НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра
спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Система збору аналітики додатку для розробників ігор»

Виконав (-ла):
студент (-ка) IV курсу, групи ТР-71
Редько Владислав Ігорович _____ _____

Керівник:
Професор, доктор технічних наук, доцент
Аушева Наталія Миколаївна _____ _____

Рецензент:
_____
_____ _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2021 року

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 122 «Комп’ютерні науки»

освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ

на дипломну роботу студенту

Редько Владиславу Ігоровичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Система збору аналітики додатку для розробників ігор”

керівник роботи д.т.н. Аушева Наталія Миколаївна
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи набір аналітичних даних, які описують успішність розробленої гри
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуюче програмне забезпечення та методи для збору аналітичних даних, обґрунтувати обрані засоби та методи для програмної реалізації, , розробити програмний продукт для збору аналітичних даних, та відображення їх у окремому додатку
5. Перелік ілюстративного матеріалу
1. Приклад використання аналітичних систем. 2. Маркетингові методи для збору аналітичних даних. 3. Приклади аналітичних систем. 4. Засоби розроблення системи. 5. Архітектура системи та структура компонентів системи 6. Інтерфейс користувача.
6. Дата видачі завдання ”___”_____202___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі		
3.	Розробка архітектури та загальної структури системи		
4.	Підготовка матеріалів		
5.	Програмна реалізація системи		
6.	Оформлення пояснювальної записки		
7.	Захист програмного продукту		
8.	Передзахист		
9.	Захист		

Студент _____ Редько В. І.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Аушева Н. М.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Аналітика – інструмент для розроблення та правильного підтримання програмних продуктів. Системи аналітики є актуальними для усіх пост-релізних етапах розробки гри – від збору аналітики для визначення залученості гравців на початкових етапах, до збору аналітики для розробки оновлень.

Мета роботи – проаналізувати існуючі системи аналітки, та на основі дослідження розробити систему аналітики, яка б легко інтегрувалась в будь-яку гру, розроблену за допомогою C#, та відображала дані в зручному для розробника вигляді.

Записка містить 64 сторінки, 29 рисунків, 4 додатки та 14 посилань.

Ключові слова: АНАЛІТИКА, СЕГМЕНТАЦІЯ, РУШІЙ, СЕСІЙНІ ТА ПОДІЄВІ СИСТЕМИ, АНАЛІТИЧНІ ПОКАЗНИКИ, ПРОГНОЗУВАННЯ МЕТРИК ПРОДУКТУ

ABSTRACT

Analytics is a tool for developing and properly maintaining software products. Analytics systems are relevant for all post – release stages of game development-from collecting analytics to determining player engagement in the initial stages, to collecting analytics to develop updates.

The aim of the work is to analyze existing analytics systems, and based on the research, develop an analytics system that would easily integrate into any game developed using C#, and display data in a convenient form for the developer.

The note contains 64 pages, 29 Figures, 4 attachments and 14 links.

Keywords: ANALYTICS, SEGMENTATION, MOVEMENTS, SESSION AND EVENT SYSTEMS, ANALYTICAL INDICATORS, PRODUCT METRIC FORECASTING

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

UI – User Interface (інтерфейс користувача).

VS – Visual Studio (середовище програмування розроблене компанією Microsoft для платформи .Net).

SDK – Software Development Kit(Набір інструментів розробки)

Unity – кросплатформене середовище розробки ігор та додатків

TCP – transmission Control protocol(протокол передачі даних)

Retention – утримання(одна з метрик аналітичних систем для аналізу успішності гри)

API – Application programming interface(опис методів, процедур та функцій за якими одна програма може взаємодіяти з іншою)

ВСТУП

Сучасна ігрова індустрія ділиться на декілька відокремлених ринків — мобільний, соціальний та ринок комп’ютерних ігор. І хоча вони дуже відрізняються, на жодному з них не можна випустити успішну гру без використання системи аналітики.

Хоча ігри створюються для того щоби розважати гравців, основним завданням гри, як і будь-якого іншого продукту, є заробіток грошей. А на ринку ігор найбільше грошей заробляють безкоштовні ігри. Такі продукти починають заробляти гроші не під час свого виходу до магазинів, а пізніше — коли наберуть аудиторію й будуть заробляти за допомогою внутрішньоігрових покупок. Саме тому ще на етапі бета-тестів потенціальний заробіток потрібно рахувати не за кількістю завантажень із магазинів ігор, а по захопленості гравців до гри.

Захопленість є ключовим компонентом ефективного продукту. Якщо гравець не бере участь у грі, то він не бачить ніякої особистої цінності та будь-яка спроба продати йому щось буде приречена на провал. Чим більше часу людина проводить у грі та чим більше вона йому подобається — тим вище ймовірність того, що одного разу він зробить внутрішньоігрову покупку. [1]

Захопленість людей можна оцінити багатьма параметрами, але є декілька основних — кількість часу, який гравець витрачає на гру, кількість сесій у день, відвал гравців на перший, сьомий та тридцятий день.

У разі продовження підтримування гри, цілі аналітики розширюються. Після остаточного релізу, у гру починають додаватися монетизаційні механіки, а за допомогою аналітики можна отримати детальні відгуки про їхню успішність, та скорегувати шляхи розвитку продукту.

Успішне підтримування проєкту полягає не тільки у монетизації, але і в утриманні_цікавості гравців упродовж років. У разі правильного налаштування, аналітика буде показувати всі зміни в динаміці параметрів проєкту, що дасть змогу

випускати відповідні контентні оновлення — оновлення спрямовані на створення нового досвіду та утримання гравців у грі.[2]

Під час проходження практики була розроблена система аналітики для розробників ігор. Створений продукт легко інтегрується в будь-яку гру розроблену за допомогою Unity, надає змогу збирати як основні метрики — активність користувачів, кількість покупок, Retention в грі та кількість сесій, так і специфічні для кожної гри — кількість пройдених рівнів, кращий рахунок за матч, загальний отриманий досвід.

Розроблена система поділяється на 4 елементи — бібліотека для інтеграції в гру, сервер для обробки підключень та розподілу інформації, база даних для збереження даних аналітики, мобільний додаток для розробників, який показуватиме дані аналітики.

Інтегрування системи проходить у два етапи:

— Реєстрація у мобільному додатку та створення сторінки гри

— Інтегрування бібліотеки до гри та додавання ідентифікатору гри до проєкту

Після інтегрування розробник може налаштовувати параметри, з яких він хоче збирати аналітику.

Для мобільного ринку й розроблення ігор на рушії Unity існують багато систем аналітики — DevToDev, GoogleAnalytics, GameAnalytics, Flurry. І хоча більшість із них надає змогу збирати потрібні дані, у системах існують недоліки — ціна, обмеженість у параметрах, складність інтеграції в гру, час, за який можна одержати дані.

Під час використання розробленої системи розробник отримає доступ до аналітики своїх продуктів у вигляді графічних та текстових даних, на своєму смартфоні.

1. ПОСТАНОВКА ЗАДАЧІ РОЗРОБЛЕННЯ СИСТЕМИ АНАЛІТИКИ ДОДАТКУ ДЛЯ РОЗРОБНИКІВ ІГОР

Метою дипломної роботи є розробка системи аналітики додатків, які зможуть використовувати розробники ігор, дослідження існуючих методів збору аналітики , аналіз існуючих аналітичних систем. Розроблена система має включати можливість аналізувати основні показники проєкту, легко інтегрувалась в будь-яку гра написану на C# і мала зручний мобільний додаток для відстеження зібраних даних.

Для розв’язання поставленої задачі були сформовані наступні завдання:

- Проаналізувати існуючі системи аналітики додатків для розробників ігор
- Проаналізувати доступні маркетингові методи для збору аналітичних даних
- Обрати засоби розробки програмного забезпечення, на основі аналізу методів та систем
- Створити структуру системи, для зручного розроблення програмного продукту
- Розробити усі модулі та поєднати їх у єдину систему аналітики

1.1 Призначення розробленої системи

Ігрові студії часто використовують аналітику як перший крок до створення нових проєктів. Сучасні безкоштовні ігри будуються таким чином: для залучення гравців геймдизайнери та продюсери створюють нові різновиди розваг, але механіки збільшення надходжень переносяться від проєкту до проєкту та покращуються. Такий підхід дає максимальний прибуток із мінімальними ризиками.

Під механіками монетизації, які переносяться між проєктами, можна вважати колесо фортуни, щоденні нагороди, накопичувальна скриня, спеціальні акції,

початкова пропозиція. Аналізуючи дані з попередніх проєктів, розробники можуть успішно використовувати та оновлювати ці механіки, або будь-які інші.

Останні роки собівартість розроблення ігор підіймається, і розробники намагаються знайти нові способи заробітку. Одним із таких способів стала форма ігор під назвою “Game as a service”. Такі ігри підтримуються роками за допомогою контентних оновлень. Для успішного випуску таких оновлень треба не тільки розуміти аудиторію своєї гри, але й бачити успішності проходження контенту та захопленості гравців.

Але одним із головних досягнень стала система на основі збору аналітики — сегментація користувачів, яка є найбільш успішним підходом до торгівлі ігровими предметами. Сегментація гравців — це умовний поділ користувачів, залежно від їхніх покупок, який надає змогу створювати унікальні пропозиції для певних категорій гравців.

У методах сегментації, гравців традиційно поділяють на три категорії:

— “Мальки” — користувачі, які витрачають порівняно невеликі суми. За кількістю “мальків” набагато більше, ніж “китів” або “дельфінів”, але їхня частка в загальному доході набагато менше.

— “Дельфіни” — користувачі, які витрачають значні суми, але набагато менше ніж кити.

— “Кити” — їх менше, ніж усіх основних типів користувачів у проєкті, але вони приносять основний дохід.

У цих категорій гравців можуть бути різні показники Retention-у, кількості сесій, довжини сесій, витрат валюти та інші. Для досягнення максимального прибутку, усі ці категорії гравців треба оцінювати по-різному і працювати з ними по-різному.[3]

1.2 Задачі які розв’язує система

Завдання аналітики — збирати та показувати дані, які б допомогли покращити проєкт за різними показникам. Складно розповісти про конкретні задачі системи без контексту, тому краще буде розглянути декілька прикладів. Складність полягає в тому, що цей інструмент дуже корисний тільки за правильного налаштування та в досвідчених руках.

В ігровій індустрії є сутність під назвою “фан”(fun), або як його ще можна назвати — радість, хоча це й не зовсім так. Аналітика може показувати еквівалент “фану” у вигляді сукупності різних параметрів. Якщо це ігри в жанрі “merge”, то фан можна оцінити за допомогою часу витраченого на виконання завдання в сукупності з кількістю об’єднань предметів.

А в іграх жанру “солітер” гравцю потрібно зібрати всі картки з поля за обмежену кількість кроків. Для таких ігор часто використовують FUUU-показник. Цей показник відповідає за те, яку долю спроб гравець зробив і майже пройшов рівень. Наприклад, на рівні, де гравцю потрібно зібрати двадцять карток, він програв три рази, коли йому залишилося зібрати лише одну карту, і ще сім рази він програв на самому початку. Завдячуючи системі аналітики, геймдизайнер знає, що FUUU-показник замалий для рівня, і скоригує рівень. Приклад використання FUUU-показника в іграх жанру солітер приведено на рисунку 1.

Рисунок 1 - Приклад використання FUUU-показника

Якщо гравцю не вистачило одного кроку, то він із більшою ймовірністю купить запропонований у той же момент додатковий крок.

Більш загальним прикладом, не залежним від жанру, є використання певних механік, таких як колесо фортуни. Монетизація цієї механіки призначена для азартної частину аудиторії, які б крутили колесо за рекламу. Якщо ж гравці не використовують цю механіку, то на це може бути декілька причин: вона розташована в поганому місці, нагорода не виправдовує очікування або гравці не азартні. У цьому випадку, аналітика може вказати на правильний варіант виправлення механіки. До того ж, якщо виправлення механіки викликало зміни в економіці гри, то аналітика покаже, якщо ці зміни шкодять екосистемі гри.

Як можна побачити, в аналітики дуже багато застосувань, які здебільшого залежать від методу використання, аніж від системи. Система, яку використовують впливає в більшості на складність аналізу даних, їхню точність та складність налаштування звітів.[4]

Зважаючи на усе вищеописане, можна виділити основні задачі, яка має розв’язувати розроблена система:

— Збирати аналітичні дані з ігор

— Відображати дані в зручному вигляді

— Мати можливість сортувати дані за різними показниками

1.3 Модулі системи та їх призначення

Для виконання поставлених задач, система повинна мати наступні модулі:

— Бібліотека зв’язку з сервером — відповідає за зв’язок гри з сервером. Ця бібліотека інтегрується в гру як окремий файл, і може бути використана для з’єднання з сервером, відправлення даних на сервер та приймання даних.

— Мобільний додаток для розробників, який показує дані аналітики. Цей додаток використовує систему авторизації, яка запобігає потраплянню аналітичних даних гри до інших користувачів. Після авторизації, розробнику

будуть доступні усі зібрані дані по його проєктам. Дані демонструються по проєктах, параметрах, та часовим проміжках, а також мають декілька режимів сортування. Дані показуються у графічному та текстовому вигляді, що дозволяє порівнювати їх між собою у межах однієї вибірки.

— Сервер для обробки підключень та розподілу інформації. Сервер працює одразу у двох напрямках — на з'єднання з грою, і з'єднання з мобільним додатком.

— Взаємодія з грою — сервер встановлює підключення з кожним окремим гравцем і приймає від нього дані.

— Взаємодія з додатком — встановлює підключення з кожним окремим розробником, приймає запити на отримання даних, створює запити до БД і передає отримані дані до додатку.

— База даних для збереження даних аналітики. Приймає запити від серверу віддає потрібні дані. Має таблиці розробників, ігор та параметрів аналітики, а також додаткові таблиці для жанрів ігор і таблиці для типів гравців.

Система потребує вхідні дані:

— для авторизації в мобільному додатку, такі як унікальне ім'я користувача та пароль.

— в грі потрібно визначити параметри, які будуть відправлятись на сервер. Це можна зробити під час налаштування гри, або дані будуть створюватись на етапі виконання програмного продукту.

2. МЕТОДИ АНАЛІТИЧНОГО АНАЛІЗУ

Системи аналітики пройшли великий шлях розвитку, з початку свого існування і з розвитком ринків які мали б аналізуватись, змінювались і системи. Першою електронною аналітикою можна вважати веб-аналітику. Для відстеження входів на сайт використовувалась сесійна аналітика, а коли цього стало недостатньо, то з’явилась і подієва аналітика.

2.1 Опис сесійної аналітики

Сесійна аналітика з’явилась у результаті бажання розробників відстежити популярність свого сайту. Сесія(“вхід” або “відвідання”) – група подій, яка притаманна одному користувачу і зв’язана між собою деякою логікою.

Під час зародження такого типу аналітики, для загрузки елементів сайту, він мав перезавантажуватись. Через такий підхід до реалізації веб-сторінок, легко було закупати трафік та відстежувати кількість сесій і навіть активність кожного конкретного користувача.

Теперішні веб-сторінки мають більше інтерактивних елементів, які не призводять до перезавантаження, що робить цей метод застарілим навіть для веб-аналітики.[5]

Недоліки методу:

— Сесії неінформативні, якщо користувач заходить з різних пристроїв. У такому випадку, показник відвідування виросте, але реальна кількість користувачів не зміниться

— Складність використання у других сферах, окрім веб-розробки. Сесії в мобільному додатку можуть не відображати усієї потрібної інформації, на відміну від відвідування сайту.

— Користувач може заходити на сайт декілька разів, і зробити якусь активність лише в одну з декількох випадків. Це сильно вплине на показник конверсії і може буте неправильно проаналізовано.

2.2 Опис подієвої аналітики

Подієва аналітика стала найбільш актуальним методом збору аналітичних даних, на якому будуються майже усі сучасні аналітичні системи.

Подієва аналітика направлена не на кількість, а на якість трафіку і його дій в додатку або на сайті. Для оцінки якості, події відображають активність користувачів, їх конверсію з різних джерел і як контент впливає на показники.

Якщо програмний продукт направлений на монетизацію користувачів, то кількість сесій не буде достатньою для розуміння того, як користувач взаємодіє з додатком або веб-сторінкою. А якщо не розуміти, хто ваша цільова аудиторія, то неможливо зробити додаток краще. Розуміння аудиторії – ключ до розробки успішного додатку.[6]

У будь-якої події є тип, користувач та параметри. На основі цих даних можна будувати графіки, та сегментувати аудиторію.

Недоліки методу:

— Перш ніж вводити аналітику, треба визначити параметри по яким її треба збирати

— Кожна подія потребує окремого налаштування

2.3 Опис методу аналізу трафіку

Хоча аналіз трафіку побудований на подієвому методу збору аналітичних даних, його можна рахувати окремим видом аналітики. Аналіз трафіку не збирається

через інтеграцію з звичайними системами аналітики, а будується на основі вже створених систем відтворення реклами.

Але є і інші варіанти збору таких даних. Під час натискання на рекламний банер, користувач проходить через зовнішнє посилання, де збираються усі дані про телефон, країну, демографічні дані, якщо можливо, і відсилаються до системи аналітики. Там вони порівнюються з усіма існуючими даними, і якщо є збіг, то користувача заносять в окрему категорії трафіка.

На мобільному ринку додатків є два основні джерела трафіку:

— Реклама в інших мобільних додатках

— Реклама на веб-сторінках

Проте, аналіз трафіку поділяється не лише за його типом, але і за країною, в якій він купується та в додатках яких програється – іграх, музичних програвачах, спортивних додатках. Приклад методу аналізу трафіку приведено на рисунку 2.

