка Unreal Engine 4, а именно предоставляемых ей программных интерфейсов, таких как авторизация, античит-система, система достижений и т. п., которые широко применяются в современных многопользовательских играх. На основе полученных знаний реализовать интеграцию авторизации пользователя в существующий шаблон многопользовательской игры с помощью программного интерфейса подсистемы на уровне приложения. Также необходимо проанализировать другие доступные программные интерфейсы выбранной подсистемы, для дальнейшего улучшения функционала существующего шаблона многопользовательской игры.

Данная цель является актуальной, так как изучение концепции создания многопользовательских игр является необходимым условием освоения рынка видеоигр, которые в свою очередь стремительно развиваются и набирают большую популярность в сфере информационных технологий.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Современные многопользовательские игры включают в себя множество возможностей, которые они могут предоставить пользователям. Самыми распространенными из них являются авторизация, система достижений, матчмейкинг (система подбора игроков), создание лобби, таблицы лидеров, система голосового чата и античит-система.

В UE 4 существует несколько OSS, которые предоставляют доступ к возможностям современных многопользовательских игр, а именно Online Subsystem EOS, Online Subsystem Steam, Online Subsystem Oculus, Online Subsystem Google Play, а также Online Subsystem Null. Каждая из перечисленных подсистем предоставляет возможность использовать возможности современных многопользовательских игр, добавляя собственные интеграции.

Online Subsystem EOS является хорошим выбором, так как предоставляет широкие возможности выбора интерфейсов для реализации возможностей многопользовательских игр, которые могут расширить функционал любого шаблона многопользовательской игры. Также EOS имеет подробную документацию, которую необходимо использовать при интеграции предоставленных интерфейсов в проект.

1.1. Интерфейсы EOS, интегрируемые в проект

EOS подразделяется на два вида сервисов: EAS и EOS. Оба сервиса предоставляют большое количество интерфейсов, которые добавляют в игру возможности современных многопользовательских игр. Оба плагина могут использоваться как одновременно, так каждый по отдельности в независимости друг от друга.

EAS в основном предоставляет интерфейсы для авторизации пользователей с помощью УЗ Epic Games и управления списком друзей. Epic

Games Services предоставляют интерфейсы, которые связаны с управлением многопользовательского игрового процесса пользователей.

Для того, чтобы в приложении можно было использовать данные сервисы, необходимо провести предварительную настройку приложения на DevPortal Epic Games, где нужно получить необходимые данные для инициализации приложения и разрешить использование двух вышеперечисленных сервисов (рисунок 1.1).

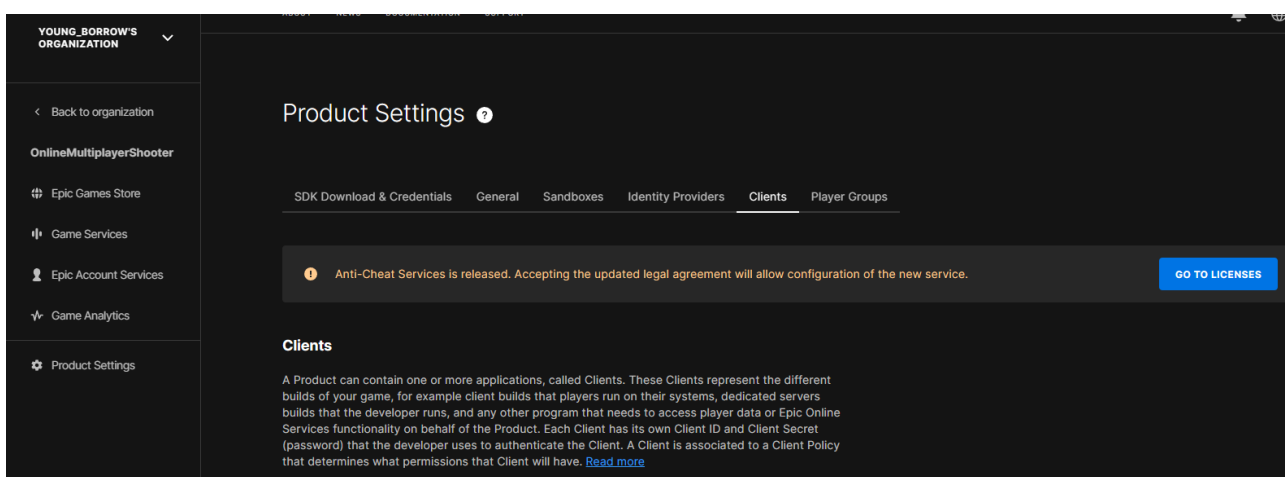


Рисунок 1.1 — Панель управления продуктом на DevPortal

1.1.1 Auth Interface

Auth Interface — программный интерфейс, предоставляемый EAS. Данный программный интерфейс авторизации позволяет игрокам (пользователям) входить в свою УЗ Epic Games прямо из игры, чтобы они могли получить доступ к функциям, предоставляемым EAS, таким как взаимодействие пользователя со списком друзей, просматривание активности других пользователей и даже совершение покупок в Epic Games Store.

В рамках данного проекта использование программного интерфейса авторизации прежде всего необходимо для обеспечения минимального доступа к возможностям многопользовательской игры, а именно, создание игровой сессии, присоединение к уже существующей игровой сессии и доступ к игровому оверлею Epic Games.

Каждый пользователь, успешно авторизованный в систему Epic Games через свою УЗ получает доступ к игровому оверлею, где возможно управлять списком друзей, просматривать полученные достижения из системы игровых достижений (рисунок 1.2).