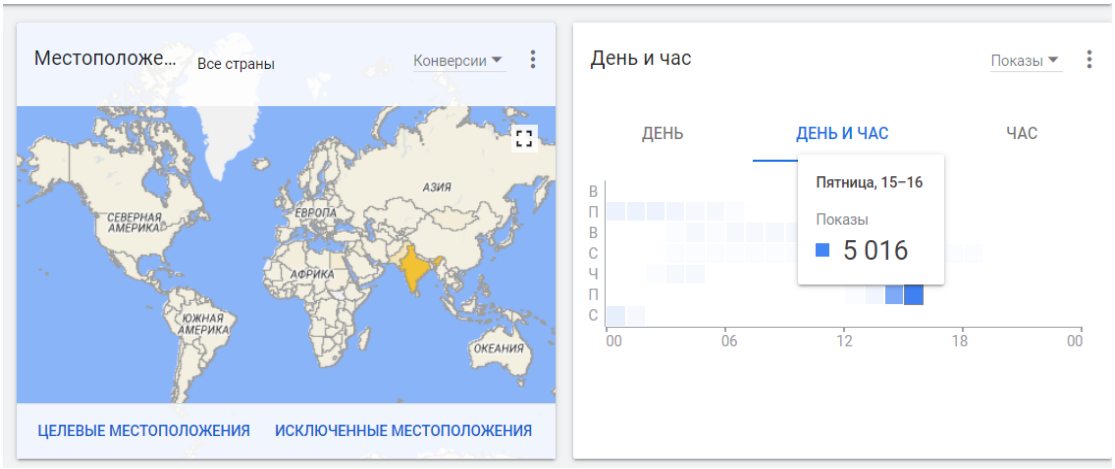

Рисунок 2 - Приклад аналізу трафіку

На кожному з зазначених типів ігрових ринків різна аудиторія, вподобання та ціна конверсії, тому і працювати з аудиторією треба по різному, для досягнення максимально ефекту.

Наприклад, трафік в Індії дешевший за трафік в Америці, але і якість користувачів набагато вища. Американці частіше купують дорогі речі, в них більше показник утримання, а через високу ціну трафіку, ціна показу реклами теж набагато більша.

Такі системи є дуже корисними, але, у більшості випадків, інформацію про якість трафіку можна отримати лише там, де цей трафік закупається. Тому, часто, системи аналізу трафіку мають можливість для інтеграції в інші аналітичні системи, для зберігання усієї інформації про додаток в одному місці.

2.4 Аналіз системи Push-повідомлень

Push-повідомлення – короткі повідомлення які система відсилає на мобільні пристрої для повернення гравця до додатку. Ця система не так часто використовується для соціальної або комп'ютерної ігрової індустрії, але є дуже важливою для підтримання показників утримання на мобільному ринку.

Системи аналітики в зв'язці з сервісом Push-повідомлень дозволяє цілеспрямовано працювати з кожним із сегментів аудиторії – повертати старих гравців, додавати сесії активним гравцям та допомагати їм, працювати з аудиторією що платить та не платить.

Хоча здається, що сервіс Push-повідомлень простий в реалізації, насправді це складна технологія. На рисунку 3 зображено принципи роботи системи Push-повідомлень.

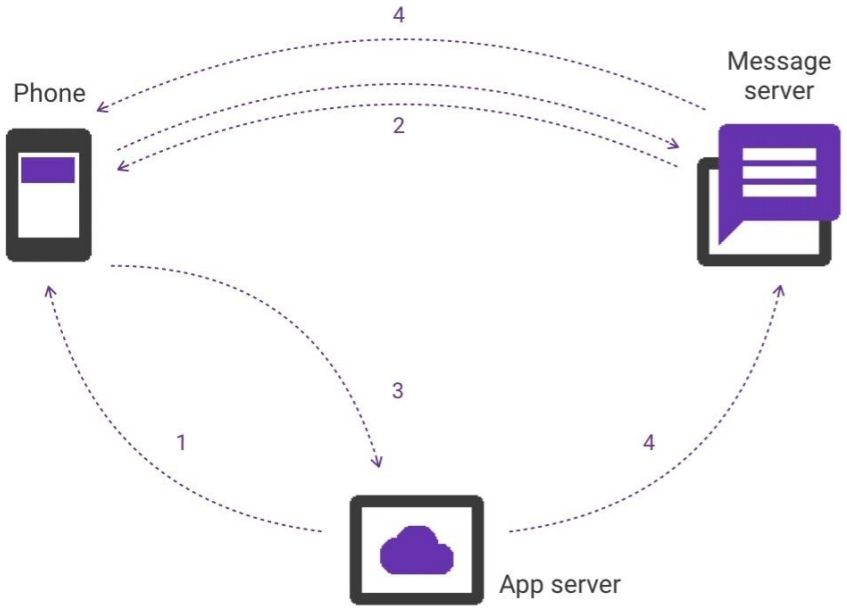

Рисунок 3 - Технологія Push повідомлень

Опис роботи системи можна розглянути по пунктах, на прикладі Push-повідомлень веб сторінок:

— Відправлення веб-сторінки на телефон користувача

— Клієнт робить підключення до серверу повідомлень і отримує ID користувача

— Відправлення отриманого раніше ідентифікатора до серверу. Сервер зв’язує пристрій з ідентифікатором користувача

— Сервер відправляє запит на показ Push-повідомлення, через сервер повідомлень, використовуючи ідентифікатори користувача та пристрою

Push-повідомлення це не просто метод повернення гравців до гри, але і метод аналітичного аналізу, який допомагає відображати утримання гравців та їх активність після отримання Push-повідомлень, дозволяє побачити як різні категорії гравців реагують на повернення та яка з механік повернення працює найкраще.

2.5 Опис методу A/B тестування

Метод A/B тестування полягає у тому, що основна частина аудиторії порівнюється з деякою тестовою групою користувачів, для яких версія додатку змінена. Цей метод використовується для оцінки ефективності змін, з метою отримання найкращий варіант.

Найчастіше, A/B тестування додатків направлене на частини користувацького інтерфейсу, такі як текстові поля та їх кольори, кнопки, їх стилі та розміщення на екрані, зображення. А для сторінки магазину, різній аудиторії показується різні іконки додатків, промо-зображення та текстовий опис додатків.

Важливо зазначити, що для проведення такого тестування, розробники повинні мати альтернативні версії додатків, що можуть включати додаткові ресурси, які повинні бути розроблені.

Насправді, метод настільки корисний, що його використовують і під час тестування, і після повноцінного релізу.

Логіка A/B тестування показана на рисунку 4.

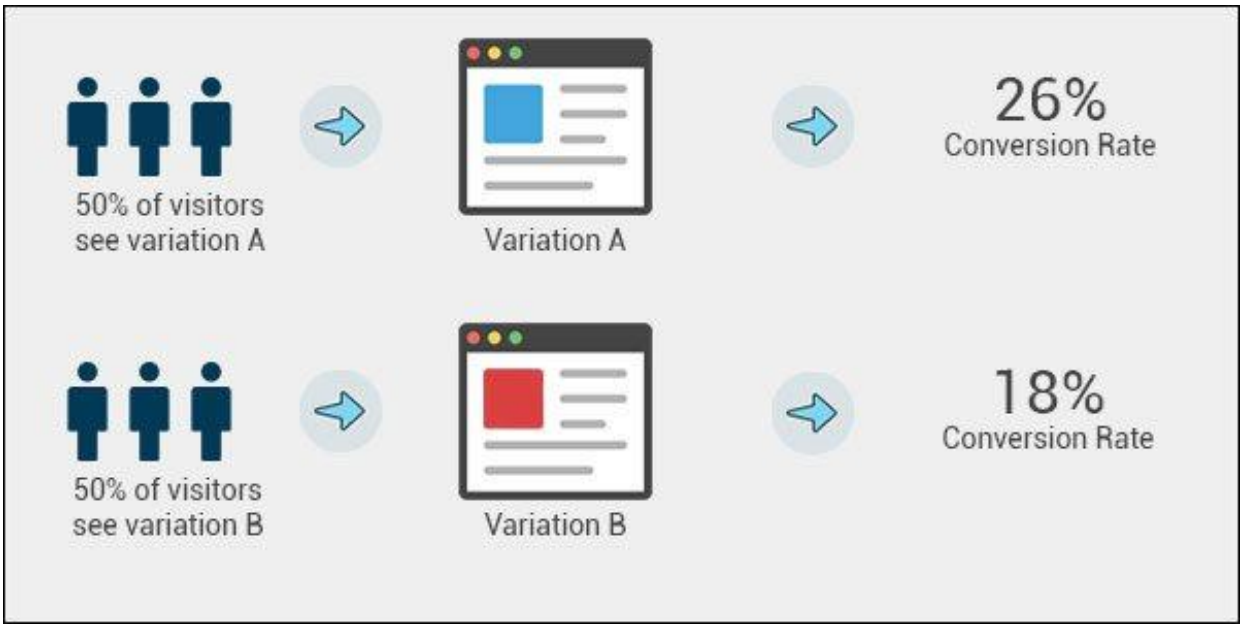

Рисунок 4 - Логіка A/B тестування

Розглядаємий метод необхідний для отримання аналітичних даних про засоби покращення основних метрик продукту.

Окрім розділення аудиторії, метод використовується під час випускання оновлення. Оновлення спочатку випускається на деякий невеликий процент аудиторії, тестується і потім процент залучення аудиторії збільшується, якщо тестування успішне.

3. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ ЗБОРУ АНАЛІТИЧНИХ ДАНИХ

Сучасна індустрія розроблення ігор не може існувати без аналітики, тому на ринку існує багато варіантів аналітичних систем: DevToDev, GoogleAnalytics, GameAnalytics, Flurry, MixPanel, deltaDNA, sensorTower. Перераховані системи системи мають різну направленість, і підходять для різних команд розробників, тому їх аналіз був першочерговою справою, під час написання розробленої системи аналітики. [7]

3.1 Характеристика системи DevToDev

DevToDev — одна з найпопулярніших аналітичних систем, яку вже використовують більш ніж півтори тисячі компаній.

На рисунку 5 показаний інтерфейс системи:

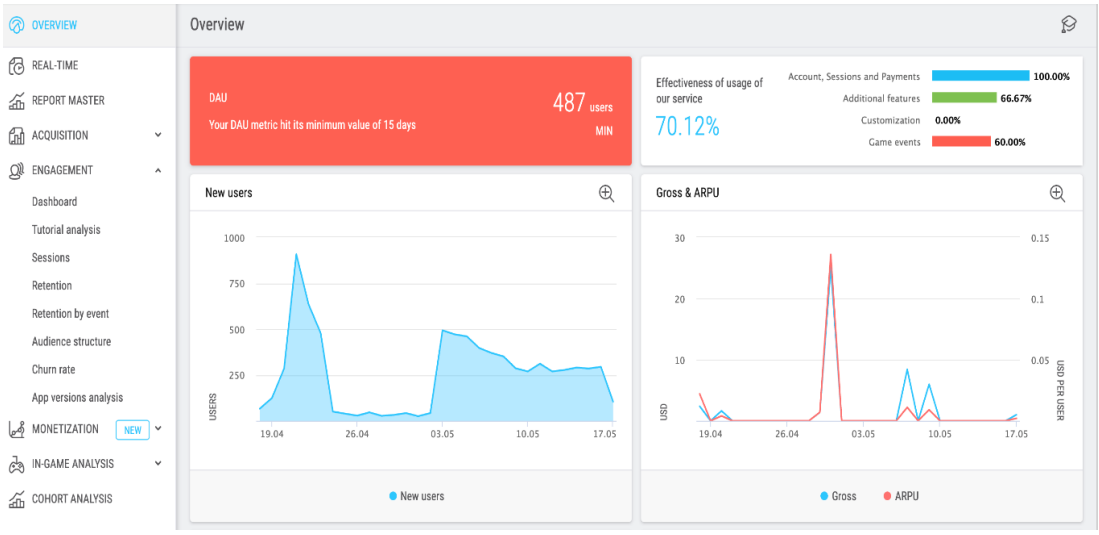

Рисунок 5 — Інтерфейс DevToDev

Система аналітики дуже об’ємна, що спричиняє проблеми в орієнтуванні по розділах і даних. Має безліч функцій, більшість з яких потрібні лише великим компаніям з командою маркетологів.

Опис функціоналу “Звітів”

Як вже було сказано, DevToDev має безліч функцій для сортування аналітики, деякі з яких варті огляду. Перша з них – звіти. Це основний функціонал системи аналітики, який дозволяє отримувати інформації по різним параметрам та за різним сортуванням. На рисунку 6 зображено готовий звіт:

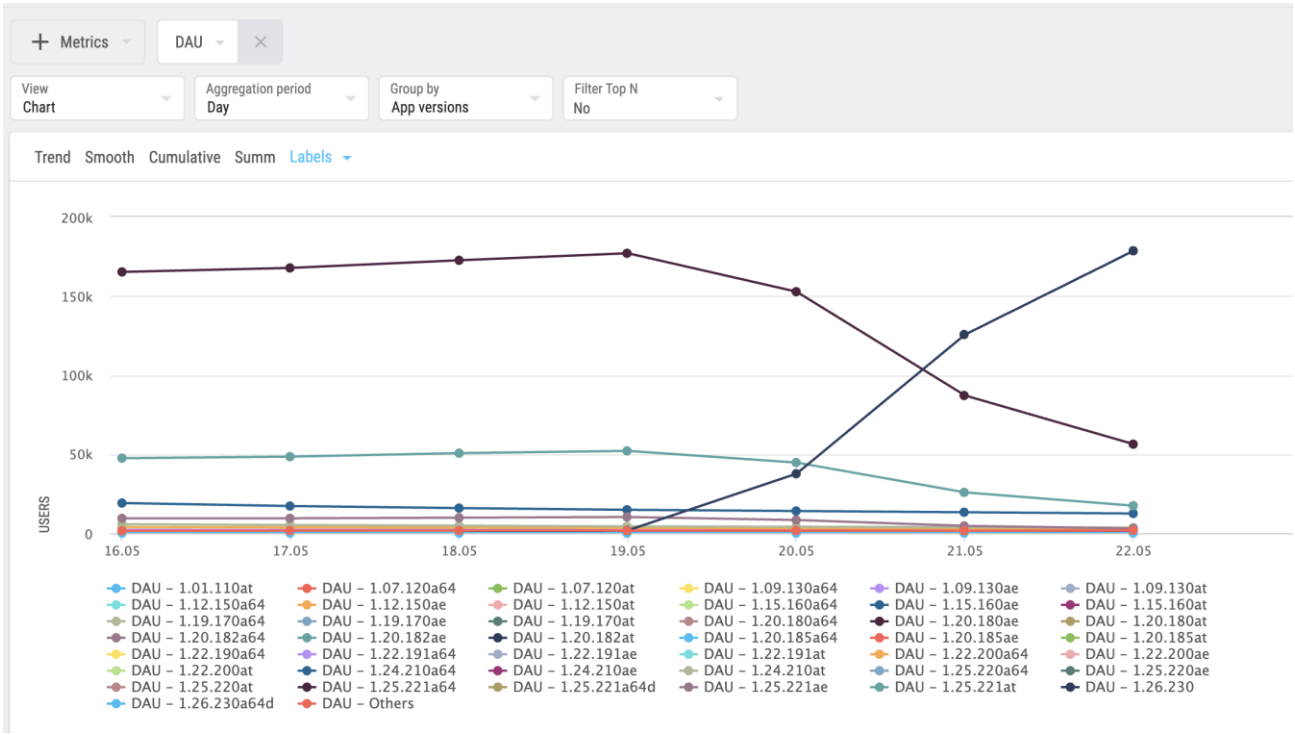

Рисунок 6 - Готовий звіт за активністю гравців

Звіти можна робити за декількома параметрами одночасно. Це дозволяє бачити залежність одних параметрів від інших, по годинах, днях, неділях, місяцях, та роках.

Дані з звітів за деякими іншими параметрами – версією додатку, країнами, середнім чеком та іншими. Це групування дозволяє сегментувати аудиторію і працювати з різними категоріями гравців по-різному.

Огляд функції “Подій” в DevToDev

Події – список декількох зв’язаних між собою дій, які гравець потрібен виконати щоб подія рахувалась виконаною.

Один з методів використання подій – пошук шляхів за якими гравці монетизуються. Наприклад, цей шлях може складатись з наступних дій – вхід, використання енергії, покупка енергії. Якщо гравець не купує енергію після її закінчення, то це може означати неправильні місця продажі.

Будування прогнозів для метрик аналітики

Прогнози автоматично будуються на основі зібраних даних. Вони допомагають побачити, в якому напрямі рухається проект і на які показники треба сподіватись. Це може бути корисно для великих компаній, менеджмент яких має бути впевнений в успішності гри не тільки в найближчій перспективі, але і в майбутньому.

Підведення підсумків аналізу системи DevToDev

Ця аналітична система одна з найпопулярніших на ринку, що забезпечується великою кількістю корисних фіч, постійною підтримкою продукту і інформаційним полем. Тому що, DevToDev це не тільки система аналітики, але і великий портал, на якому розміщуються статті, керівництва, ресурси для навчання та курси. Працівники компанії часто викладають в загальний доступ якісь матеріали для допомоги з аналітикою, що не тільки розвиває, але і дає відчуття залученості у великий колектив розробників.

Підводячи підсумки, ця системи має деякі значні переваги:

— Має декілька тарифів, які підійдуть як початківцям, так і великим компаніям.

— Має як вбудовані метрики, так і дозволяє створювати користувальницькі події.

— Інтеграція до проєкту розробленого майже в будь-якому середовищі

— Велика кількість документації та додаткової інформації. Уся інформація з сайту зібрана для допомоги користувачу в використанні системи

Але система має і недоліки:

— Велика кількість фіч потребує детального огляду та інструкції по використанню.

— Деякі з аналітичних параметрів повторюються у різних розділах, що спричиняє перевантаження інтерфейсу

— Для отримання розгорнутої аналітики кожний параметр системи має налаштовуватись окремо, спеціалістами з досвідом

3.2 Розглядання системи GameAnalytics

GameAnalytics — безкоштовний сервіс, що аналізує поведінку користувачів мобільних ігор. Призначений для відстежування додатків, що працюють на платформах і рушіях iOS, Android, Unity, JavaScript, Unreal4, GameMaker, Xamarin, Air, Corona.

GameAnalytics збирає базову інформацію про завантажування, тривалість сесій, активність гравців, кількість нових користувачів з можливістю поділу за країнами, а також про дохід і ступінь залученості відвідувачів. Інтерфейс системи можна побачити на рисунку 7.

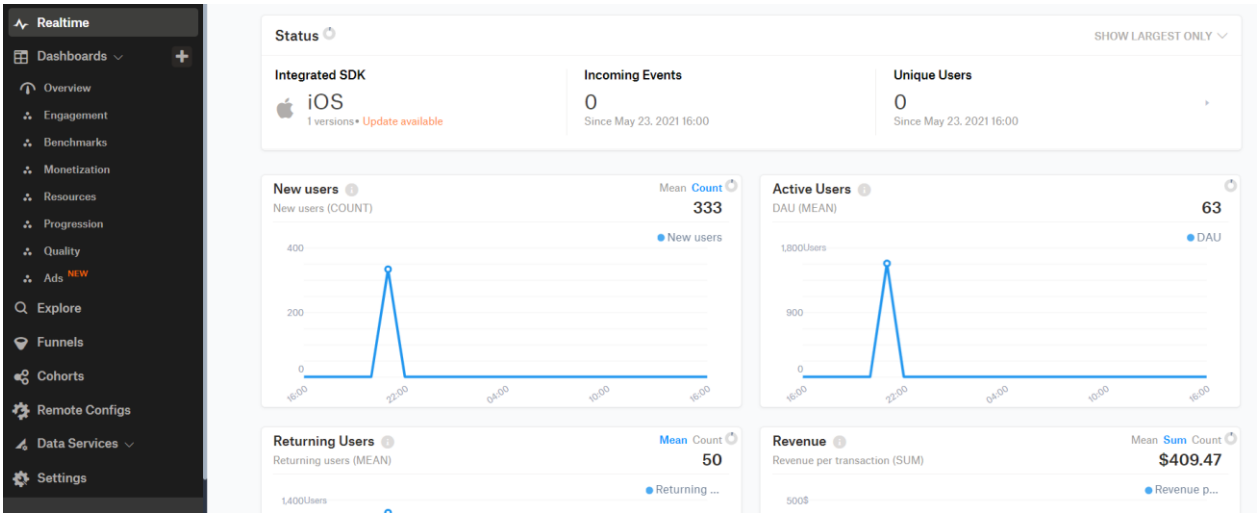

Рисунок 7 - Інтерфейс GameAnalytics

Сервіс аналізує, як зміни або акції впливають на поведінку аудиторії, при цьому можлива сегментація гравців за такими параметрами, як пройдені рівні або витрачена сума грошей.

Основна особливість GameAnalytics – налаштування користувацьких панелей. Панелі – особливий тип звітів, в який можна додавати деяку кількість віджетів, для відображення інформації. Екран налаштування віджетів зображений на рисунку 8.

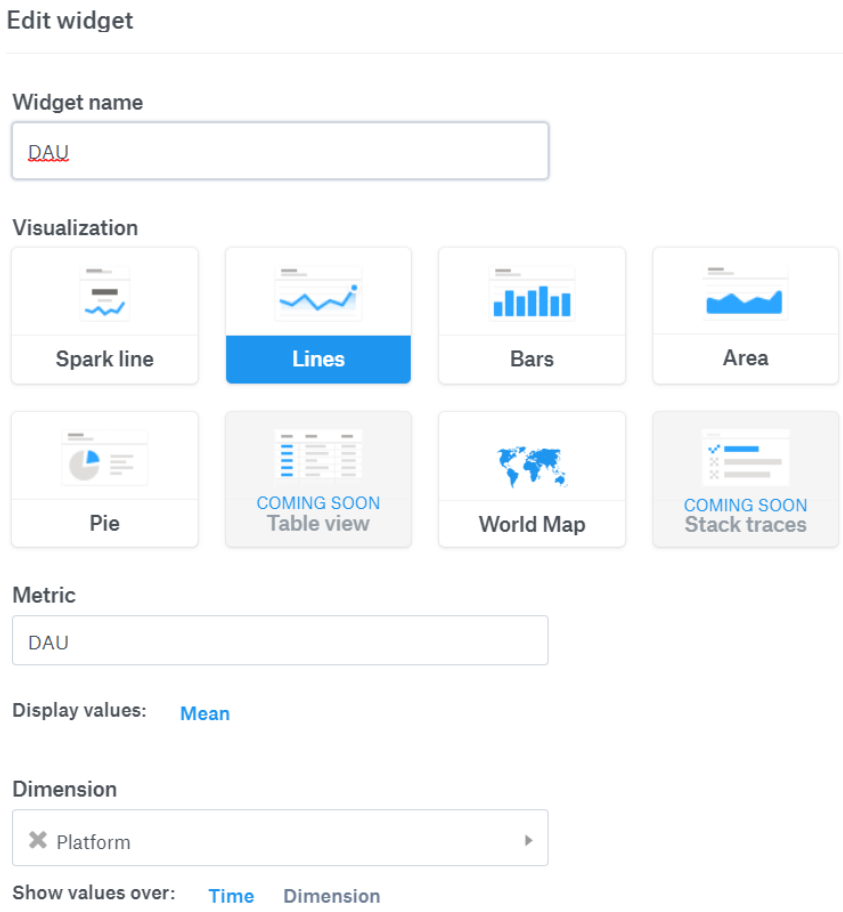

Рисунок 8 – Налаштування віджетів

Віджети можуть бути у різних графічних представленнях:

— Лінійних діаграмах

— Гістограмах

— Діаграма з областями

— Кругові діаграми

— Табличний вигляд

— Карта світу, з розділенням аудиторії по країнам

Після вибору методу візуалізації, залишається вибір метрики, яка буде відображатись в віджеті. Такий підхід до відображення аналітики зручний, у порівнянні з відокремленими звітами.

Особливості GameAnalytics в порівнянні з іншими системами:

— Безкоштовний

— Інформація про помилки

— SDK розроблено спеціально для ігор

З недоліків можна виділити відсутність інформації про сервіс і його використання.

3.3 Опис системи аналітики SensorTower

Не зовсім звичайна система аналітики. SensorTower показує аналітику конкурентно — у порівнянні з іншими схожими додатками. Її використання доступно лише за платною підпискою, але безкоштовно можна переглядати сторінки додатків, зміну позицій у магазинах та історію відгуків. На рисунку 9 зображено інтерфейс системи.

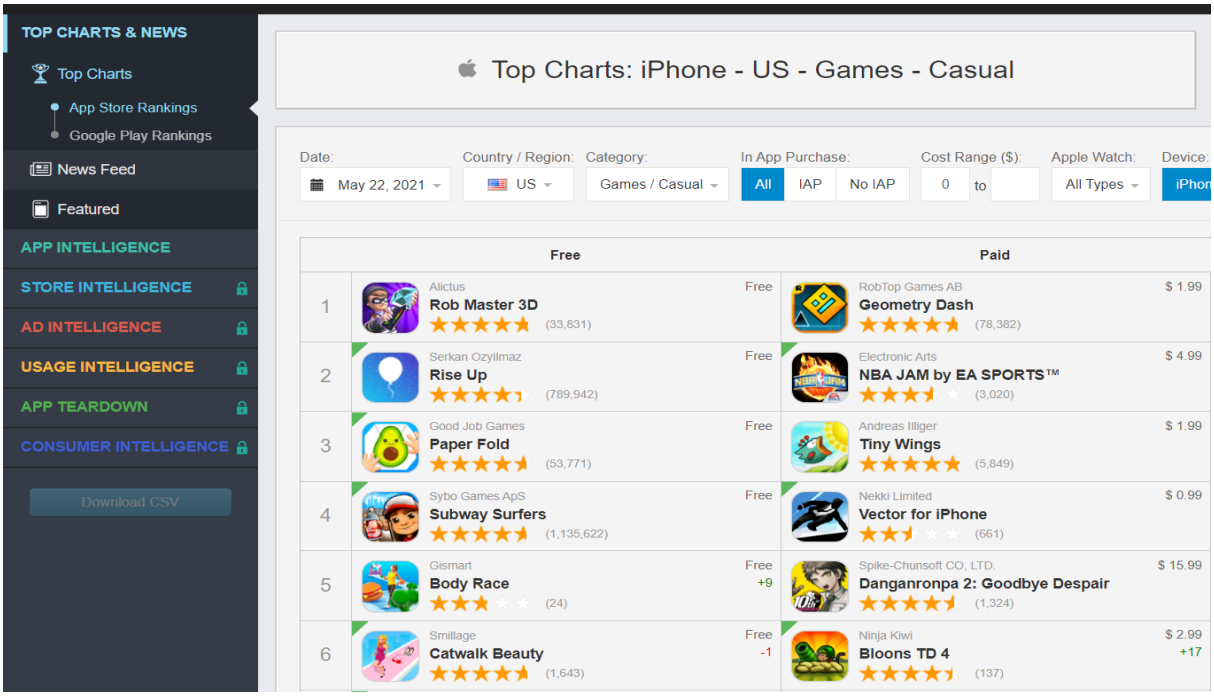

Рисунок 9 - Інтерфейс SensorTower

Окрім цього, SensorTower робить моніторинг рекламних компаній — платформа збирає дані про обсяги аудиторії, залучених із різних джерел трафіку, у тому числі з рекламних майданчиків.

Зазвичай трафік до додатка закупають, але розробники розраховують і на пасивний трафік. Для його забезпечення, треба аналізувати схожі додатки, їхні описи на сторінках магазинів, знаходити ключові слова та будувати свою сторінку на основі отриманих даних. SensorTower дає змогу отримувати всі ці дані по підписці на сервіс.

Ця система аналітики може автоматично проводити аналіз ринку та аудиторії. У дослідженні аудиторії збираються дані про активність користувачів у різних країнах, коефіцієнт повернення, демографічні дані відвідувачів і тривалість сесій. Доступні відомості про загальні тенденції на ринках. Ця інформація дає змогу створювати продукт не тільки на припущеннях про пристрій ринку, але і ґрунтуючись на перевірених даних.

SensorTower пропонує можливість отримати дані про використання різних SDK в популярних додатках. Ця інформація може бути корисна для компаній, які в пошуках кращих інструментів для розроблення.

Особливості SensorTower:

- Збір інформації про тенденції ринку
- Конкурентне порівняння
- Глибокий аналіз ринку, а не конкретного додатку

3.4 Аналіз вбудованої Google аналітики

Окремо можна виділити системи аналітики магазинів додатків. Ці системи загалом дуже схожі, однак розглядати краще саме систему Google через більш відкриту структуру, на відміну від систем AppStore.

Після релізу свого додатку до Play Маркету, Google починає збирати аналітику з проєкту:

— джерела трафіку та конверсії

— помилки в програмі

— успішність сторінки магазину

— унікальні активні користувачі

Окрім описаних аналітичних даних, система може збирати дані про успішність різних етапів тестування гри. Ці метрики особливо важливі через те, що під час повноцінного релізу гри, Google просуває додаток в магазині зважаючи на його успішність під час тестування.

На рисунку 10 зображений деякий функціонал аналітики:

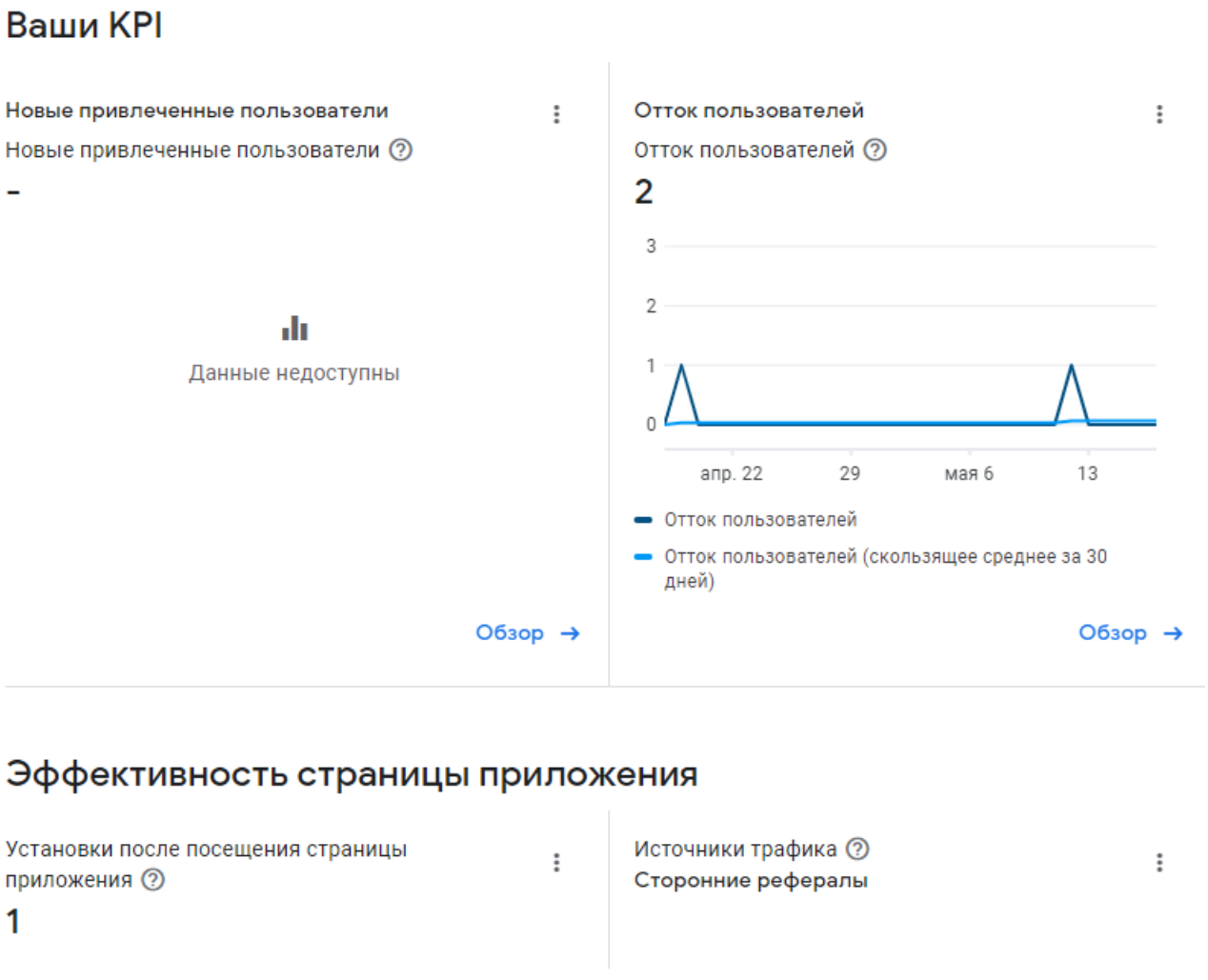

Рисунок 10 — Частина функціоналу аналітики Google

Усі описані дані збираються без будь-якого втручання розробника, інтеграції інших систем або бібліотек, що робить цей тип аналітичної системи дуже зручним

навіть для новачків. Єдина умова коректної роботи аналітики – достатня кількість користувачів.

Розглядання способів використання Android Vitals

Можна окремо виділити пункт аналітики з назвою Android Vitals. У цьому пункті можна подивитися дані про швидкість запуску програми, витрати заряду батареї, стабільності додатку, швидкості відтворення графіки й частоті кадрів. А разом з розділом Android Vitals йдуть інструкції щодо поліпшення програми-скорочення розміру програми, поліпшення продуктивності та оптимізації сторінки.

Ще одним важливим показником є розмір додатку. Хоча на даний момент часу пам’ять телефону досягає сотні гігабайт, розмір файлу додатку дуже важливий на мобільному ринку – чим менше розмір додатку, тим більший шанс що його скачають. При використанні аналітики Google, Android Vitals може як можна зменшити розмір додатку, а також вказати середній розмір схожих додатків, опираючись на аналіз магазину.

Використання A/B тестування в Google аналітиці

Як вже було зазначено, A/B тестування – однин з найважливіших та найвживаніших методів маркетингового аналізу даних, який дозволяє сегментувати аудиторію по версіях додатку, за допомогою регулювання доступності версій гри, збирати дані про різні види додатків, та знаходити правильні шляхи для розвитку програмного продукту.

Google дозволяє проводити A/B тестування версій гри, сторінки додатку в магазині а також рекламних компаній, але не відстежувати внутрішньоігрові аналітичні метрики різних версій.

Для розширеного A/B тестування в межах цієї ж системи, треба використовувати окремий сервіс Google Optimize.

Висновки з аналізу Google аналітики

Окрім переліченого функціоналу, Google має безліч окремих систем, які надають можливість купувати трафік і бачити його ефективність, налаштовувати системи досягнень і рейтинги та оцінювати ступінь залучення гравців до них. Ця “екосистема” навіть дозволяє інтегрувати та відстежувати показ реклами у додатку.

Для всіх пунктів в аналітиці та для окремих систем є докладні інструкції з використання та інтеграції, різними мовами. Укупі з усіма попередніми можливостями системи, вона стає незамінимою для розробників.

Плюси:

— Відсутність будь-якої додаткової роботи з боку розробника

— Різноманітні параметри дозволяють оцінювати потенціал програми

— Зручне зображення даних

Мінуси:

— Обмежений функціонал

— Неможливість використання, якщо для розповсюдження додатку використовується будь-який інший магазин додатків

4. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

Існує багато систем аналітики, але більшість з них представлені у вигляді сайтів, а не мобільних додатків. Мобільні додатки мають значно більше переваг – інтерфейс ближчий до звичного інтерфейсу комп'ютерів, швидкість роботи, інтеграція з платформою розробки, інтеграція з магазином та доступ до їх аудиторії.