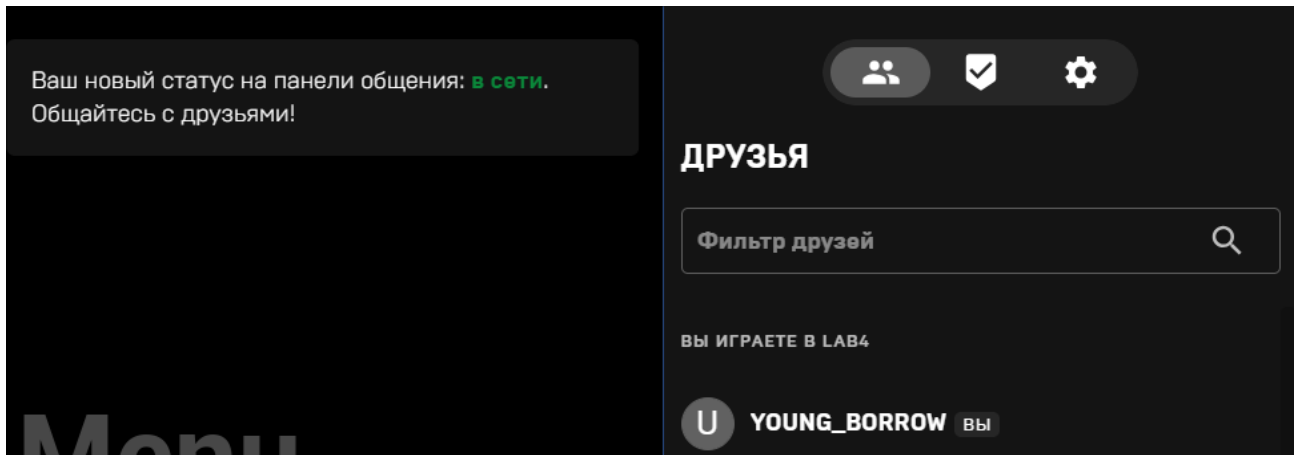


Рисунок 1.2 — Игровой оверлей Epic Games

Auth Interface предлагает пять основных способов авторизации пользователя:

1. Данные от УЗ — данный тип авторизации предполагает, что у пользователя есть возможность ввести данные от своей УЗ Epic Games, а именно почта, к которой привязан аккаунт и пароль.
2. Лаунчер Epic Games — данный тип авторизации эффективен, когда уже готовая версия игры будет запущена через лаунчер Epic Games. Таким образом приложение получает токен авторизации от лаунчера и данный токен используется для входа пользователя в систему.
3. Инструменты разработчика — данный способ авторизации предполагает запуск сервера авторизации на локальной машине, который предоставляет токен. Такой способ подходит во время этапа разработки и не может быть использован пользователем на уровне приложения.

4. Браузер — данный тип авторизации не требует данных пользователя от его УЗ Epic Games. Для авторизации пользователя лишь необходимы данные о настройке приложения, которые были получены в DevPortal. Авторизация пользователя происходит в веб-браузере.
5. Внешняя авторизация — данный способ предполагает авторизацию пользователя через OpenID сервисы, к которым привязана УЧ Epic Games. При таком типе необходим токен авторизации, который был получен от внешнего приложения. В качестве подобного OpenID провайдера может выступать игровая площадка Steam или Discord.

Ниже на рисунке 1.3 представлена схема, по которой EOS работает с токеном авторизации.

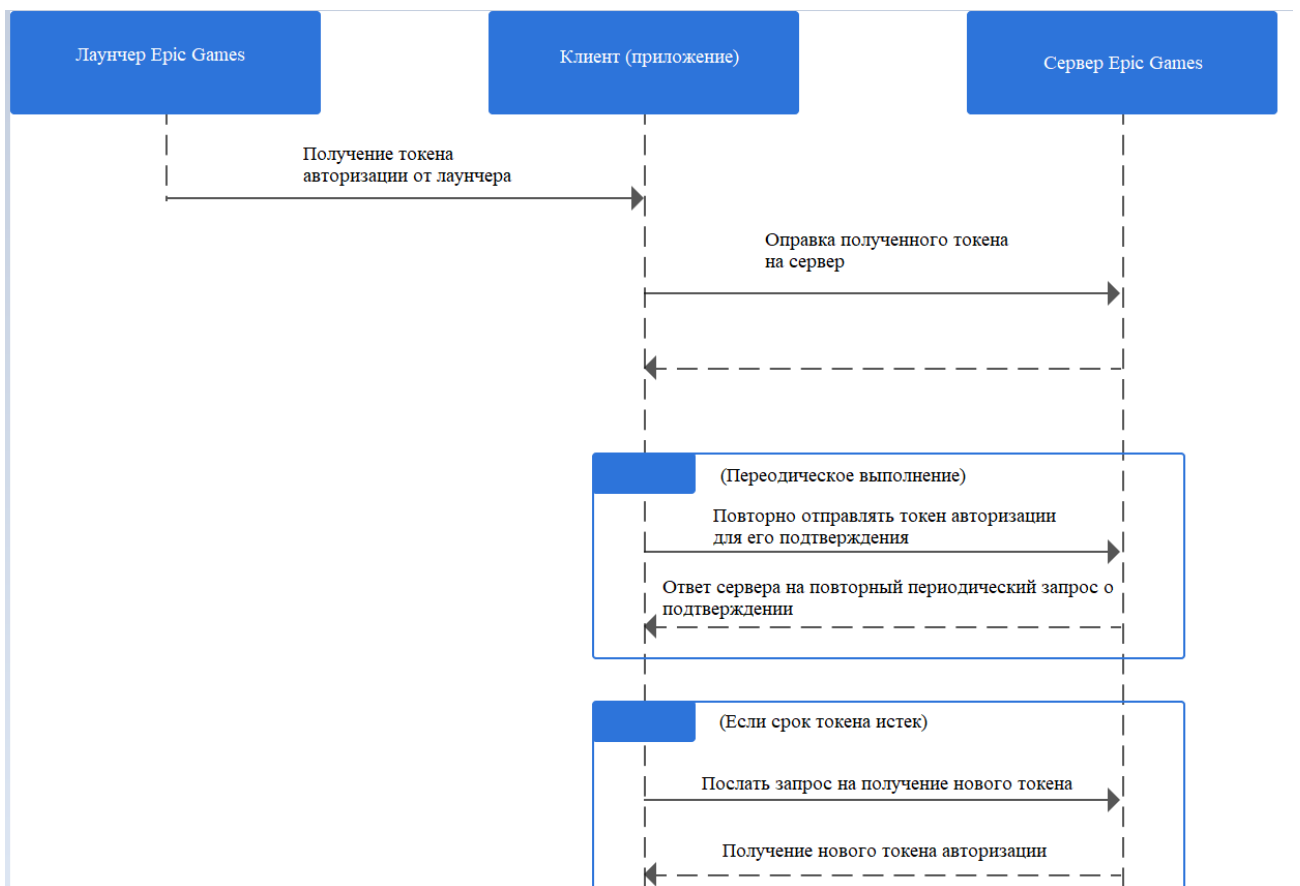


Рисунок 1.3 — Схема авторизации через лаунчер Epic Games

По схеме 1.3 видно, что сначала лаунчер отправляет клиенту игры при запуске токен авторизации через параметры запуска. Клиент получает данный токен и при авторизации посылает его серверу Epic Games. Сервер Epic Games в свою очередь проводит валидацию сгенерированного токена и отправляет обратно клиенту ответ. Если токен проходит валидацию, то пользователь успешно авторизуется в системе EOS.