В ході написання системи аналітики було створено C# бібліотеку та консольний проект C#. Для написання елементів на C# була обрана платформа .Net Framework 4.7.2. Також був створений проєкт на рушії Unity з використання C# скриптів, і базу даних MySql з використанням MySql Workbench.

Як середовище програмування було вибрано Microsoft Visual Studio 2019 Community. Мова програмування – об'єктно-орієнтована мова програмування C#. Вибір мови та середовища розробки був зроблений враховуючи технології та мови програмування, що використовуються на базі практики, та з урахуванням майбутніх потреб користувачів.[8]

4.1 Перелік особливостей .Net Framework

.Net – це безкоштовна платформа від компанії Microsoft, яка підтримує створення та виконання додатків Windows. Платформа .Net забезпечує об'єктно-орієнтовне середовище розробки для декількох мов програмування.

.Net Framework складається з двох частин – середовища виконання(CLR) та бібліотеки класів .Net Framework. [9]

Опис середовища виконання CLR

Common Language Runtime (CLR) – середовище виконання, яке керує виконанням коду і надає програмі доступ до різних служб, таких як керування

пам’яттю та керування потоками. CLR забезпечує виконання коду на різних мовах, за допомогою Intermediate Language. Intermediate Language – низькорівневий код, схожий на Assembler, в який перетворюється високорівневий код.

Середа виконання також забезпечує:

— безпеку виконання коду, через сурову типізацію та перевірки коду

— Автоматичне управління пам’яттю, що виключає її виток та недійсні посилання

— Зручність написання коду будь-якою зручною мовою програмування, через надання можливості використання усіх переваг середовища, бібліотек та компонентів, написаних іншими розробниками

Опис бібліотека класів .Net Framework

Набір типів, які інтегруються в середовище розробки CLR. Бібліотека надає типи від яких користувач може успадковувати функції, що значно полегшує роботу з типами, надає можливість об’єднувати користувацькі компоненти та зменшує час на вивчення засобів платформи.

4.2 Характеристика рушія Unity

Unity — це безкоштовне середовище для розроблення ігор і додатків. Основними перевагами середовища є наявність візуального середовища розроблення, модульної системи компонентів і кросплатформенність.

Розглядання допоміжних систем Unity

Unity містить безліч корисних функцій, включаючи вбудовану систему контролю версій і паралельного розроблення під назвою Collaborate, магазин ассетів і підтримку користувальницьких плагінів.

Collaborate — безкоштовне, для невеликих команд, рішення, вбудоване в Unity, яке дозволяє зберігати, обмінювати і синхронізувати дані між співробітниками компанії, де б вони не знаходилися. Collaborate простий в підключенні і використанні, так як розроблявся спеціально для Unity. На рисунку 11 зображено використання Collaborate-а в одному з проектів.

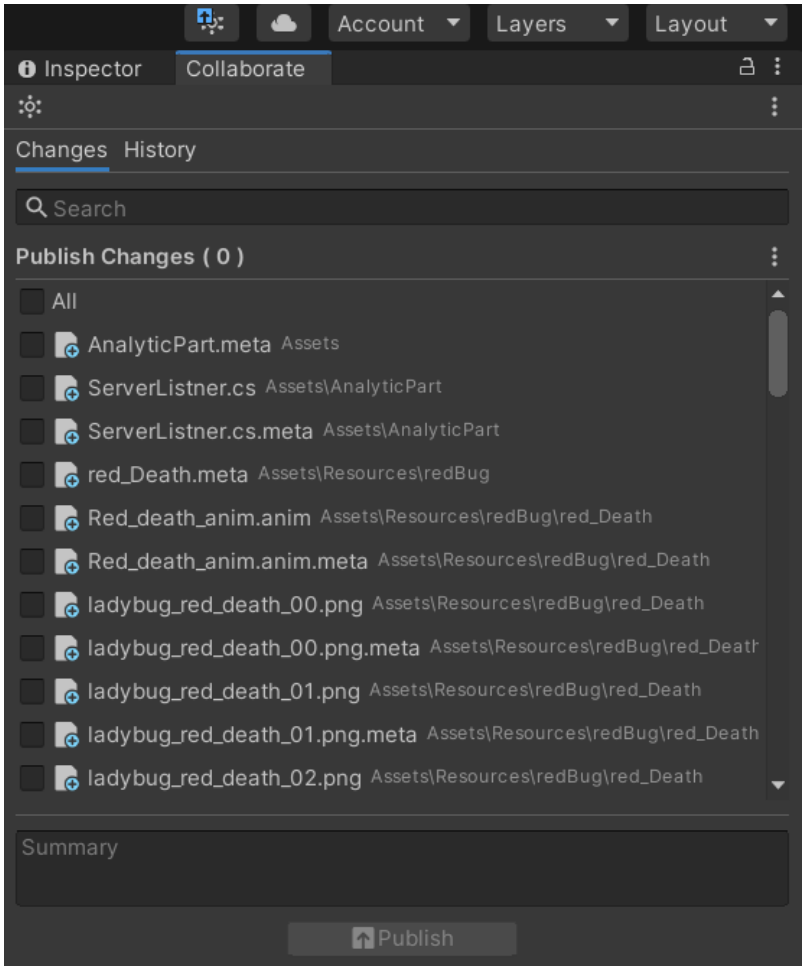

Рисунок 12 - Інтерфейс Collaborate-а в проекті

Для кожного члена команди, Collaborate створює окрему гілку, в якій розробники можуть робити будь-які зміні, відміняти їх або повертатись до попередніх версій гри.

Усі зміни з поточної головної гілки мають бути збережені, для впровадження нових змін. Якщо під час впровадження змін виникає конфлікт, то програма пропонує його виправити, вказуючи файл та його проблемні елементи. Єдиним

недоліком системи можна вважати відсутність можливості контролю версії іншими учасниками розробки, а також відсутність прозорості.

Магазин асетів — ще одна перевага Unity перед іншими середовищами розроблення. Магазин дозволяє шукати і купувати вже створені іншими користувачами об'єкти, скрипти, елементи UI, спрайти або навіть окремі функціональні системи. Після купівлі всі файли доступні для завантаження в самому рушії, і будуть готові для роботи, без будь-яких додаткових налаштувань. Важливо зазначити, що деякі асети можна редагувати, а деякі – ні. Умови використання залежать від продавця. Екран завантаження асетів зображений на рисунку 13.

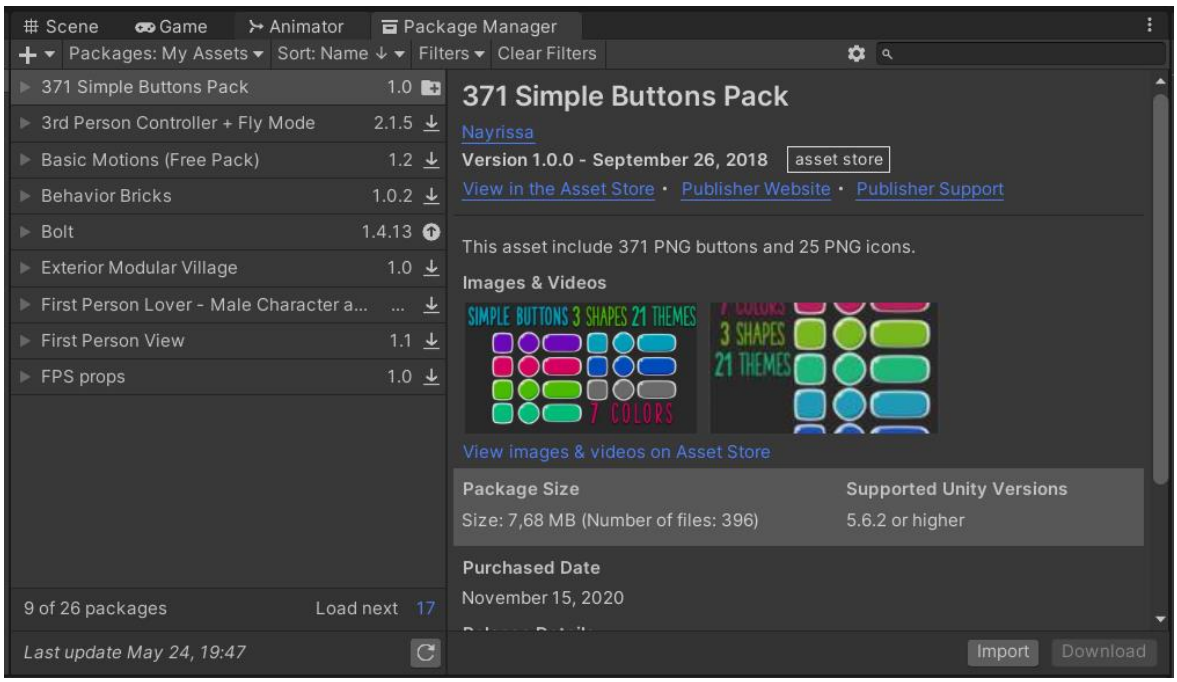

Рисунок 13 - Екран завантаження асетів в рушії Unity

На екрані завантаження асетів можна побачити декілька асетів які використовувались під час розробки ігор. Файли можна завантажувати усі, або вибирати окремі, з усіх файлів ассету.

У Unity зазвичай використовуються скрипти для створення функціоналу проєкту, але також можна підключити код, створений поза Unity, у вигляді плагіна. У Unity можна використовувати два типи плагінів: керовані плагіни і нативні плагіни.

Керовані плагіни — це керовані збірки .NET, створені за допомогою таких інструментів, як Visual Studio або MonoDevelop. Вони містять тільки код .NET, що

означає, що вони не можуть отримати доступ до функцій, які не підтримуються бібліотеками .NET. Однак керований код доступний для стандартних інструментів .NET, які Unity використовує для компіляції скриптів. Таким чином, існує невелика різниця у використанні керованого коду модуля і коду сценарію Unity, за винятком того факту, що модулі компілюються поза Unity, і тому вихідний код може бути недоступний.

Нативні плагіни — це бібліотеки нативного коду для конкретної платформи. Вони можуть отримати доступ до таких функцій, як виклики ОС і сторонні бібліотеки коду, які в іншому випадку були б недоступні для Unity. Однак ці бібліотеки недоступні для інструментів Unity, на відміну від керованих бібліотек. Наприклад, якщо ви забудете додати в проєкт керований файл модуля, ви отримаєте стандартні повідомлення про помилки компілятора. Якщо ви зробите те ж саме з власним плагіном, ви побачите звіт про помилку тільки при спробі запустити проєкт.[10]

Аналіз інтерфейсу рушія Unity

Повне розуміння проєктів потребує розуміння інтерфейсу рушія, який буде розглянутий, до акцентування уваги структурі побудови додатків. Опис сцени, об’єктів на ній та сховище асетів відображені на рисунку 14.

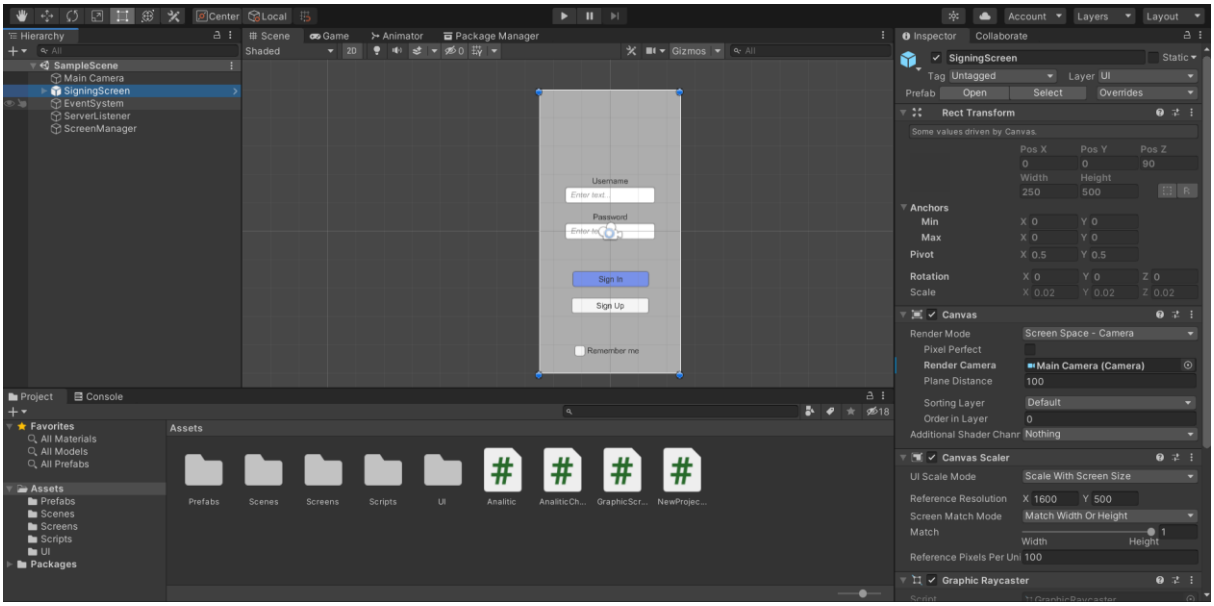

Рисунок 14 - Інтерфейс рушія Unity

Інтерфейс рушія поділяється на окремі секції:

— Верхня частина інтерфейсу відведена під налаштування проекту, його виду в редакторі та різним функціям тестування.

— В лівій частині відображаються об’єкти на сцені. В цій же частині можна створювати нові об’єкти, та перетаскувати їх до сховища асетів. Для швидкого стоврення додатків, Unity пропонує використовувати шаблони об’єктів.

— Нижня частина відведена під сховище асетів та консольний вивід. У сховищі асетів зберігаються об’єкти, які можна скопіювати на сцену у будь-який момент. Об’єкти на сцені можуть не мати аналогів в сховищі. Об'єкти, що зберігаються на сцені, будуть завантажені при запуску сцени, а ті що в сховищі - називаються “префабами”, і їхні екземпляри повинні бути створені з інших, вже створених на сцені, об'єктів.

— Права частина відображає модулі вибраного об’єкту та дозволяє створювати нові, видаляти вже створені та редагувати їх. Unity пропонує безліч вбудованих модулів, які ідеально підходять як для розробки додатків, так і ігор різних жанрів.

Розглядання структури проєктів Unity

Проєкт в Unity ділиться на сцени, кожна з яких може містити свої власні об'єкти і налаштування. На сцені зазвичай існує декілька вбудованих компонентів, таких як камера, та менеджер подій. Усі інші об’єкти розробник має створювати сам, на етапі розробки сцена, або об’єкти будуть створюватись з скриптів.

Спочатку об'єкти являють собою лише компонент з позицією на сцені, проте модульна структура дозволяє додавати будь-які компоненти — спрайти, 3D моделі, компоненти обробки фізики, UI компоненти або скрипти.

Скрипти в Unity — це спосіб задання логіки виконання програми. Для взаємодії між скриптами, у одному з них потрібно оголосити об'єкт такого ж типу, як і назва

скрипта, посилання на який потрібно отримати. Створеному об'єкту потрібно присвоїти конкретний об’єкт скрипта з іншого об'єкта, і це можна зробити двома способами:

— Вказувати напряму посилання на скрипт конкретного об'єкта в редакторі, але іноді це зробити неможливо через відсутність конкретного об'єкта на сцені. Тоді використовується другий варіант.

— За допомогою вбудованого пошуку об'єктів, Unity може знайти будь-який об'єкт на сцені, а після — отримати скрипт-компонент знайденого об'єкта.

У Unity також використовується система подій під назвою UnityEvents. На рисунку 15 відображається налаштування події:

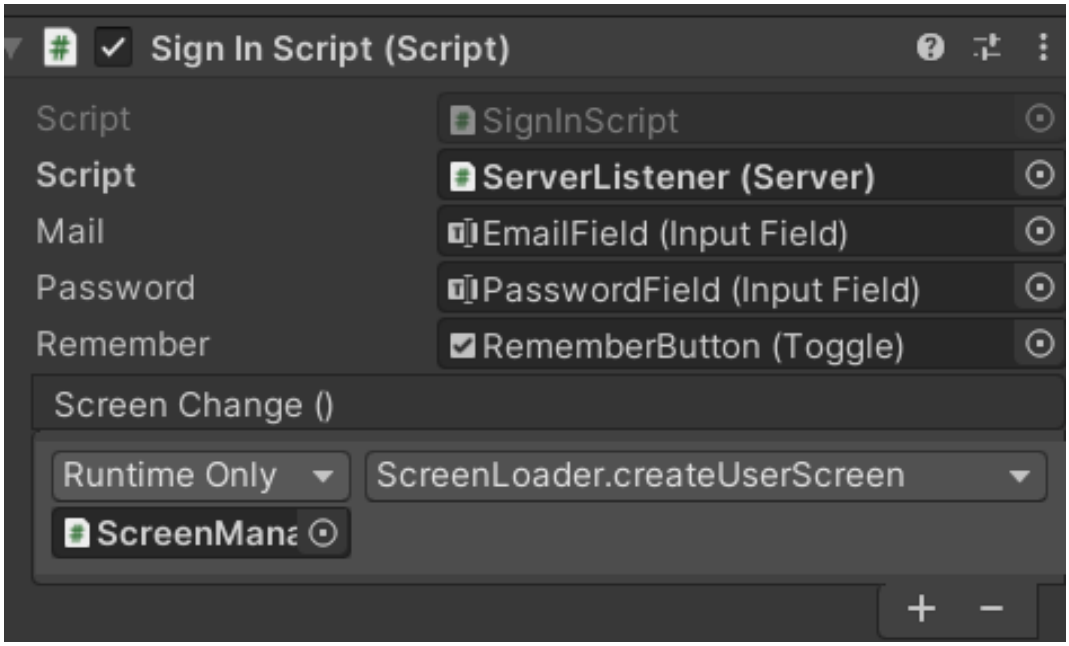

Рисунок 15 — Налаштована подія, прив’язана дозміни екрану.

В наведеному прикладі, ScreenChange – подія, яка оголошується публічною змінною та викливається в SignInScript. Після цього, скрипт налаштовується в інспекторі – додається посилання на потрібний скрипт, в якому і буде викликатись публічний метод.

UnityEvents дозволяє використовувати керовану користувачем функцію виклику від моменту редагування до моменту запуску, без необхідності додаткової зміни коду і конфігурації скриптів. UnityEvents викликаються в коді як звичайні .Net

делегати, але можуть бути додатково налаштованими в інспекторі об'єкта. Так як методи зазвичай потребують деякі параметри для роботи, то ця можливість була врахована при розробці подій Unity. За допомогою UnityEvents можна передавати до чотирьох параметрів одночасно.

І хоча швидкість виконання цих операцій за звичай може бути до 20 разів повільніше, ніж звичайні C# делегати, швидкість вионання все ще дуже вилика, і UnityEvents простіше використовувати.[11]

4.3 Обгрунтування вибору СУБД MySql

Під час розробки програмного продукту, з’явилось питання щодо вибору типу бази даних, який би був задовільний для використання в системі. Розглядались як реляційні, так і не реляційні бази даних.

Порівняння реляційних та не реляційних баз даних

Реляційні бази даних зберігають дані структуровано, а записи в них можуть відображати сутності реального світу. Так, наприклад, запис може описувати користувачів аналітики – назву компанії, пароль для входу, кількість розробників в команді.

Структура реляційних баз даних дозволяє поєднувати дані з декількох різних таблиць за допомогою зовнішніх ключей, які використовуються для знаходження будь-якого окремого елементу бази даних.

Для використання реляційних баз даних використовується мова запитів Sql. Використання мови Sql потребує визначення точної структури бази даних, і будь-яка зміна в структурі може потребувати виправлення програмного продукту.

На відміну від реляційних баз даних, нереляційні пропонують динамічну структуру, яка може приймати різні вигляди:

— У вигляді представлення “Ключ, значення”. Кожному значенню в таблиці відповідає унікальний ключ, за яким і здійснюється пошук. З одної сторони, цей тип нереляційної бази даних дуже простий, і дозволяє масштабувати систему до поки вистачає обчислювальних потужностей, а дані в таких таблицях можуть мати будь-які типи. З іншої сторони, така проста структура не дозволяє проводити звичні, для реляційних баз даних, операції.

— Документо-орієнтовний вигляд представляється як ієрархічна структура. Структура має коріневий вузол, який може містити внутрішні вузли, або листові вузли. В листових вузлах міститься кінцева інформація, яка зберігається в індексах бази даних, і за якими можна проводити швидкий пошук. Загалом, цей тип бази даних є більш складною версією “Ключ, значення” – структура не дозволяє легко працювати з БД, в яких багато елементів даних зв’язані між собою.

— У вигляді графів. Дана модель використовує ребра та вершини для представлення даних. Найбільш продуктивним використанням такої структури можна домогтись у базах даних, представлених у вигляді графів. Наприклад, мережеві моделі.

— По колонках. Дані в базі даних структуруються у вигляді матриці, в якій стовпчики та рядки є унікальними ключами, за якими виконується пошук.

Як можна побачити, лише один тип не реляційних баз даних міг бути використаний для розроблення системи аналітики, а саме – документо-орієнтовний. Такі СУБД як MongoDB, eXist, CouchBase дозволяють працювати з цією моделлю даних, але вибір нереляційної бази даних накладає деякі складності та особливості розроблення системи. Перша з них – відсутність єдиної термінології. Таблиця порівняння термінології показана на рисунку 16.

MongoDB	DynamoDB	Cassandra	Couchbase
Коллекция	Таблица	Таблица	Корзина данных
Документ	Элемент	Ряд	Документ
Поле	Атрибут	Столбец	Поле
ObjectId	Первичный ключ	Первичный ключ	ИД документа
Индекс	Вторичный индекс	Индекс	Индекс
Представление	Глобальный вторичный индекс	Материализованное представление	Представление
Встроенный документ	Карта	Карта	Карта
Массив	Список	Список	Список

Рисунок 16 - Таблиця порівняння термінології NoSql СУБД

Через відсутність універсальної мови запитів Sql, різні системи використовують свої внутрішні API, тому можлива зміна СУБД потребувала б змін в коді програм. Більш того, кожна API побудована на основі мови Sql, що випливає в їх менший функціонал. Окрім цього, нереляційні бази даних мають складну структуру, та невисоку швидкість обробки даних, що зв’язано з пошуком даних по ключу.

Опис СУБД MySql

Зважаючи на усе вищеописане, для збереження даних про користувачів була обрана реляційна СУБД MySql. Вона відмінно підходить для обробки великих баз даних і легко масштабується. У MySql реалізована багатопотоковість, що гарантує високу швидкість роботи бази даних. При цьому, СУБД є безпечною.

MySql дозволяє виконувати усі потрібні для розробленого програмного продукту SQL запити, підтримує сортування та групування даних, використання функцій та об’єднання таблиць у запитах.

Для MySql існує графічне середовище Workbench, яке надає розробникові можливість проєктувати, інтегрувати, моделювати, створювати і експлуатувати БД в єдиному оточенні. На рисунку 17 показано інтерфейс MySql Workbench.[12]

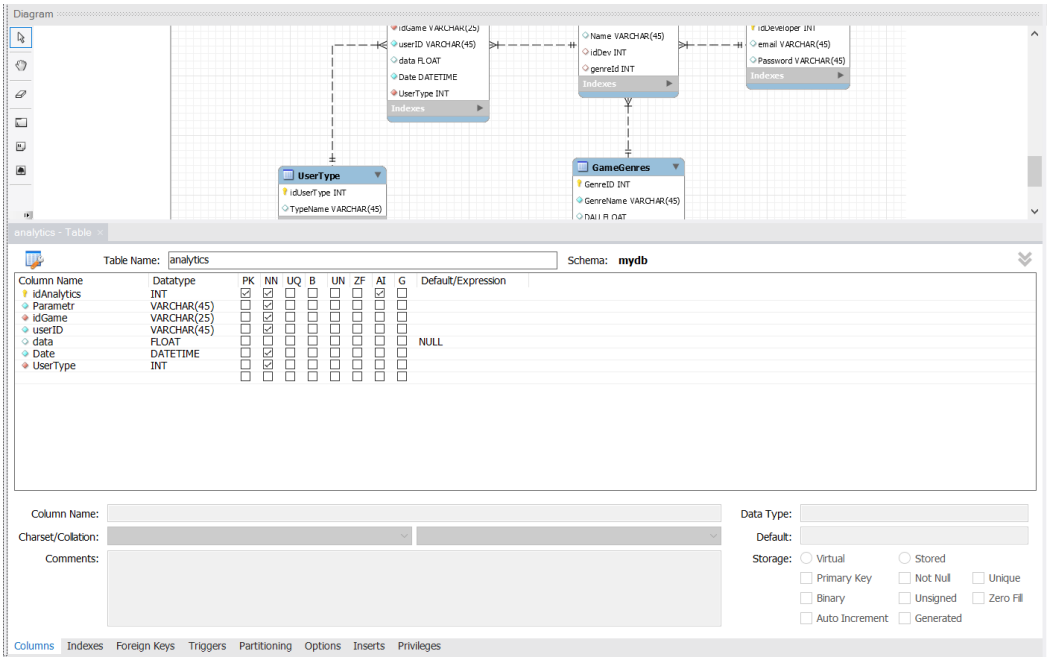

Рисунок 17 - Інтерфейс MySql Workbench

Можливості Workbench:

— Представлення БД в графічному вигляді

— Створення структури таблиць з уже існуючої БД

— Встановлення зв'язків між таблицями у візуальному поданні

— Редагування таблиць в графічному поданні, а не за допомогою SQL запитів

— Редактор SQL пропонує користувачу підсвічування синтаксису, автозаповнення частин запитів

— Дає користувачу інструменти для підвищення швидкості роботи MySql додатків

Крім перерахованих вище переваг, MySql легко зв'язується з Visual Studio і C#. Для зв’язування бази даних та серверної частини, було використано бібліотеку MySqlClient.

5. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Створена система аналітики поділяється на 4 окремі частини, які можна розглянути окремо одна від одної: бібліотека взаємодії з сервером, сервер, БД і мобільний додаток. Ці частини притаманні майже кожній з аналітичних систем.

Архітектура системи показана на рисунку 18.

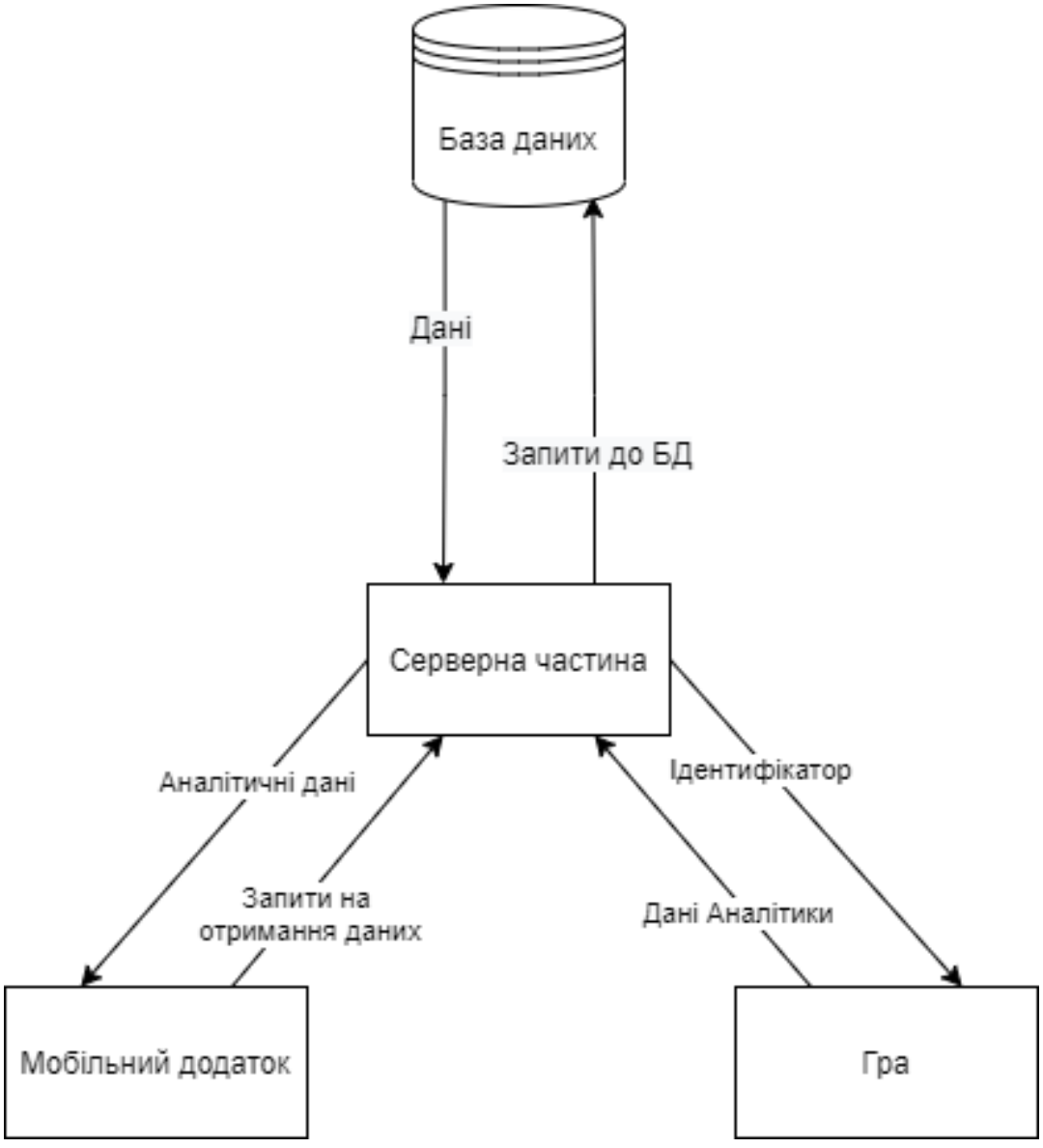

Рисунок 18 — Архітектура системи аналітики

Така структура системи аналітики дозволяє легко вносити зміни в будь-яку частину системи, не чіпаючи інші.

5.1 Структура бібліотеки взаємодії з сервером

Бібліотека взаємодії відповідає за підключення гри до серверу, та передачу даних аналітики. Вона інтегрується додаванням файлу до файлів проєкту. Усі можливі параметри, які потрібні бути задані для під'єднання до сервера визначаються в тілі класу, і не можуть бути змінені розробником, для уникнення помилок.

Бібліотека містить основні методи обміну даними з сервером — під'єднання до серверу, відправку даних та їх прийому, а також методи задання та отримання ідентифікаторів. Методи бібліотеки використовують протокол TCP для з'єднання і обміну інформації з сервером.

В методі під'єднання до сервера не тільки створюється з'єднання з сервером, а і відправляються дані для ініціалізації підключення, такі як ідентифікатор гравця та ідентифікатор гри. Якщо ідентифікатор гри ще не був заданий, тобто гравець перший раз заходить в гру, то під час підключення цей ідентифікатор очікується з сервера.

Якщо підключення не вдалось виконати, то метод поверне розробнику значення False. Це можна використовувати, якщо логіка роботи гри зміниться, в залежності від підключення системи аналітики.

Для збору аналітики розробник повинен викликати в коді гри метод відправлення даних на сервер, з параметром аналітики і даними для збору. Приклад встановлення аналітики зображений на рисунку 19.

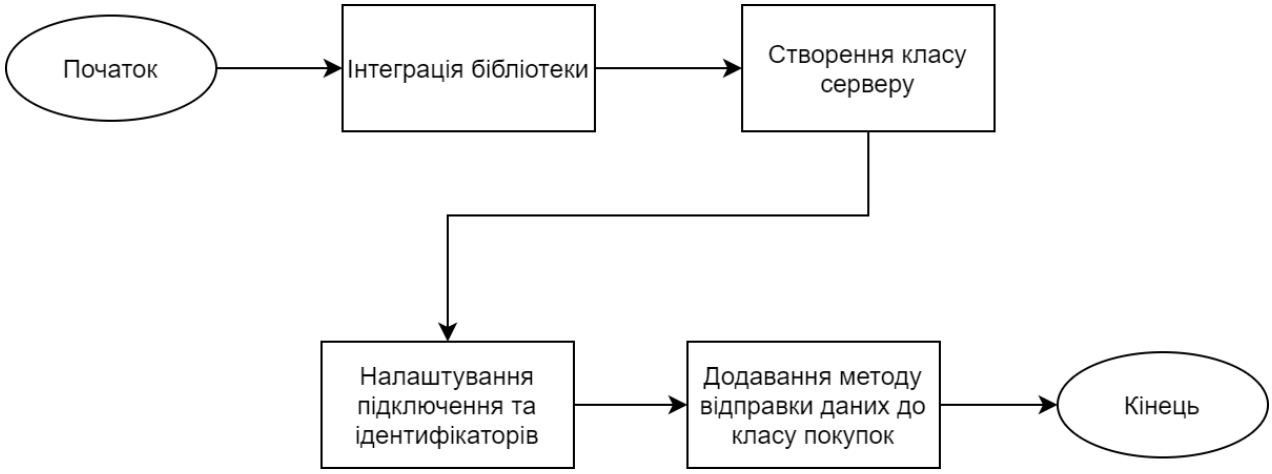

Рисунок 19 - Приклад встановлення аналітики

Після виклику методу, на сервер відправляються два повідомлення – повідомлення з параметром аналітики, та повідомлення з даними. Для коректного відправлення обох повідомлень, в методі відправки визначається довжина кожного рядка, перед відправкою, і ця довжина відправляється на сервер перед самим повідомленням. При цьому, довжина рядка яка визначає довжину рядка, який описує дані задана ідентичною для кожного компоненту системи.[13]

5.2 Структура серверу для обробки підключень

Серверна частина відповідає за правильну обробку підключень, а також за передачу даних між БД та додатками. На рисунку 20 можна побачити структуру серверної частини.