EOSSDK самостоятельно периодически выполняет запросы на повторную проверку токена авторизации, который имеет определенный срок действия. Если срок токена истек, клиент самостоятельно формирует запрос серверу на получение нового токена, и сервер в свою очередь отправляет клиенту уже обновленный токен авторизации.

Схема работы с токеном работает аналогично для других типов авторизации.

1.1.2. Leaderboards Interface

Leader Boards Interface — программный интерфейс предоставляемый плагином Epic Games Services. Данный интерфейс дает возможность ранжировать результаты всей своей базы игроков, чтобы игроки могли соревноваться друг с другом за лучший результат. Каждая игра может поддерживать несколько таблиц лидеров, собирать баллы из разных источников и ранжировать их с помощью различных режимов подсчета очков. Данная фишка многопользовательской игры отлично подходит для такого жанра игр, как многопользовательский шутер и при дальнейшей работе будет интегрирована в проект.

Для того, чтобы начать начислять пользователям очки за определенные игровые успехи, на ресурсе DevPortal необходимо создать статистику, по которой будет отслеживаться успех каждого пользователя, а также необходимо проинициализировать саму таблицу, где будут показываться актуальные результаты наилучших игроков (рисунок 1.3).

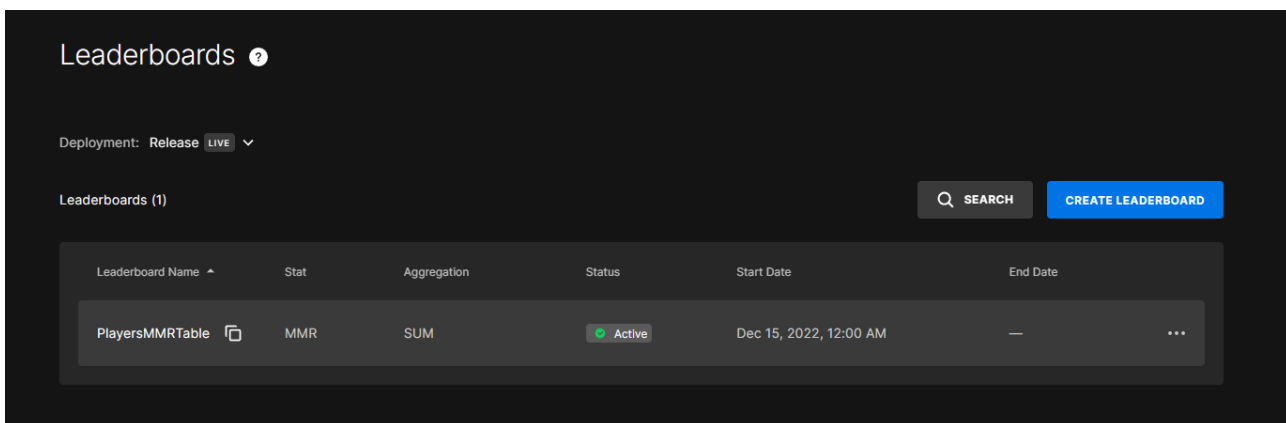


Рисунок 1.3 — Панель управления таблиц лидеров

Для созданной таблицы лидеров всегда можно просмотреть ее содержимое (рисунок 1.4).

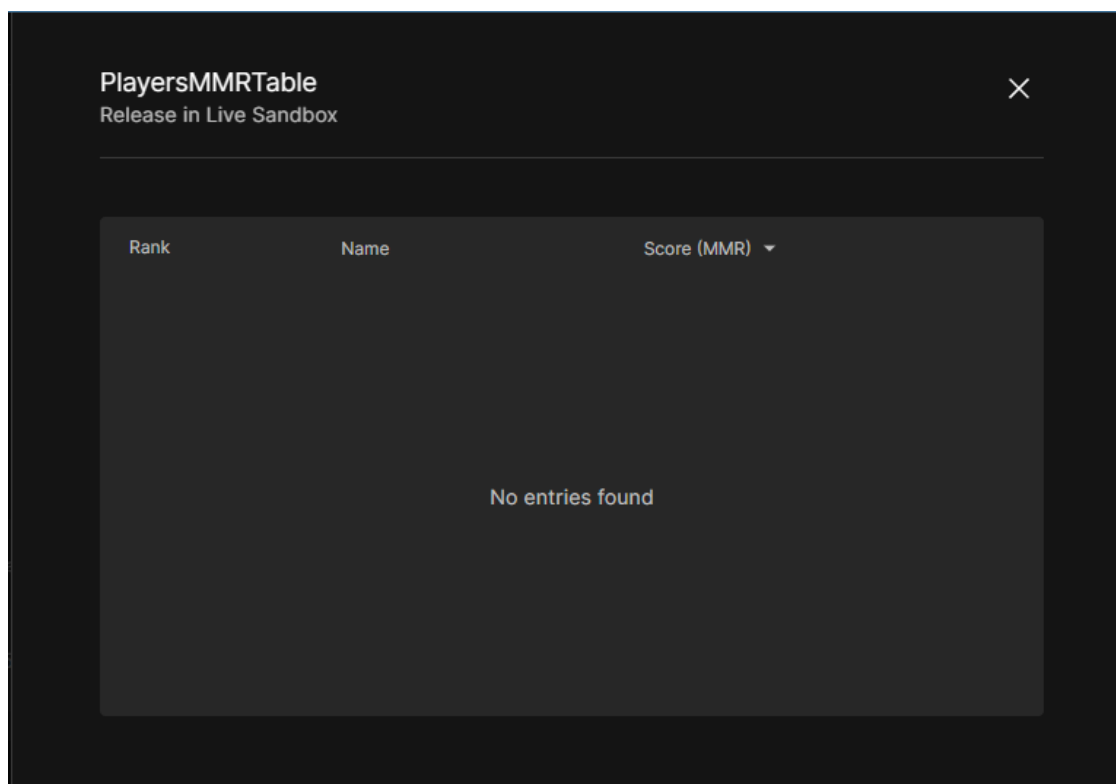


Рисунок 1.4 — Просмотр содержимого существующей таблицы лидеров

Для того, чтобы рассчитывать статистику, по которой будет происходить сортировка таблицы, необходимо разработать собственный метод начисления очков или же выбрать существующую систему начисления очков, на которую опираются большинство современных многопользовательских игр.

1.1.3. Lobbies Interface

В любой многопользовательской игре перед началом матча набирается нужное количество игроков. После подключения к еще не начавшейся игре все игроки должны помещаться в лобби — отдельный игровой уровень, где уже подключившиеся игроки ожидают других для набора нужного количества игроков.

Как уже было сказано ранее, в игре лобби представляет собой отдельный игровой уровень. В этом уровне игроки могут спокойно перемещаться по карте, дожидаясь остальных. Как правило, игра не начинается, пока не набралось нужное количество игроков или не истек таймер до начала матча.

Epic Games Services предоставляет программный интерфейс для управления списками лобби. На уровне EOS лобби — это сущность, которая содержит в себе информацию о предстоящем матче.

В функционал Lobbies Interface входят следующие возможности:

1. Создание и удаление существующего лобби.
2. Поиск лобби и присоединение к найденной игре.
3. Приглашение в существующее лобби игроков из списка друзей.
4. Удаление игроков, находящихся в лобби.

Также Lobbies Interface может предоставить существующую информацию о любом лобби: ID лобби, владелец лобби, максимальное количество игроков, количество свободных слотов, настройки игры и т.п.

В DevPortal после авторизации игрока и создания игровой сессии во вкладке Epic Games Services / Lobbies можно посмотреть информацию о созданной игре (рисунок 1.5). Также данная информация доступна из API Lobbies Interface.

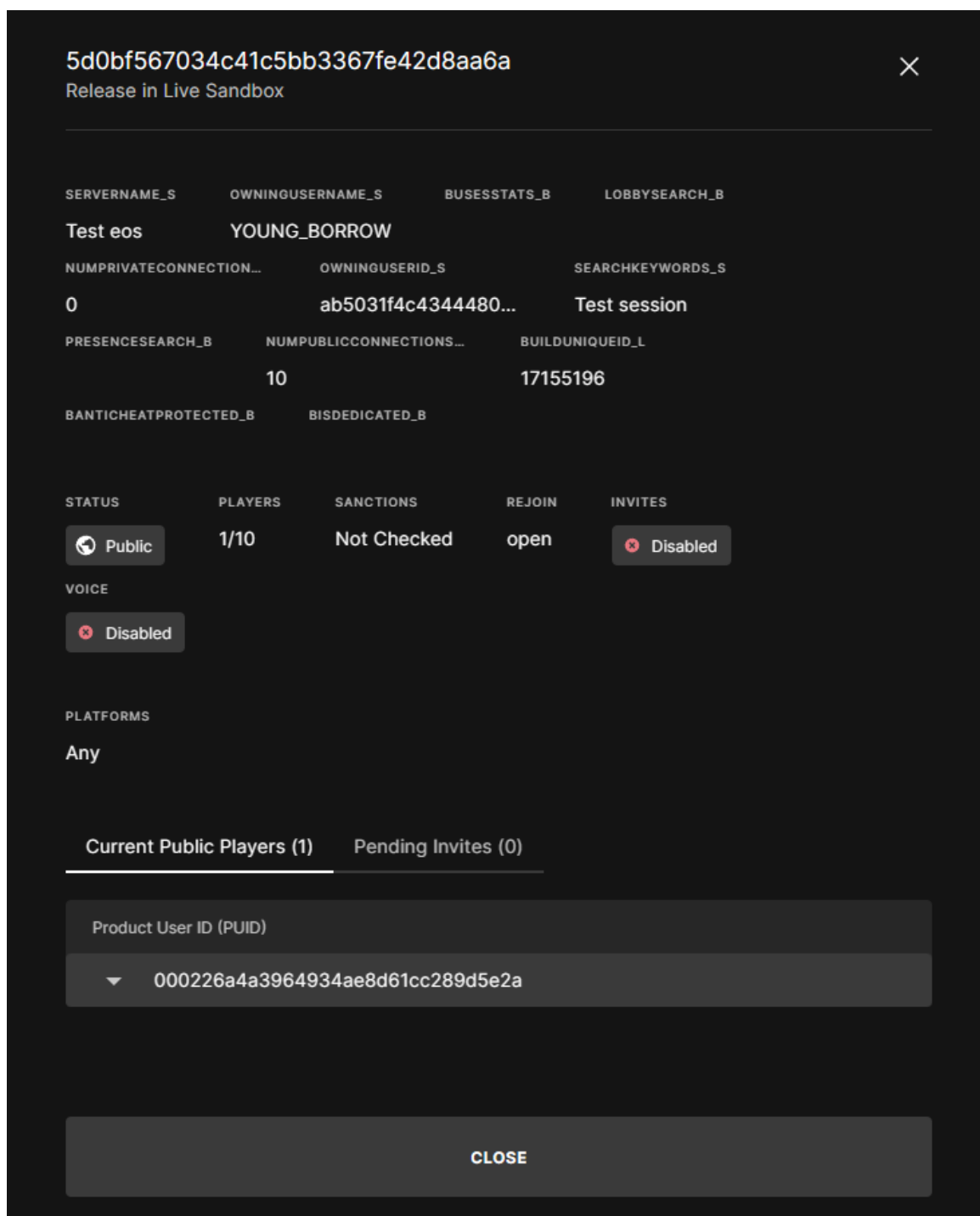


Рисунок 1.5. — Информация о настройках и состоянии созданный игры

Для того, чтобы начать использовать вышеописанный программный интерфейс в проекте необходимо в движке UE4 создать отдельный игровой уровень, который будет представлять из себя простое лобби, а также внедрить программные интерфейс Lobbies Interface в проект.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1. Создание клиентов и политик для авторизации пользователя

Перед тем, как использовать возможности EOS, для начала необходимо на DevPortal создать приложение, которое будет использовать EOS. После создания приложения необходимо создать тип клиента, который будет использовать данное приложение, и политику приложения, которое определяет доступные клиенту возможности такие, как доступ к системе достижений, системе таблиц лидеров, системе матчмейкинга и т.д.

При создании политики приложения необходимо указать, какие возможности может использовать клиент. Для простоты можно указать, чтобы у клиента был доступ ко всем возможностям EOS. Ниже на рисунке 2.1 представлен процесс создания политики для клиентов.