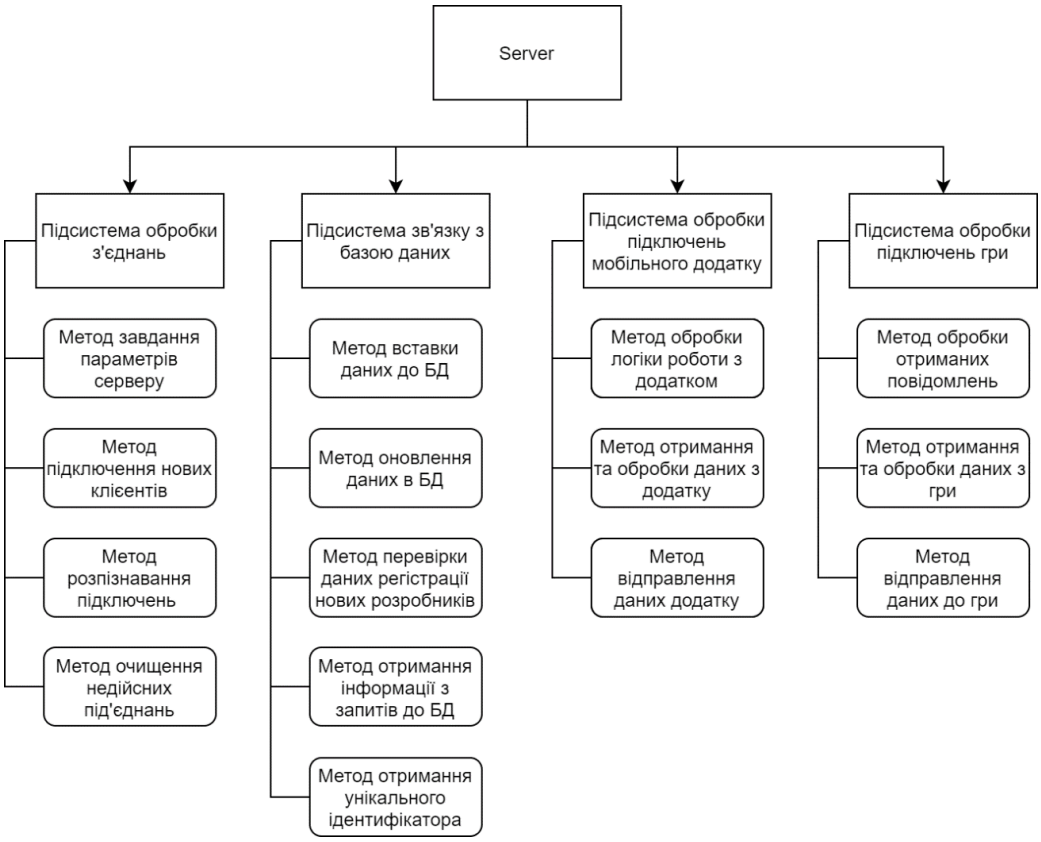

Рисунок 20 - Структура серверної частини

Для написання серверного підключення був обраний TCP протокол, який забезпечує легку та надійну передачу даних між мережевими вузлами.

Опис підсистеми обробки з’єднань

Головний клас розробленої програми, під назвою Connector відповідає за визначення основних параметрів серверу, та обробку логіки підключень.

Клас містить алгоритм обробки підключень, який визначає тип підключення — підключення з гри або підключення з мобільного додатку. Тип підключення визначається за допомогою обробки першого отриманого повідомлення. Наприклад, під час встановлення з’єднання, гра завжди буде відправляти свій ідентифікатор, у якого є унікальна частина, яка використовується лише в грі. В іншому випадку, підключення було зроблено з мобільного додатку.

Після визначення типу підключення, сервер створює окремий потік та новий екземпляр одного з класів: ServerToDevApp(підсистема обробки підключень мобільного додатку) або ServerToApp(підсистема обробки підключень гри), які відповідають за правильний обмін даними з додатками. Екземпляри створених класів додаються до окремного списку усіх об’єктів підключення.

Якщо будь-яке з підключень стає розірваним, то клас обробки цього підключення залишається існувати в пам’яті програми. Для уникнення ситуації з переповнення пам’яті, був написаний метод, який проходить по усім об’єктам зі списку підключення, і видаляє усі об’єкти які вже не використовуються.

Опис підсистеми обробки підключень мобільного додатку

Після створення об’єкту обробки підключення мобільного додатку, він починає працювати у окремому потоці, не перешкоджаючи розпізнаванню інших підключень.

ServerToDevApp — клас обробки даних з мобільним додатком відображення аналітики. Він відповідає за дозвіл авторизації користувача, отримання запитів до інформації про проєкти та їх аналітику, а також відправлення даних з сервера до додатку.

Так як мобільний додаток підключається саме в час реєстрації або авторизації, то першим чином, в конструкторі визначається ініціалізація користувача. Сервер отримує логін і пароль, з якими і буде працювати. Більш того, логін і пароль записуються в параметри підключення, для випадків, коли треба обробити окреме підключення.

Якщо в даному підключені зроблена спроба реєстрації, то серверна частина перевіряє наявність збігів логіна з даними в базі даних. По результатах запиту з бази даних до додатку відправляються дані про можливість або неможливість реєстрації. Неможливість реєстрації може бути по різних причинах. Одні визначаються в мобільному додатку – короткий пароль або логін, або недоступні символи, а інші причини визначаються на сервері – неможливість використання логіну для реєстрації, так як він вже зареєстрований іншим користувачем. Якщо реєстрація можлива, то сервер відправляє до БД запит на занесення даних про нового користувача в базу даних.

Під час авторизації, сервер, за допомогою запиту до БД визначає збіги не тільки за логіном, а і за паролем. Обробка запиту результатів запиту лягає на сервер. За результатами обробки результатів до додатку відсилаються дані про можливість авторизації.

Алгоритм підключення зображений на рисунку 21.

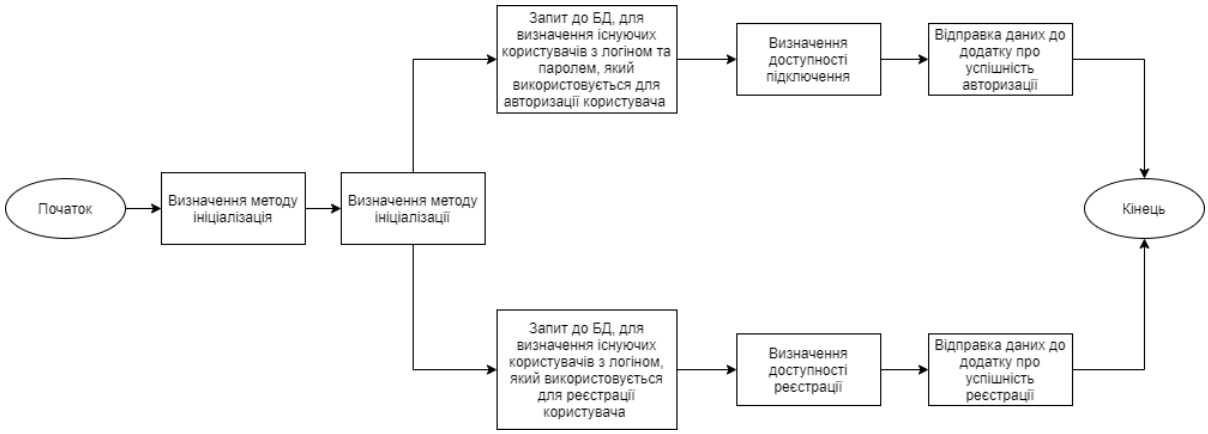

Рисунок 21 - Алгоритм підключення додатку до сервера

Основний метод класу налаштований таким чином, що буде приймати будь-які дані з додатку, і оброблювати їх згідно алгоритму. Дані, які приходять на сервер,

першим чином проходять перевірку на те, яка дія виконується в мобільному додатку. Таким чином, сервер може по різному оброблювати надіслану йому інформацію – перевіряти назву нового проєкту та створювати унікальні ідентифікатори, або виконувати запити до бази даних.

Коли користувач створює новий проєкт, то на сервер відправляється назва, яку ввів користувач. Назва перевіряється на унікальність, для даного користувача, за допомогою запиту до бази даних. Якщо перевірка на унікальність була успішно пройдена, то сервер створює унікальний ідентифікатор гри.

Для створення ідентифікатора гри, сервер робить запит до БД, і знаходить останній створений ідентифікатор. Якщо такого немає, то створюється перший ідентифікатор, з номером “00000001”. У випадку якщо хоча б один ідентифікатор вже існує в базі даних, то він переводиться в чисельний вигляд, після чого створюється нове значення ідентифікатора, на одиницю більше ніж попереднє. Отримане чисельне значення переводиться в назад в строкове представлення – рахується кількість цифр в числі, а число переводиться до string типу даних, після чого до нього додаються нулі, яких не вистачало для правильного представлення ідентифікатора. Після цього, створений ідентифікатор передається до додатку.

Запити до БД, які надсилаються до сервера з додатку передаються до підсистеми обміну даних з базою даних, а отримані значення записуються до списку типу String, та відправляються до додатку.

Опис підсистеми обробки підключень з гри

ServerToApp — клас обробки підключень з гри. Він налаштований на ідентифікацію кожного підключення з гри і на приймання аналітичних даних. Для ідентифікації, кожному підключенню присвоюється ідентифікатор гравця і ідентифікатор гри, який передається серверу під час підключення.

Ідентифікатор гри розробники копіюють з мобільного додатку, коли створюють новий проєкт і вставляють в гру, під час інтеграції бібліотеки підключення. Потім,

цей ідентифікатор можна буде знайти на екрані вибору проєктів у мобільному додатку.

Ідентифікатор гравця створюється для клієнта під час першого підключення до серверу. Коли гравець підключається до сервера, то гра передає неповний ідентифікатор гравця серверу у вигляді – “user-id-”. Сервер розпізнає ідентифікатор, і робить запит до бази даних, який би визначив останній створений ідентифікатор, або повернув пусту строку, якщо ідентифікаторів ще не було створено. На основі отриманих даних, сервер генерує новий ідентифікатор гравця, ідентично до генерування ідентифікатору додатку.

Після отримання ідентифікатора, сервер передає його до бібліотеки підключення, до гри. Отриманий ідентифікатор має зберігатись розробником на клієнті, для подальшої ідентифікації гравця.

Якщо ідентифікатор був стертий з клієнта під час оновлення прогресу, то гравець буде ідентифікуватись як новий, і буде мати окремі аналітичні звіти.[14]

Опис підсистеми зв’язку з базою даних

Для обробки запитів до БД був створений окремий клас – ServerToDB. Цей клас відповідає за підключення до бази даних, запитів до неї та отримання даних. Для різних типів запитів було створено різні методи для обробки інформації:

— Запити для отримання одного значення – заносились в окрему змінну і передавались до потрібного класу підключення

— Запити для отримання масиву даних – заносились в масив даних, де кожні декілька записів відповідали одному кортежу з бази даних. У такому ж вигляді вони передавались до додатків. Для уникнення ситуації з неправильною кількістю отриманих даних, кількість очікуваних атрибутів передавалась з додатків, під час відправки запиту на отримання даних.

— Запити на зміну та внесення даних в таблицю аналітики бази даних – такі типи запитів отримується лише від підключення з гри. Ці два запити були

винесені в окремі методи для створення гнучкої структури програмного продукту, та потенціалом на розширення.

— Запит на внесення даних в таблицю розробників та ігор – запити виконуються під час реєстрації нових розробників, або створення об’єкту гри в мобільному додатку.

Після завершення сеансу гри, сервер закриває з’єднання з клієнтом, і примусово видаляє екземпляр класу підключення, який вже не буде використовуватись.

5.3 Структура мобільного додатку

Як вже було розглянуто раніше, Unity має особливу структуру проєктів. Найважливішим об’єктом в ієрархії розробленого додатку є сцена, на якій розміщується декілька об’єктів при її завантажені – об’єкт під’єднання до серверу та об’єкт управління екранами. Ці елементи додатку були створені в рамках виконання дипломного проекту. Окрім них, на сцену завантажується камера, та менеджер подій.

Для правильного відображення користувацького інтерфейсу, камера була налаштована статично, і не була прив’язана до інших об’єктів. У свою чергу, Canvas-елементи мали посилання на об’єкт камери, та відображали усі створені екрани в над будь-якими іншими створеними об’єктами.

Під час запуску, компонент сцени який відповідає за управління екранами створює новий об’єкт – екран авторизації, з деякими заданими параметрами, а потім переходить в режим очікування, поки не буде викликана функція зміни екрану.

Елемент управління екранами був розроблений для простого видалення та додавання екранів до вікна перегляду. Він дозволяє не губити посилання на усі можливі екрани в додатку, і дозволяє створювати їх з одного місця, викликаючи функції з різних об’єктів. Окрім цього, функцією елемента є зберігання даних про поточний екран, який бачить користувач, та переключання камери на нього, під час змін в додатку.

Додаток можна поділити на декілька екранів, які будуть розглянуті у наступних розділах. Екрани додатку були створені за допомогою об’єкту Canvas. Елемент Canvas використовується для розміщення UI елементів, і дозволяє динамічно масштабувати його дочірні об’єкти, в залежності від розширення екрану.

Особливості екрану авторизації

Екран авторизації — пропонує розробнику декілька опцій: створити новий акаунт або використати існуючий, для отримання доступу до аналітики.

Дані авторизації можна зберігати на клієнті, якщо натиснути на toggle елемент з назвою “Remember Me”. Цей елемент зберігає дані авторизації за допомогою PlayerPrefs, якщо він увімкнений, і видаляє дані з додатка, якщо його відключити. Метод зберігання даних також використовується під час входу до акаунту, на випадок якщо після зберігання дані були змінені.

Екран авторизації містить поле помилок, в яке виводяться усі помилки авторизація, якщо вони виникають. На рисунку 22 показано екран реєстрації з помилкою:

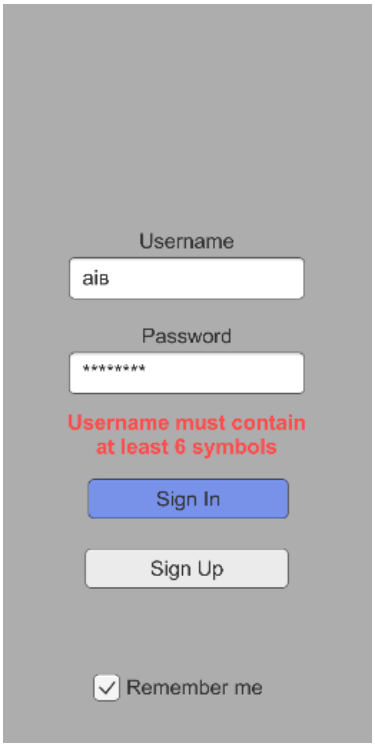

Рисунок 22 - Екран реєстрації з помилкою

Помилки авторизації можуть бути наступними:

— Введені дані невірні

— Реєстрація неможлива, через те що ім’я акаунта вже зайняте

— Ім’я акаунта має містити хоча б шість символів

— Пароль має містити хоча б вісім символів

— Неможливо під’єднатись до сервера

Опис екрану проектів

Екран проектів — після авторизації, програма робить запит до БД і створює об'єкти проєктів. Під час завантаження даних про проекти, додаток відображає не більше ніж сім елементів за раз, але записує усі дані до списку. На рисунку 23 зображено екран проектів.

Рисунок 23 - Екран проектів

Якщо проектів більше ніж можна відобразити на екрані, то кількість сторінок збільшується відповідно до кількості проектів. Перегортаючи сторінки, дані про проекти не завантажуються з бази даних, а використовується збережений список.

Алгоритм роботи з екраном створення проектів

Екран створення проєктів. На екрані створення проектів, розробник може ввести нову, унікальну, назву для свого проєкту, та отримати ідентифікаційний код гри. Для назв проекту були введені деякі обмеження:

— Назви не можуть повторюватись для одного і того ж розробника

— Назва проєкту не можу бути коротша за 4 символи, і більша ніж 20

Перевірки назв відбуваються як на клієнті, так і на сервері. У випадку незадовільної назви, розробник буде сповіщений про це спеціальним повідомленням.

Якщо ж назва задовольняє умовам, то сервер згенерує новий ідентифікатор гри, базуючись на останньому занесеному в базу даних ідентифікатору, і відправить його в мобільний додаток, розробнику.

Екран вибору аналітики

Екран вибору аналітики — під час створення екрану, об’єкт робить запит до БД, який дозволяє отримати усі актуальні параметри аналітики. Отримані параметри заносяться до спеціального UI елементу, який дозволяє обирати одну з опцій.

Коли користувач змінює період часу, то створюється новий запит до бази даних, на отримання дат, в яких є хоч один запис. Отримані дані наповнюють Dropdown компонент датами, за якими можна проводити сортування.

Запит будується по частинах, що дозволяє змінювати окремі частини які відповідають за сортування, типи полів та проміжки часу.

Дані можна сортувати по годинах останніх 30 днів, днях останніх 4 тижнів та тижнях останнього місяця. Від вибору періоду буде відрізнятись графічне представлення аналітики.

Опис модулів та алгоритмів під час використання екрану відображення аналітики

Екран зображення аналітики — виводить аналітичні дані на екран у вигляді графіків та тексту. Для більш зручного порівняння даних за різні проміжки часу, було розроблено спеціальний алгоритм відображення даних.

Для коректного відображення даних на екрані зображення аналітики, був написаний модуль програми, який відповідає за відображення даних.

Коли програма робить запит на отримання даних до бази даних, то на виході отримує масив лише з існуючими даними за потрібний період, залишаючи пустими ті часові проміжки, коли нічого не відбувалось. За допомогою написаного модулю, програма доповнює масив нульовими даними по годинам, дням та тижням. На виході, отримаємо масив даних, за усі дні вибраного проміжку, а не тільки ті, за якими є дані в БД.

Дані в аналітиці можуть дуже відрізнятись, як по своїй кількості, так і по значенням. Занадто великі дані, або занадто маленькі потребують зручного відображення на екрані телефону. Тому, для їх зображення, був написаний алгоритм стандартизації відображення даних у графічному представлені.

Ще одним окремим алгоритмом є коректне відображення днів тиждня, під час відображення аналітики за увесь місяць або за тижні. Для годин вибірка проста – вона починається з нульової години і закінчується 23, для будь-яких випадків. Проблема відображення полягає у тому, що під час вставки нульових даних неможливо визначити, які саме дні існують в аналітичній вибірці.

Для цього, дата початку вибірки, яка зберігається в звичайному форматі, переводилась в день поточного року. До цього числа покроково додавались дні, і

переводились назад в дату звичайного формату. Такий алгоритм використовувався як для екрану вибору аналітики, так і для екрану її відображення

Алгоритм зображення даних можна побачити на рисунку 24.

Рисунок 24 — Алгоритм зображення даних.

Максимальне значення для поточної вибірки стає коефіцієнтом розміру графічних елементів. Таким чином, графічний елемент з максимальним значення вибірки буде мати максимальний дозволений розмір, а розмір інших графічних елементів буде залежати від їх значення відносно максимального.