New client policy [X]

Client policy type *
Custom

Client policy type contains a predefined set of permissions for different uses.

CONDITIONS

☒ **User required**
Checking this requires a logged-in user to execute an action with this policy

FEATURES

Feature	Actions Enabled	Toggle
Achievements	6 actions enabled	On
Connect	All actions disabled	Off
Leaderboards	2 actions enabled	On
Lobbies	5 actions enabled	On
Matchmaking	10 actions enabled	On
Metrics		

Configurable actions for the Matchmaking service.

Action	Checked
status Gets global capacity information for a deployment	<input checked="" type="checkbox"/>
findInvitesByUserId Read all pending invites for signed in user	<input checked="" type="checkbox"/>
findSessions Find all public sessions	<input checked="" type="checkbox"/>
findSessionsByUserId Read all sessions the signed in user is in	<input checked="" type="checkbox"/>
sendInvite Send an invite	<input checked="" type="checkbox"/>

CLOSE **SAVE & EXIT**

Рисунок 2.1 — Создание политики приложения

После создания политики, необходимо создать клиента, указав его имя и политику приложения, которую он будет использовать. Ниже на рисунке 2.2 представлен процесс создания клиента приложения.

New client

Client Name *

EOS_GeneralClient

Client policy determines the level of access for different features.

Client policy *

Peer2Peer Client Policy

Test Policy 2

Peer2Peer Client Policy

Client Test Policy

+ Add new client policy


Redirect URL

+ ADD ANOTHER



Рисунок 2.2 — Создание клиента приложения

После того, как были созданы клиент и политика приложения, можно воспользоваться всеми данными приложения, которые доступны в DevPortal, а именно ID приложения, ID клиента, Sandbox ID и т.д. Все необходимые данные для приложения для использования EOS представлены ниже на рисунке 2.3.


PRODUCT

Product	Product ID
OnlineMultiplayerShooter	8bd04340aae74d73b3ea092484309e0d 


CLIENTS

Custom	Client ID	Client Secret
NewClient1	xyza7891o7wcWqcrWKfjyH216l7wCdl8 	***** 

APPLICATIONS

Application name	Application ID
OnlineMultiplayerShooter	fghi4567uFCqHVcHCYnfXR1GnrWES6m1 

SANDBOXES

Sandbox type	Sandbox ID
Live	080960589ec84e44aaf746e6c31bd352 

DEPLOYMENTS


Deployment name	Sandbox	Deployment ID
Release	Live	0a33e5dadf2c410281355e84fe717854 

Рисунок 2.3 — Данные приложения, необходимые для EOS

Для того, чтобы использовать возможности EAS, на DevPortal также необходимо произвести настройки приложения во вкладке Epic Account Services Settings. В данном разделе необходимо указать настройки, как показано на рисунке 2.4.

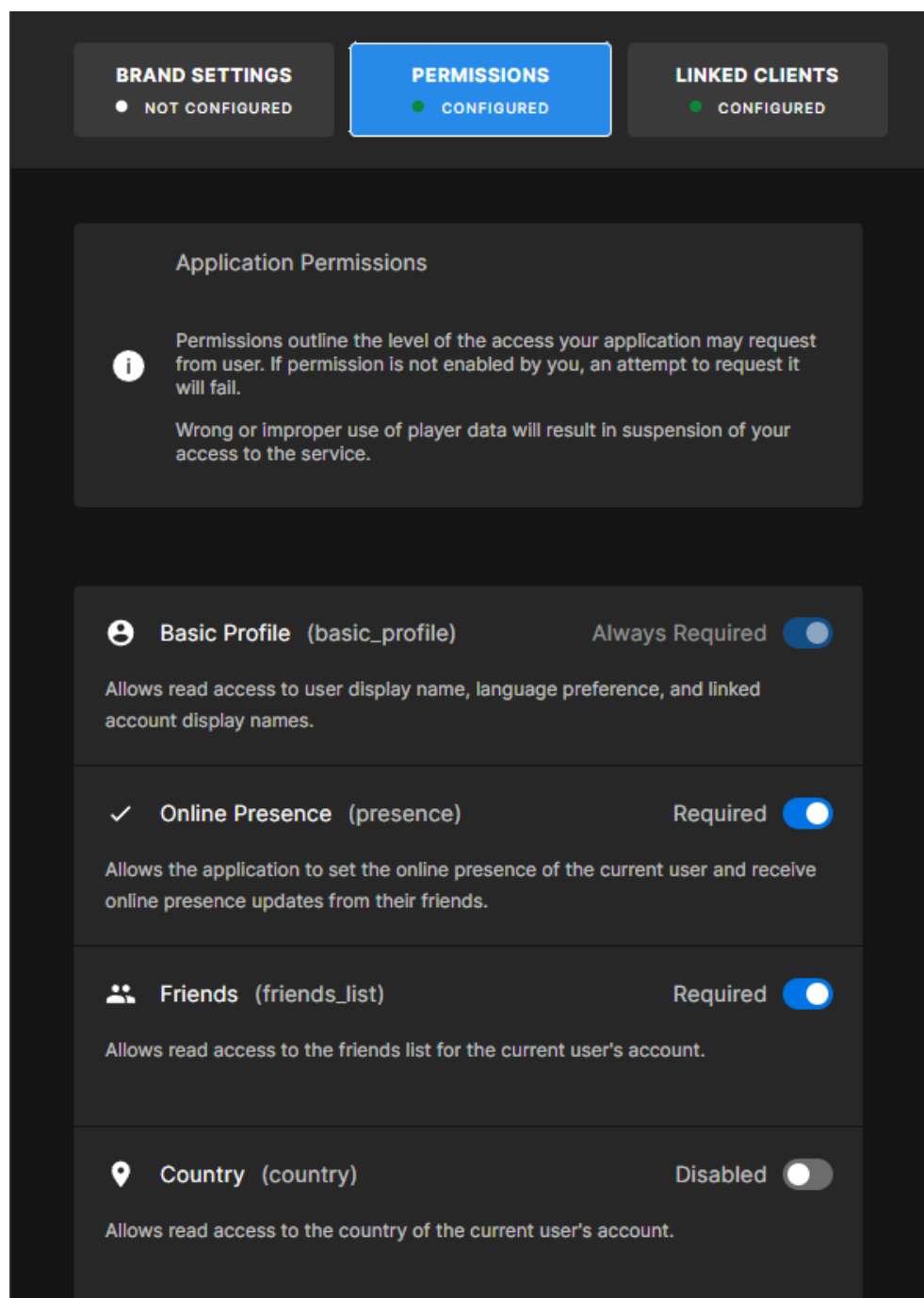


Рисунок 2.4 — Настройка приложения для EAS

2.2. Авторизация пользователя с помощью данных аккаунта Epic Games

Чтобы использовать интерфейс авторизации EAS, для начала необходимо добавить EOSSDK в проект. Для этого добавим EOSSDK в массив зависимостей проекта.