В проміжок часу 23-00 в гру зайшло 3 унікальних користувача, а в проміжку 22-23 — 1 унікальний користувач. При цьому, розмір останнього графічного елементу буде той самий, навіть якщо б його значення збільшилось в десятки або навіть сотні разів.

5.4 Структура баз даних

База даних була створена за допомогою MySql Workbench, і відповідає усім стандартам розроблення баз даних.

Створену базу даних можна поділити на п’ять таблиці:

— Таблиця розробників — зберігає дані про усіх розробників: їхні нікнейми та паролі, а також унікальні ідентифікатори розробників.

— Таблиця ігор — зберігає назви ігор, ідентифікатори розробників та унікальні ідентифікатори ігор

— Таблиця аналітики — зберігає ідентифікатори ігор, назви параметрів, дані прив'язані до параметрів, унікальні ідентифікатори користувачів та час, в який був зроблений запис.

— Таблиця з типами гравців. Ці типи відповідають за сегментацію гравців, що дозволить робити більш розгорнуті звіти. Гравці можуть бути не платящими, або належати до одної з категорій сегментування.

— Таблиця з жанрами ігор — відповідає за розподіл ігор по жанрам, що дасть можливість зробити конкурентне порівняння своєї гри, з іншими подібними.

На рисунку 25 зображено структура бази даних, та зв'язки між таблицями.

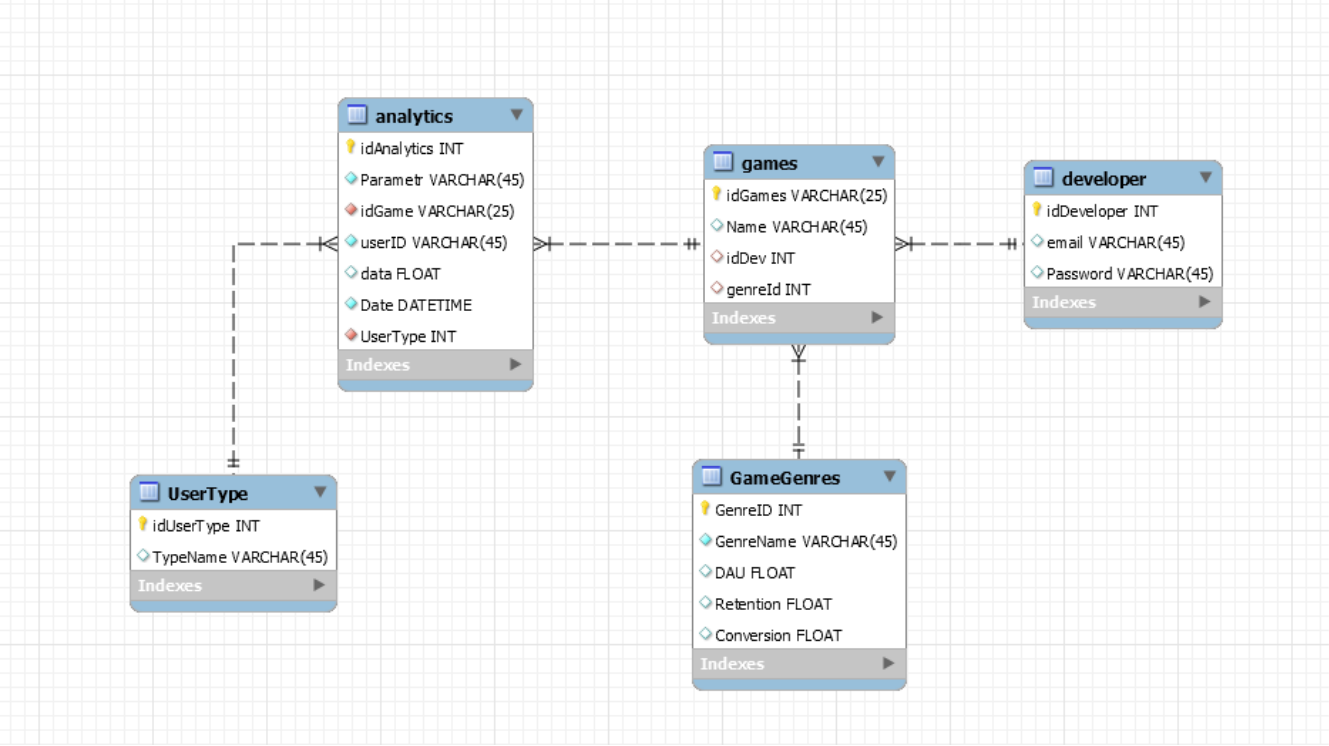

Рисунок 25 — Структура БД

Усі зв'язки між таблицями таблицями мають тим – один до багатьох. Таким чином, один розробник може мати декілька ігор, а ігри можуть мати безліч аналітичних параметрів та записів.

Для правильного видалення даних з таблиць, до бази даних було додано каскадне редагування та каскадне видалення даних. Таким чином було забезпечено безпечність записів таблиці.

Представлена структура бази даних повністю задовольняє потреби розробленої системи – зберігання даних по користувачам та їх сегментування, можливість конкурентного порівняння гри, зберігання аналітики проектів по користувацьким параметрам.

6. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для використання розробленої системи аналітики і коректної її роботи, треба дотримуватись деяких умов розроблення програмного забезпечення, а також провести підготовчу роботу та ознайомитись з інструкцією по експлуатації.

6.1 Умови розроблення ігри

Головною умовою використання системи аналітики є розроблення програмного продукту за допомогою мови програмування C#. Це обумовлено тим, що один з компонентів системи для інтеграції у гру розроблений за допомогою C#, і не може інтегруватись з будь-якими іншими мовами програмування.

6.2 Підготовка до інтеграції бібліотеки підключення

Для використання аналітичної системи, розробник має завантажити та встановити мобільний додаток аналітики.

Далі, треба зареєструвати нового користувача або пройти систему авторизації, якщо акаунт вже створений. На рисунку 26 показаний екран авторизації:

Рисунок 26 — Екран авторизації

Екран авторизації переводить користувача до вікна проєктів, на якому розробник може створити новий проєкт, ввівши бажану назву. Екран створення проєкту можна побачити на рисунку 27.

Рисунок 27 — Екран створення нового проєкту

Після створення проєкту, розробник отримує унікальний ідентифікатор проєкту у вигляді — app-id-XXXXXXXX, де X — цифра.

На цьому етапі підготовка завершується і можна переходити до етапу інтеграції бібліотеки підключення.

6.3 Інтеграція бібліотеки підключення

Для інтегрування бібліотеки підключення, треба завантажити файл бібліотеки і додати його до свого C# проєкту. Після цього в розробника буде можливість створити об'єкт класу Server.

Перед використання аналітики, треба зробити декілька дій:

— Об’явити змінну user-id, в якій, у клієнті, буде зберігатись ідентифікатор гравця.

— Встановити ідентифікатор гравця Server.user-id на той, що збережений у проєкті, або на порожній рядок, якщо це перший запуск гри.

— Встановити ідентифікатор гри, отриманий з мобільного додатку у вигляді app-id-XXXXXXXX.

— У створеного об’єкту треба викликати метод connect, який забезпечить підключення до серверу.

— Після методу connect треба зберегти змінну Server.user-id у клієнті. Зберігання цієї змінної розробник може виконати як забажає. В Unity це можна зробити за допомогою PlayerPrefs — простий спосіб зберігання даних у вигляді “Ключ, значення”.

6.4 Встановлення аналітики

Для відправлення даних аналітики на сервер треба викликати метод Server.sendData(string, float).

Так як у аналітики багато способів використання, то для універсальності, спосіб її встановлення має лягати на розробника. Для збирання аналітики, розробник має викликати один з методів бібліотеки, з параметрами назви аналітики, та даними. Але складність полягає у тому, що місця виклику методів розробник має обирати сам.

Наприклад, для збирання даних за сесіями користувача, можна викликати цей метод з параметрами “Sessions, 1” під час завантаження базового класу гри. Тобто, кожен раз, при запуску гри, це повідомлення буде відправлятись на сервер, і вноситись до таблиці аналітики.

А для збирання аналітики за внутрішньоігровими покупками, можна прив'язати цей метод з параметрами “shop_buy, “ціна”” до покупки гравця.

6.5 Алгоритм пошуку даних в додатку

Одразу після авторизації, створений раніше проєкт буде відображатися у списку проєктів. Для кожного проекту створюється об’єкт з його назвою, ідентифікатором гри та кнопкою параметрів. Натиснувши на кнопку параметрів проекту, відкриється екран з різними налаштуваннями вібирки даних.

Екран сортування можна побачити на рисунку 28:

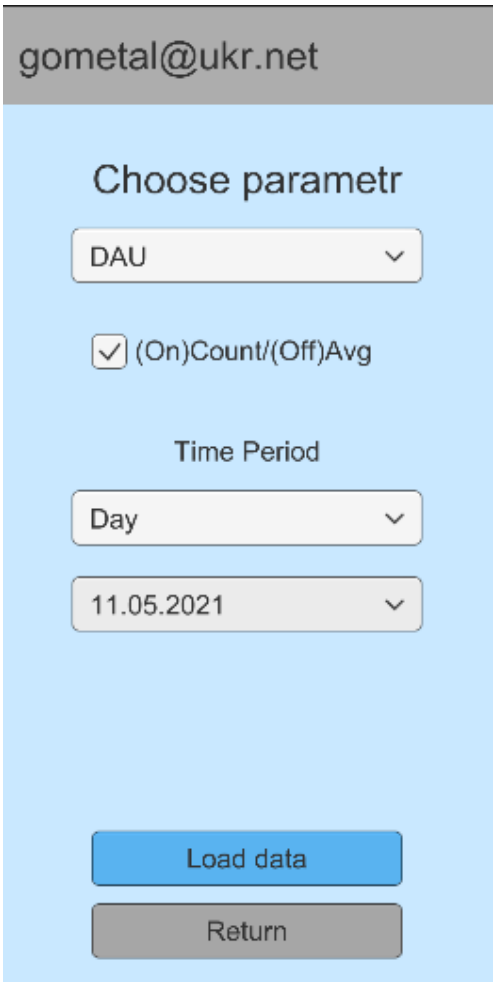

Рисунок 28 — Екран вибору аналітики

Для зображення розгорнутої аналітики були введені такі параметри вибірки:

— Вибір параметра аналітики

— Метод зображення — порахувати кількість записів з параметром або середнє значення даних, які відносяться до заданого параметра

— Сортування за різними періодами часу: години, дні, неділі

— Сортування за конкретним днем або тижнем, залежно від обраного періоду часу

Після налаштування усіх параметрів і натискання на кнопку отримання аналітики, додаток змінює екран. Отримані з БД дані представляються у зручному графічному та текстовому виглядах одночасно. Графічне представлення дозволяє швидко знайти піки аналітики, а текстові дані створюють завершений опис програмного продукту. Графічне та текстове представлення даних в додатку зображене на рисунку 29.

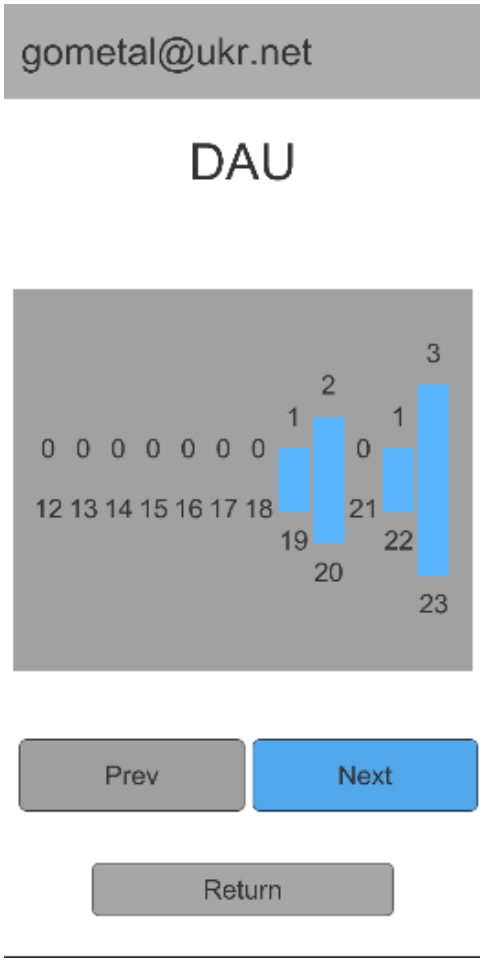

Рисунок 29 - Графічне та текстове представлення даних в додатку

Для зміни параметра можна швидко повернутись на попередній екран додатку, а кнопки “Next” та “Previous” дозволяють переключатись між даними різних проміжків часу, в межах одного параметру.

ВИСНОВКИ

Дипломну роботу присвячено актуальному для IT ринка напрямку – збору аналітики додатків, а саме – ігор. При вирішені задач поставлених у першому розділі, було зроблено наступне:

1. Проаналізовано ринок систем аналітики. Досліджено системи аналітики, такі як GameAnalitycs, SensorTower, DevToDev, та вбудовану Google аналітику. Визначено сильні та слабкі сторони усіх проаналізованих систем.
2. Проаналізовано методи збору аналітичних даних, та обрано найзручніший з методів, для збору аналітики додатків - подієвий метод
3. Обрані найзручніші методи розробки програмного забезпечення, а саме – мова програмування C#, для розробки сервера та мобільного додатку, рушій Unity та система управління базами даних MySql
4. Створено структуру та архітектуру системи, які забезпечують гнучку роботу з кожним компонентом системи та дозволяють легко масштабувати систему
5. Розроблено та поєднано компоненти системи аналітики – бібліотеку з’єднання з сервером, сервер, БД, мобільний додаток для відображення аналітики

Програмне забезпечення розроблено мовою програмування C# для платформи .Net в середовищі Visual Studio 2019.

В ході виконання дипломної роботи були досліджені та порівняні методи та системи для збору аналітичних даних ігор. На основі проведеного дослідження було розроблено систему аналітики, яка може збирати будь-які аналітичні дані проєкту, легко інтегрується в ігри створені за допомогою мови програмування C#, та відображає зібрані аналітичні дані в зручному, для користувача, представлені.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Magy Seif El-Nasr, Anders Drachen, Alessandro Canossa, Game Analytics. Maximizing the Value of Player Data / Magy Seif El-Nasr, Anders Drachen, Alessandro Canossa // Springer - 2013. — 791 p.
2. Сабиров В. Игра в цифры. Как аналитика позволяет видео играм жить лучше / Сабиров В. // Бомбора, 2020. — 376
3. О “китах” и удержании пользователей в условно-бесплатных приложениях [Електронний ресурс]. — Режим доступу: https://app2top.ru/columns/e-rik-syofert-o-kitah-i-uderzhanii-pol-zovatelej-v-uslovno-besplatny-h-prilozheniyah-65482.html
4. Как выглядит идеальная система аналитики для игрового проекта. [Електронний ресурс]. — Режим доступу: https://vc.ru/flood/13832-game-analytics-2
5. Session-based-аналитика должна умереть? Будущее за user-centric? [Електронний ресурс]. — Режим доступу: https://vc.ru/marketing/153892-session-based-analitika-dolzhna-umeret-budushchee-za-user-centric
6. Когортный анализ. Метрики продукта vs метрики роста [Електронний ресурс]. — Режим доступу: http://gopractice.ru/cohort_analysis/
7. Сравнение систем аналитики для игровых проектов — MixPanel, Localytics, Flurry, devtodev, deltaDNA и GameAnalytics. [Електронний ресурс]. — Режим доступу: https://vc.ru/flood/12121-game-analytics
8. Karl E. Wiegers, Joy Beatty. Software Requirements [Електронний ресурс] / Karl E. Wiegers, Joy Beatty., — Режим доступу: https://www.booksfree.org/wp-content/uploads/2020/07/Wiegers-K.pdf
9. Общие сведения о платформе .Net [Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview

10. Руководство Unity [Електронний ресурс]. — Режим доступу: https://docs.unity3d.com/ru/530/Manual/UnityManual.html
11. Джозеф Хокінг Unity в действии. Мультиплатформенная разработка на C# / Джозеф Хокінг // Питер, 2016. — 336
12. Основы работы с MySQL Workbench: быстрый старт, управление схемой данных [Електронний ресурс]. — Режим доступу: https://mithrandir.ru/professional/soft-and-hardware/mysql-workbench-basics.html
13. Интеграция системы аналитики в игру [Електронний ресурс]. — Режим доступу: https://dtf.ru/gamedev/13555-integraciya-sistemy-analitiki-v-igru
14. Клиент-серверное приложение на сокетах TCP [Електронний ресурс]. — Режим доступу: https://metanit.com/sharp/net/3.2.php

ДОДАТОК А

Система аналітики додатків для розробників ігор

Специфікація

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР71236_21Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ ”_ТЕФ_АПЕПС_ТР71236_21Б 81-1	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ ”_ТЕФ_АПЕПС_ТР71236_21Б 12-1	ServerListener.dll	Бібліотека підключення гри до сервера
УКР.НТУУ”КПІ ”_ТЕФ_АПЕПС_ТР71236_21Б 12-2	Analitycs.apk	Мобільний додаток для відображення аналітичної інформації
УКР.НТУУ”КПІ ”_ТЕФ_АПЕПС_ ТР71236_21Б 12-3	Server.exe	Програма для даних з гри та запитів до бази даних
УКР.НТУУ”КПІ ”_ТЕФ_АПЕПС_ ТР71236_21Б 12-4	AnalyticsBD.mwb	База даних для зберігання аналітики