Листинг 2.1 — Настройка зависимостей проекта

```
PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject", "Engine",  
"InputCore", "HeadMountedDisplay", "UMG", "XmlParser", "OnlineSubsystemEOS",  
"OnlineSubsystem", "EngineSettings", "EOSSDK" });
```

Также в настройках проекта в UE4 Editor в секции плагинов указываем, что проект использует EAS.

Для того, что использовать интерфейс авторизации для начала необходимо инициализировать SDK. Для этого реализуем функцию, которая проводит инициализацию SDK (листинг 2.2).

Листинг 2.2 — Инициализация SDK

```
void ULab4GameInstance::InitializeSDK()  
{  
    EOS_InitializeOptions SDKOptions;  
    SDKOptions.ApiVersion = EOS_INITIALIZE_API_LATEST;  
    SDKOptions.AllocateMemoryFunction = nullptr;  
    SDKOptions.ReallocateMemoryFunction = nullptr;  
    SDKOptions.ReleaseMemoryFunction = nullptr;  
    SDKOptions.ProductName = "OnlineMultiplayerShooter";  
    SDKOptions.ProductVersion = "1.0";  
    SDKOptions.Reserved = nullptr;  
    SDKOptions.SystemInitializeOptions = nullptr;  
    EOS_EResult SDKInitializeResult = EOS_Initialize(&SDKOptions);  
  
    if (SDKInitializeResult == EOS_EResult::EOS_Success)  
    {  
        UE_LOG(LogTemp, Warning, TEXT("SDK success"));  
    } else if (SDKInitializeResult == EOS_EResult::EOS_AlreadyConfigured)  
    {  
        UE_LOG(LogTemp, Warning, TEXT("SDK already configured"));  
    }  
    else  
    {  
        UE_LOG(LogTemp, Warning, TEXT("SDK another error"));  
    }  
}
```

Для инициализации SDK зададим структуру *EOS_InitializeOptions* *SDKOptions*. Самыми важными параметрами являются поля *ApiVersion*, которое всегда задается константой *EOS_INITIALIZE_API_LATEST*, *ProductName*, которое должно совпадать с названием продукта на DevPortal, а также

EOS_Platform_Create(&EOS_Platform_Options), которая принимает структуру *EOS_Platform_Options* с данными приложения.

Также после инициализации Platform Interface необходимо выполнять функцию *EOS_Platform_Tick(&EOS_HPlatform)*, чтобы функции EOSSDK для авторизации, обновления токенов и выхода могли выполнять работу.

Так как работа с авторизацией производится в сущности класса UGameInstance, который является производным от класса UObject и не имеет виртуальной функции Tick, как класс AAActor, необходимо создать собственную циклическую функцию, которая будет выполняться на протяжении работы приложения. Реализация данной функции представлена ниже в листинге 2.4.

Листинг 2.4 — Реализация функции Tick для EOS_Platform_Tick

```
void ULab4GameInstance::Init()
{
    TickDelegateHandle =
    FTicker::GetCoreTicker().AddTicker(FTickerDelegate::CreateUObject(this,
    &ULab4GameInstance::Tick), 0.1f);
}

void ULab4GameInstance::Shutdown()
{
    FTicker::GetCoreTicker().RemoveTicker(TickDelegateHandle);
    Super::Shutdown();
}

bool ULab4GameInstance::Tick(float DeltaSeconds)
{
    if (PlatformInterface != nullptr)
    {
        EOS_Platform_Tick(PlatformInterface);
    }
    return true;
}
```

После того, как все подготовительные работы для авторизации были проведены, необходимо реализовать функцию по авторизации пользователя. Реализация данной функции приведена ниже в листинге 2.5.

Листинг 2.5 — Реализация функции авторизации пользователя

```
void ULab4GameInstance::InitializeAuthInterface()
{
    AuthInterface = EOS_Platform_GetAuthInterface(PlatformInterface);

    if (AuthInterface == nullptr)
    {
        UE_LOG(LogTemp, Error, TEXT("AuthInterface has not been initialized"));
        return;
    }

    TArray<FText> UserCredentials = m_pMainMenu->GetCredentials();
    FString UserEmail = UserCredentials[0].ToString();
    FString UserPassword = UserPassword[1].ToString();

    EOS_Auth_LoginOptions LoginOptions;
    EOS_Auth_Credentials AuthCredentials;

    LoginOptions.ApiVersion = EOS_AUTH_LOGIN_API_LATEST;
    AuthCredentials.Id = *UserEmail;
    AuthCredentials.Token = *UserPassword;
    AuthCredentials.ApiVersion = EOS_AUTH_CREDENTIALS_API_LATEST;
    AuthCredentials.Type = EOS_ELoginCredentialType::EOS_LCT_Password;
    AuthCredentials.SystemAuthCredentialsOptions = nullptr;
    LoginOptions.Credentials = &AuthCredentials;

    EOS_Auth_Login(AuthInterface, &LoginOptions, nullptr, &CompletionDelegate);
}
```

Функция *EOS_HAuth EOS_Platform_GetAuthInterface(&EOS_HPlatform)* дает доступ к интерфейсу авторизации, для которого необходим объект типа *EOS_HPlatform*, полученный на предыдущем шаге.

Для авторизации типа *EOS_LCT_Password* в качестве данных пользователя необходимы электронная почта и пароль от аккаунта пользователя. Важно знать, что с помощью данного типа авторизации невозможно провести авторизацию аккаунта с двухфакторной аутентификацией, о чем написано в документации Epic Games.

В структуру *EOS_Auth_Credentials* записываем следующие обязательные поля:

- Id — Id пользователя. Для данного типа авторизации в данное поле записывается электронная почта пользователя.
- Token — в данное поле для данного типа авторизации записывается пароль пользователя
- Type — тип авторизации. Перечисление *EOS_ELoginCredentialType* содержит в себе множество типов авторизации. Для авторизации с помощью электронной почты и пароля от УЗ Epic Games используется *EOS_ELoginCredentialType::EOS_LCT_Password*.