ДОДАТОК Б

Система аналітики додатків для розробників ігор

Текст програми

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР71236_21Б 12-2

Аркушів 5

Київ 2019

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Analitic : MonoBehaviour
{
    public string message;
    public int partsCount;

    public Server script;
    public GameObject GraphPart;
    public GameObject Back;

    public List<GameObject> allDataGraphs;
    public List<string> newData;

    public Text parametrName;

    int pageCount;
    int currentPage;

    int MaxSize = 100;
    float maxData;

    float width;
    float space;
    float OneWidth;
    DateTime week;

    // Start is called before the first frame update
    void Start()
    {
        script = GameObject.Find("ServerListener").GetComponent<Server>();
        message = GameObject.Find("Inside-screen").GetComponent<InsideScript>().message;
        partsCount = GameObject.Find("Inside-screen").GetComponent<InsideScript>().countParts;
        parametrName.text = GameObject.Find("Inside-screen").GetComponent<InsideScript>().parametr;
        week = GameObject.Find("Inside-screen").GetComponent<InsideScript>().date;

        script.sendData(message);
        script.sendData("3");

        List<string> Analitics = script.getDataFromBD();
        allDataGraphs = new List<GameObject>();


        pageCount = 1;

        switch (partsCount)
        {
            case 24:
                pageCount = 2;
                break;
            case 7:
                pageCount = 1;
                break;
            case 4:
                pageCount = 1;
                break;
        }


        newData = addEmpty(Analitics);

        maxData = getMax(newData);

        width = 220f;
        space = width/(newData.Count/(3*pageCount));
        OneWidth = space - 2;

        int counter = 0;
        for (int i = 0; i < newData.Count / pageCount; i+=3)
        {
            Vector3 position = new Vector3(-width/2 + (counter) * space + OneWidth/2, 0, 0);
```

```
            GameObject graph = Instantiate(GraphPart, Back.transform);
            graph.GetComponent<GraphicScript>().constructor(newData[i + 1], newData[i + 2], position, MaxSize *
(float.Parse(newData[i + 1]))/ maxData, OneWidth);

            allDataGraphs.Add(graph);

            counter++;
        }

        currentPage = 1;
    }

    List<string> addEmpty(List<string> data)
    {
        List<string> newData = new List<string>();
        List<string> dates = new List<string>();
        List<string> tempdata = new List<string>();

        List<string> d = new List<string>();

        switch(partsCount)
        {
            case 24:
                for(int i = 0; i < partsCount; i++)
                {
                    d.Add(i.ToString());
                }
                break;
            case 7:
                for (int i = 0; i < partsCount; i++)
                {
                    DateTime tempdate = week;

                    d.Add((Convert.ToInt32(tempdate.AddDays(-partsCount+i + 1).Day)).ToString());
                }
                break;
            case 4:
                for (int i = 0; i < partsCount; i++)
                {
                    DateTime date = DateTime.Today;
                    d.Add((Convert.ToInt32(date.DayOfYear)/7 - partsCount + i + 2).ToString());
                }
                break;
        }

        for (int k = 2; k < data.Count; k+=3)
        {
            dates.Add(data[k]);
        }

        for (int k = 1; k < data.Count; k += 3)
        {
            tempdata.Add(data[k]);
        }


        for (int i = 0; i < partsCount; i++)
        {
            float date = -1;

            int index = dates.IndexOf(d[i].ToString());

            if (index == -1)
            {
                newData.Add(parametrName.text);
                newData.Add("0");

                if (partsCount == 4)
                {

                    string week = getWeek(d[i]);
                    newData.Add(week);
                }
                else
                {
                    newData.Add(d[i]);
```

```
                }
            }
            else
            {
                newData.Add(parametrName.text);
                newData.Add(tempdata[dates.IndexOf(d[i].ToString())]);

                if (partsCount == 4)
                {

                    string week = getWeek(d[i]);
                    newData.Add(week);
                }
                else
                {
                    newData.Add(d[i]);
                }

            }

        }

        return newData;
    }

    string getWeek(string week)
    {
        int weekNumber = Convert.ToInt32(week);
        int currentYear = DateTime.Now.Year;
        var startDate = new DateTime(currentYear, 1, 4);
        int firstMonday = startDate.DayOfWeek == DayOfWeek.Sunday ? 6 : (int)startDate.DayOfWeek - 1;

        int offsetToDemandedMonday = -firstMonday + 7 * (weekNumber - 1); // смещение к искомому понедельнику, в
днях
        var mondayOfTheGivenWeek = (startDate + new TimeSpan(offsetToDemandedMonday, 0, 0, 0)); // вычисляем дату
искомого понедельника

        int offsetToTheEnd = -firstMonday + 7 * weekNumber - 1;
        var endweek = (startDate + new TimeSpan(offsetToTheEnd, 0, 0, 0));


        return mondayOfTheGivenWeek.Day.ToString() + "-" + endweek.Day.ToString();
    }

    float getMax(List<string> data)
    {
        float maxSize = 0;

        for (int i = 0; i < data.Count; i+=3)
        {
            if(i==0)
            {
                maxSize = float.Parse(data[i+1]);
            }

            if(float.Parse(data[i+1]) > maxSize)
            {
                maxSize = float.Parse(data[i + 1]);
            }

        }

        if (maxSize == 0)
            maxSize = 1;
        return maxSize;
    }

    public void nextData()
    {
        if (currentPage < pageCount)
        {

            for (int i = allDataGraphs.Count - 1; i >= 0; i--)
            {
                Destroy(allDataGraphs[i]);
            }
```

```
            allDataGraphs.Clear();

            int counter = 0;
            for (int i = currentPage * (newData.Count / 2) ; i < (currentPage + 1) * (newData.Count / 2); i+=3)
            {

                Vector3 position = new Vector3(-width / 2 + (counter) * space + OneWidth / 2, 0, 0);

                GameObject graph = Instantiate(GraphPart, Back.transform);
                graph.GetComponent<GraphicScript>().constructor(newData[i + 1], newData[i + 2], position, MaxSize *
(float.Parse(newData[i + 1])) / maxData, OneWidth);

                allDataGraphs.Add(graph);
                counter++;
            }

            currentPage++;
        }
    }

    public void prevData()
    {
        if (currentPage > 1)
        {

            for (int i = allDataGraphs.Count - 1; i >= 0; i--)
            {
                Destroy(allDataGraphs[i]);
            }
            allDataGraphs.Clear();

            int counter = 0;
            for (int i = (currentPage-2) * newData.Count / 2; i < (currentPage - 1) * newData.Count / 2; i+=3)
            {

                Vector3 position = new Vector3(-width / 2 + (counter) * space + OneWidth / 2, 0, 0);

                GameObject graph = Instantiate(GraphPart, Back.transform);
                graph.GetComponent<GraphicScript>().constructor(newData[i + 1], newData[i + 2], position, MaxSize *
(float.Parse(newData[i + 1])) / maxData, OneWidth);

                allDataGraphs.Add(graph);
                counter++;
            }

            currentPage--;
        }
    }

    public void returnClicked()
    {
        GameObject.Find("Inside-screen").GetComponent<InsideScript>().chooseView(GameObject.Find("Inside-
screen").GetComponent<InsideScript>().gameName);
    }
}
```

ДОДАТОК В

Система аналітики додатків для розробників ігор

Опис програми

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР71236_21Б 13-1

Аркушів 9

Київ 2019

АНОТАЦІЯ

Розроблена система містить чотири компоненти. Бібліотека зв’язку з сервером — відповідає за зв’язок гри з сервером. Ця бібліотека інтегрується в гру як окремий файл, і може бути використана для з’єднання з сервером, відправлення даних на сервер та приймання даних. Бібліотека носить назву ServerListener.dll і розроблена за допомогою платформи .Net.

Мобільний додаток для розробників, який показує дані аналітики. Дані показуються у графічному та текстовому вигляді, що дозволяє порівнювати їх між собою у межах однієї вибірки. Додаток створений в рішії Unity – безкоштовному середовищі для розробки додатків.

Сервер для обробки підключень та розподілу інформації. Сервер працює одразу у двох напрямках — на з’єднання з грою, і з’єднання з мобільним додатком. Для розроблення серверу була використана платформа .Net.

База даних для збереження даних аналітики. Приймає запити від серверу та віддає потрібні дані. Має таблиці розробників, ігор та параметрів аналітики, а також додаткові таблиці для жанрів ігор і таблиці для типів гравців. Для створення БД була використана СУБД MySql та графічна оболонка MySql Workbench.

Програмне забезпечення було розроблене за допомогою програмного середовища Visual Studio Community 2019 об’єктно-орієнтованою мовою C#.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Одна частина компонентів системи працює на операційній системі Windows 10, а інша Android-пристроях. Для функціонування не потребує ніяких попередньо встановлених програмних засобів.

Такі компоненти системи, як сервер та бібліотека підключення були розроблені на об’єктно-орієнтованій мові програмування C# за допомогою середовища розробки Visual Studio Community 2019. Для розроблення мобільного додатку був використаний рушій Unity – безкоштовне рішення для створення кросплатформених додатків.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Програмна система вирішує завдання збору аналітичних даних з ігор, та їх відображення в окремому додатку, для розробників.

Програмна система призначена для збору аналітичної інформації з ігор, обробки інформації, її сортування та відображення у зручному для користувача вигляді.

Функціональних обмежень на використання розроблених компонентів не має.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Перед початком роботи з системою, розробник має скачати та встановити мобільний додаток аналітичної системи, в якій він має зареєструватись, створити новий проєкт гри та отримати унікальний ідентифікатор.

Після цього, в попередньо розроблену гру, розробник має інтегрувати бібліотеку підключення та встановити в ній виданий унікальний ідентифікатор гри, та налаштувати код для роботи системи.

Останнім кроком в роботі з системою є перевірка отриманих значень в мобільному додатку. В меню створеного проекту, розробник може обрати параметр аналітики для відображення, та його тип сортування.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Розроблені частини програмного продукту, такі як сервер обробки та бібліотека підключення працюють на базі платформи .Net та операційної системи Windows 10, в той час як мобільний додаток був розроблений за допомогою UnityEngine та працює на Android-системі. Система не потребує додаткових засобів для функціонування.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для роботи з системою, треба завантажити apk файл мобільного додатку відображення даних, та встановити його на телефон.

Для роботи, бібліотека підключення потребує завантаження та інтегрування до проекту гри.

Система не потребує додаткових сторонніх програмних засобів. Завантажуваний файл додатку займає 18 мегабайтів, а сам додаток має невеликі системні вимоги.

Для того щоб перестати користуватись системою, достатньо прибрати з проєкту бібліотеку підключення та виклики методів відправки даних, а також видалити програму аналітики з телефону.

ВХІДНІ І ВИХІДНІ ДАНІ

Система потребує вхідні дані для авторизації в мобільному додатку, такі як унікальне ім’я користувача та пароль. Також, в грі потрібно визначити параметри, які будуть відправлятись на сервер. Це можна зробити під час налаштування гри, або дані будуть створюватись на етапі виконання програмного продукту.

Вихідними даними є набір аналітичних даних у графічному та текстовому виглядах, які відображають обрані метрики гри.

ДОДАТОК Г

Система аналітики додатків для розробників ігор

Апробація

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР71236_21Б

Аркушів 5

Київ 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XIX Міжнародної науково-практичної конференції молодих вчених і студентів м. Київ, 20–23 квітня 2021 року

ТОМ 2

Київ- 2021

УДК 004.054

Студент 4 курсу, гр. ТР-71 Редько В. І.
Проф., д.т.н. Аушева Н.М.

СИСТЕМА ЗБОРУ АНАЛІТИКИ ПРОЕКТУ ДЛЯ РОЗРОБНИКІВ ІГОР

В одній з найбільш швидко розвиваються галузей мультимедійних розваг дуже важливо мати уявлення про швидкозмінні інтереси користувачів і володіти конфіденційною інформацією, що дозволяє зберігати унікальність продукту.

Ігрові студії часто використовують аналітику, як перший крок до створення нових проектів. Більш того, будь-які рішення щодо розвитку вже випущених ігор будуються на основі аналізу даних. Але одним з головних досягнень стала система на основі збору аналітики - сегментація користувачів, яка є найбільш успішним підходом до торгівлі внутрішньоігровими предметами. Сегментація гравців - умовний поділ користувачів, в залежності від їх внутрішньоігрових покупок, яке дозволяє створювати унікальні пропозиції для певних категорій гравців [1].

Більшість готових рішень для мобільного ринку дозволяють збирати базову інформацію про користувачів:

- Retention – утримання користувача
- Conversion - метрика, що описує другий крок потенційного користувача: від кліка по рекламному креативу до установки.
- LTV (life time value) - довічна цінність користувача
- CPI (cost per install) - вартість за установку. Ця метрика показує, скільки було витрачено на залучення одного гравця, за допомогою реклами
- DAU (daily active users) - середня кількість щоденних активних користувачів.

Хоча ці метрики є всього невеликим відсотком функціоналу аналітичних систем, вони дозволяють вирішувати більшість проблем геймдизайнерів. Однак, для ведення успішного проекту тільки цих метрик недостатньо. Мета розробника - максимізувати прибуток і найчастіше важливо знати метрики не всього проекту, а конкретної механіки, щоб розуміти, чому вона подобається, або не подобається користувачам. Такий підхід до аналізу механік монетизації дозволяє розробляти їх незалежно від проекту, що значно скорочує час розробки майбутніх додатків, за рахунок повторного використання вже готових рішень [2].

Як було сказано раніше, аналогів системи багато. Наприклад, Google або Apple пропонує вбудовану аналітику для додатків завантажуються в Google Play або AppStore, проте зібрані дані дуже обмежені.

	Metrics	Funnels	Segmentation	Traffic Analysis	Push-notifications	Game Metrics	Accuracy	1'000 MAU	10'000 MAU	100'000 MAU
Mixpanel	+	+	+	+	+	-	+	Free	$250	$2000
Localytics	+	+	+	+	+	-	+	Free	$200 or $400	$120 and more
Flurry	± No financial metrics	± limitations	± limitations	+	-	-	-	Free	Free	Free
devtodev	+	±	+	+	+	+	+	$5	$50	$500
deltaDNA	+	+	+	+	+	+	+	Free	free or $150 (if MAU >10k)	$1350 and more
GameAnalytics	+	+	± limitations	+	-	-	-	Free	Free	Free

Рисунок 1 - Таблиця порівняння систем аналітики

На відміну від аналітичної системи Google, такі сервіси як Devtodev, MixPanel,

GameAnalitics можуть збирати не тільки базові дані, але і дозволяють створювати користувачеві власні події, які потрібно відстежувати. Ці ж сервіси можуть сегментувати користувачів за різними параметрами, що дає можливість налаштовувати гру під конкретну аудиторію. Також, однією з додаткових фіч цих систем є менеджмент Push-повідомлень, які дозволяють збільшити утримання користувачів в проекті, підвищуючи прибуток десь на 10-20 відсотків.

Якщо більшістю свого функціоналу системи схожі, то вибір розробників падає на ту систему, яка в першу чергу дешевше, а по-друге - легше інтегрується в проект. Більшість систем надає безкоштовний доступ до аналітики, якщо у вашої програми менше ніж 1000 активних користувачів на місяць, однак починаючи з 10000 користувачів, розробникам доведеться платити в районі 250 доларів на місяць, і з кількістю користувачів ціна тільки зросте [3].

Відокремлено існує аналітична система sensortower. Вона дозволяє не тільки оцінювати власний додаток, але і порівнювати отримані дані з іншими додатками. Її використання допомагає розробникам коректно складати сторінки додатків, оцінювати ефективність реклами на основі середніх показників ринку, просувати додаток в магазині або збирати звичайну аналітику.

Такий підхід до аналітики особливо цінний, тому що розробник не знає всіх показників ринку не зможе тверезо оцінювати такі дані як Retention, CPI або Conversion.

Зважаючи на все вищесказане, метою даної роботи було обрано створення системи зі збору і зручного відображення аналітики будь-якого типу, яка не буде прив'язана до якихось базових показників проектів, і дозволяє розробникам максимізувати прибуток, оцінювати показники на ринку і створювати кращий користувальницький досвід.

Одним з принципів створення системи була простота використання. Не у всіх є математична освіта, і система аналітики не повинна його вимагати від розробника. Інтерфейс і функціональність системи аналітики, набір встановлених в ній звітів повинні бути максимально простими для розуміння, тому процес доступу до даних був реалізований максимально просто.

Створена система аналітики містить:

- Модуль збору інформації, який буде впроваджуватися в проект на будь-якому з етапів розробки або експлуатації. Цей модуль відповідає за збір і передачу інформації на сервер. При цьому, за допомогою такого модуля користувач зможе легко створювати власні події, в будь-яких потрібних місцях.
- Сервер, на якому буде зберігатися і оброблятися інформація. Сервер буде приймати інформацію від гри, сортувати її за гравцями, днями та показниками і відправляти дані в мобільний додаток.
- Додаток для відображення в зручному вигляді оброблених даних. Перш ніж інтегрувати модуль збору в додаток, розробнику потрібно буде створити сторінку гри в додатку, після чого програма видає унікальний номер, який треба вставити в код гри. При відправці повідомлень на сервер, цей номер дозволить визначити, до якого додатку відносяться дані.

Такий програмний продукт дозволяє будь-якому розробнику швидше вчитися на своїх помилках, аналізуючи дані і перевіряючи теорії про побудову ігрового ринку.

Перелік посилань:

1. Magy Seif El-Nasr, Anders Drachen, Alessandro Canossa., Game Analytics. Maximizing the Value of Player Data - 2013. – 791 p.
2. Как выглядит идеальная система аналитики для игрового проекта. [Електронний ресурс]. Режим доступу: https://vc.ru/flood/13832-game-analytics-2
3. Сравнение систем аналитики для игровых проектов — MixPanel, Localytics, Flurry, devtodev, deltaDNA и GameAnalytics. [Електронний ресурс]. Режим доступу: https://vc.ru/flood/12121-game-analytics