Также, для работы функции авторизации пользователя необходимо определение функции *static void EOS_CALL onLoginConnect(const EOS_Connect_LoginCallbackInfo* Data)*. Данная функция принимает в себя единственный параметр *EOS_Connect_LoginCallbackInfo* Data* — структура, которая включает в себя данные о попытке авторизации пользователя после вызова функции *EOS_Auth_Login*. Реализуем тело данной функции, чтобы обрабатывать информацию об авторизации пользователя (листинг 2.6).

Листинг 2.6 — Реализация функции CompletionDelegate

```
void EOS_CALL ULab4GameInstance::CompletionDelegate(const EOS_Auth_LoginCallbackInfo*
Data)
{
    if (Data->ResultCode == EOS_EResult::EOS_InvalidUser) {
        UE_LOG(LogTemp, Warning, TEXT("Authentication error"));
        GEngine->AddOnScreenDebugMessage(-1,
        3.f,
        FColor::Red,
        FString::Printf(TEXT("User is invalid")),
        FVector2D(3.f)
        );
    }
    if (Data->ResultCode == EOS_EResult::EOS_Success) {
        UE_LOG(LogTemp, Warning, TEXT("User authenticated successfully"));
        GEngine->AddOnScreenDebugMessage(-1,
        3.f,
        FColor::Green,
        FString::Printf(TEXT("Logged in successfully")),
        true,
        FVector2D(3.f)
    )
    }
```



```
);  
}  
}
```

Функция *CompletionDelegate* является callback-функцией, так как она передается в качестве параметра в другую функцию. В данной функции обрабатывается поле структуры *Data->ResultCode*. В зависимости от результата во вьюпорт выводится сообщение об успешном или неуспешном выполнении авторизации. Также в данной структуре содержатся поля *EOS_ProductUserId LocalUserId* и *EOS_ContinuanceToken ContinuanceToken* — Id авторизованного пользователя и токен авторизации соответственно, который был сгенерирован сервисами Epic Games. Данные поля также можно валидировать и использовать при работе с другими интерфейсами, предоставляемыми EAS.

Для того, чтобы пользователь мог производить авторизацию на уровне приложения были созданы виджеты, с помощью которых можно выбрать способ авторизации и ввести данные для УЗ Epic Games (рисунок 2.5 и рисунок 2.6 соответственно).

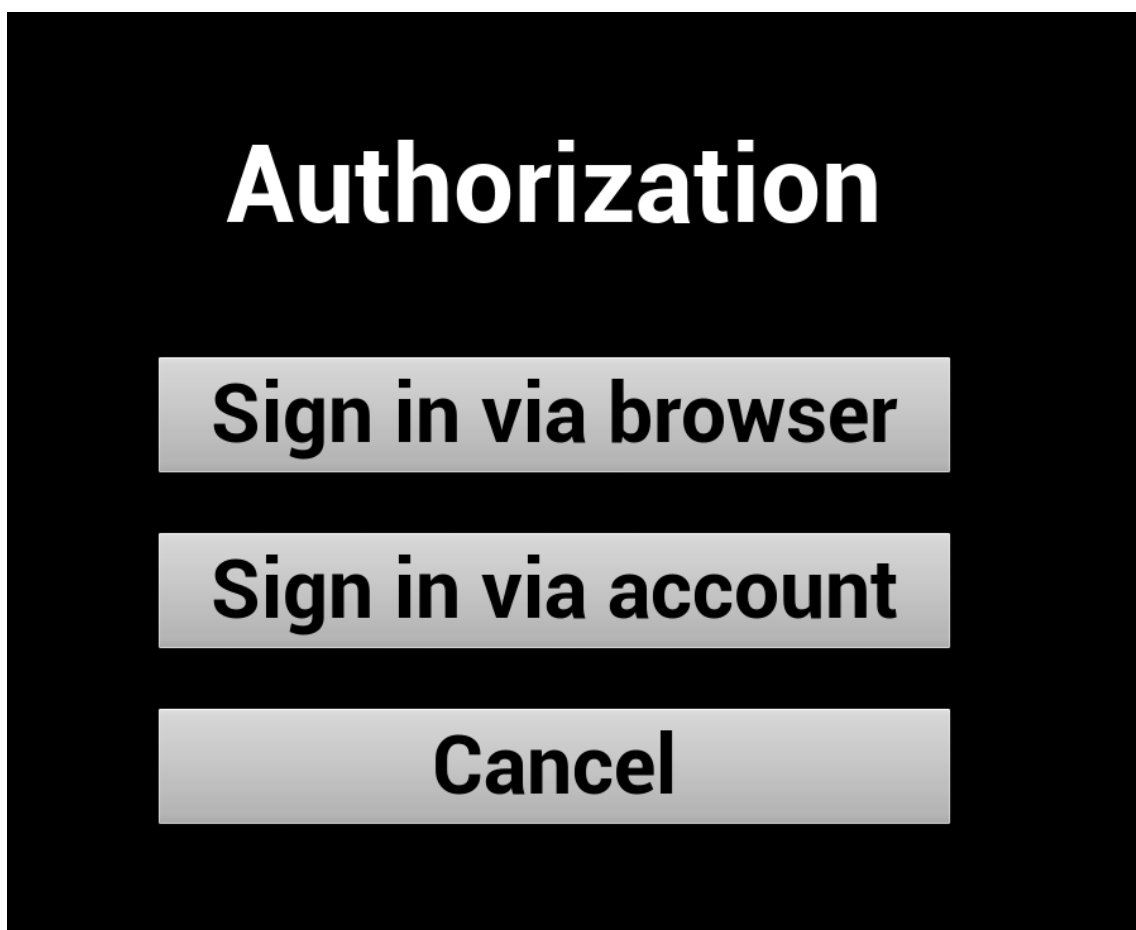


Рисунок 2.5 — Меню выбора авторизации пользователя

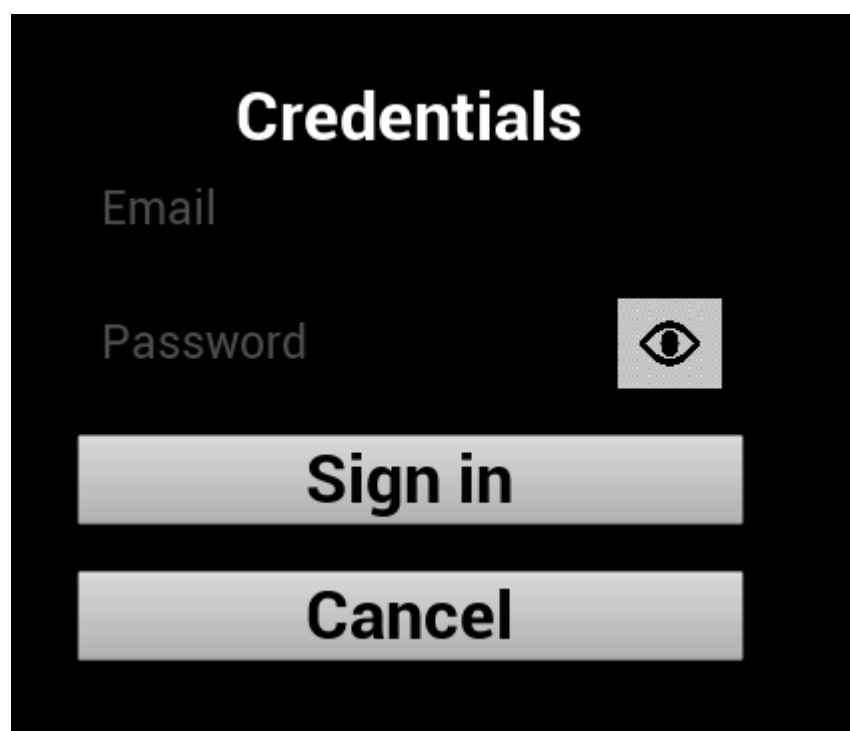
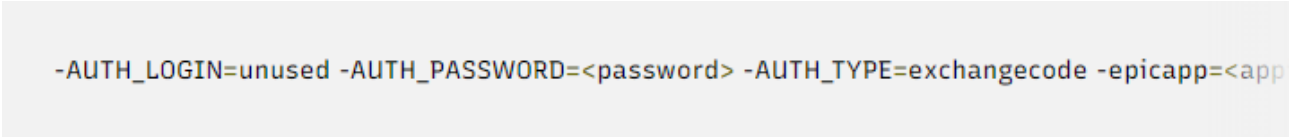


Рисунок 2.6 — Виджет ввода данных для УЗ Epic Games

2.3. Авторизация пользователя с помощью лаунчера Epic Games

Другим типом авторизации пользователя является тип авторизации через лаунчер Epic Games. Данный способ является предпочтительным способом авторизации, когда готовая версия игры уже добавлена в Epic Games Store. Данный способ авторизации будет рассмотрен и использован в дальнейшей работе над проектом для улучшения его функционала, так как данный способ не имеет никаких ограничений и поддерживает двухфакторную аутентификацию пользователя.

Авторизация данным способом осуществляется аналогично способу, описанному выше. Отличительной чертой является заполнение структуры типа *EOS_Auth_Credentials*: поле *Id* остается пустым, в поле *Token* записывается токен авторизации, который игра получила через аргументы запуска командной строки при старте от лаунчера Epic Games (рисунок 2.7). В поле *Type* записывается значение перечисления *EOS_ELoginCredentialType::EOS_LCT_ExchangeCode*.



```
-AUTH_LOGIN=unused -AUTH_PASSWORD=<password> -AUTH_TYPE=exchangeCode -epicapp=<app
```

Рисунок 2.7 — Параметры командной строки при запуске игры из лаунчера Epic Games

Для получения аргументов запуска приложения вызовем функцию *FCommandLine::Get* (листинг 2.7).

Листинг 2.7 — Получение токена авторизации приложения

```
FString Token;  
FParse::Value(FCommandLine::Get(), TEXT("AUTH_PASSWORD"), Token);
```

После получения данных можно вызывать данный метод авторизации.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были решены следующие задачи:

1. Изучены концепции сетевого программирования в UE 4, для расширения и улучшения игрового процесса.
2. Реализован алгоритм авторизации пользователя внутри приложения двумя способами с помощью плагина EAS.
3. Разработан виджет для ввода данных пользователя от УЗ Epic Games.
4. Проведен анализ существующих программных интерфейсов, которые в дальнейшем будут интегрированы в проект для улучшения и расширения игрового процесса.

Во время выполнения практических задач были улучшены необходимые в разработке навыки чтения и понимания официальной документации, улучшены навыки программирования на ЯП C++, закреплены базовые знания работы с игровыми сессиями и знания сетевого программирования в UE 4, закреплены навыки по работе и настройке приложения на DevPortal.

Полученные знания о реализации и интеграции дополнительных программных интерфейсов EOS будут применены в дальнейшей работе над шаблоном многопользовательской игры для добавления в его функционал новых возможностей современных многопользовательских игр.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Services Documentation. Unreal Engine Documentation.
URL: <https://dev.epicgames.com/docs>. Дата обращения (25.11.2022);
2. Auth Interface. Unreal Engine Documentation.
URL: <https://dev.epicgames.com/docs/epic-account-services/auth-interface>.
Дата обращения (26.11.2022);
3. Platform Interface. Unreal Engine Documentation.
URL: <https://dev.epicgames.com/docs/game-services/eos-platform-interface>.
Дата обращения (26.11.2022);
4. EOS Game Services. Unreal Engine Documentation.
URL: <https://dev.epicgames.com/docs/game-services>.
Дата обращения (26.11.2022);
5. EOS Account Services. Unreal Engine Documentation.
URL: <https://dev.epicgames.com/docs/epic-account-services>.
Дата обращения (27.11.2022);
6. EOS SDK Errors Code. Unreal Engine Documentation.
URL: <https://dev.epicgames.com/docs/epic-online-services/sdk-error-codes>.
Дата обращения (01.12.2022);
7. Уильям Шериф. Unreal Engine 4.x Scripting with C++ Cookbook / Уильям Шериф, Стивен Уиттл, Джон Доран. — Packt Publishing, 2019 г. — 708 с.
8. Арам Куксон. Unreal Engine 4 Game Development in 24 Hours, Sams Teach Yourself / Арам Куксон, Райан Даулингсока, Клинтон Крамплер. — Москва: Бомбора, 2019 г. — 528 с.

9. Маркус Померу. Blueprints Visual Scripting for Unreal Engine 2nd Edition / Маркус Померу, Брендэн Сьюэлл. — Packt Publishing, 2019 г. — 380 с.