ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерія програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ПІДТРИМКИ МЕРЕЖІ ІГРОВИХ СЕРВЕРІВ MINECRAFT З ВИКОРИСТАННЯМ ПЛАГІНІВ НА ОСНОВІ SKRIPT»**

Виконав: студент 4 курсу, групи ПД-44 спеціальності

121 Інженерії програмного забезпечення
(шифр і назва спеціальності)

Посенко Д. Р.
(прізвище та ініціали)

Керівник Золотухіна О.А.,
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Нормконтроль_____
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - "Бакалавр"

Спеціальність - 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
інженерії програмного забезпечення
_____ Негоденко О.В.

" ___"_____ 2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Посенко Данило Романович
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка веб-застосунку для підтримки мережі ігрових серверів Minecraft з використанням плагінів на основі Skript »
2. Керівник роботи к.т.н., доц. Золотухіна Оксана Анатоліївна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "24" лютого 2023 року №26

Строк подання студентом роботи "1" червня 2023 року

3. Вихідні дані до роботи:
   3.1. IntelliJ IDEA;
   3.2. Sublime Text;
   3.3. Figma;
   3.4. Офіційна документація мови програмування Java
   3.5. Офіційна документація мови програмування Kotlin
   3.6. Офіційна документація системи управління базами даних MySQL

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   4.1. Визначення особливостей роботи з Minecraft серверами
   4.2. Аналіз ринку Minecraft серверів
   4.3. Проектування системи
   4.4. Проектування інтерфейсу користувача
   4.5. Розробка додатку

5. Перелік графічного матеріалу
   5.1. Мета, об’єкт, предмет дослідження
   5.2. Задачі дипломної роботи
   5.3. Аналіз аналогів
   5.4. Вимоги до програмного забезпечення
   5.5. Програмні засоби реалізації
   5.6. Діаграма варіантів використання
   5.7. Діаграма послідовності поповнення балансу
   5.8. Діаграма архітектури веб-застосунку
   5.9. Схема бази даних
   5.10. Екранні форми
   5.11. Демонстрація роботи системи оплати
   5.12. Апробація результатів дослідження
6. Дата видачі завдання “25” лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз ринку Minecraft серверів	13.01-14.01	виконано
2	Вивчення та аналіз задачі	14.01-16.01	виконано
3	Розробка структури сервера Craftoriya	16.01-29.01	виконано
4	Розвобка веб-застосунку	29.01-10.02	виконано
5	Розробка дизайну	10.02-17.02	виконано
6	Встановлення на хостинг та налаштування	17.02-23.02	виконано
7	Оформлення пояснювальної записки	01.04-05.05	виконано
8	Розробка обов’язкових демонстраційних матеріалів	05.05-10.05	виконано
9	Попередній захист роботи	22.05	виконано
10	Подання роботи в деканат	30.05	

Студент _____ Посенко Д.Р.
(підпис) (прізвище та ініціали)
Керівник роботи _____ Золотухіна О.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 49 с., рис., 1 табл., 13 джерел.

Об`єкт дослідження – процес взаємодії користувача з ігровим Minecraft сервером.

Предмет дослідження – програмне забезпечення для підтримки Minecraft серверів.

Мета роботи – спрощення взаємодії користувача з сервером Minecraft за рахунок удосконалення системи донату та підвищення зручності особистого кабінету, що реалізовані з використанням плагінів на основі Skript.

Методи дослідження – аналіз технічної документації, методи тестування програмного забезпечення, експериментальне дослідження, опитування гравців та потенційної авдиторії геймерів.

У роботі було проведено аналіз використання гравцями веб-застосунків та внутрішьоігрових опцій для отримання позитивного ігрового досвіду та зручності певних процесів. Було проаналізовано вимоги визначеної категорії гравців до ігрових серверів, визначено необхідні завдання та шляхи їх спрощення.

Визначено необхідний для гравців функціонал, проаналізовано, який функціонал потрібно додати, а який виключити через його застарілість та неактуальність. Розроблений додаток дозволяє спростити роботу команди розробників, через що навіть одна людина може підтримувати функціонування технічної частини (веб-застосунок, бази даних, внутрішньоігрові процеси).

Галузь використання – сучасні ігрові сервери та мережі ігрових серверів як грі Minecraft так і будь-яких онлайн ігор.

ВЕБ-ЗАСТОСУНОК, MINECRAFT СЕРВЕР, БАЗА ДАНИХ, FONDY, JAVA, СЕРВЕРНЕ ЯДРО, SKRIPT, ПРОФІЛЬ ГРАВЦЯ, ВНУТРІШНЬОІГРОВИЙ МАГАЗИН, ВНУТРІШНЬОІГРОВА ВАЛЮТА

ЗМІСТ

ВСТУП **10**

1 ДОСЛІДЖЕННЯ НІШІ ІГРОВИХ СЕРВЕРІВ MINECRAFT ТА ПОСТАНОВКА ЗАДАЧІ ЩОДО ЇХ ВИРІШЕННЯ **12**

1.1 Аналіз ринку Minecraft серверів 12

1.2 Аналіз вимог та вподобань гравців 20

1.3. Особливості роботи зі спільнотою гравців Minecraft 22

1.4. Технічні особливості та типи веб-застосунків для Minecraft серверів .. 24

1.5. Постановка задач дипломної роботи 25

2 РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ MINECRAFT СЕРВЕРА **27**

2.1 Розробка та моделювання вимог до системи 27

2.2 Робробка структури бази даних 30

2.3 Розробка алгоритмів роботи основних функцій та архітектури системи 31

2.4 Обгрунтування вибору технологій для розробки програмного забезпечення 34

2.4.1 Java та Kotlin 34

2.4.2 HTML,CSS, JavaScript 34

2.4.3 Технологія API для взаємодії веб-застосунку з ігровою частиною Minecraft сервера 36

2.4.4 Вибір платіжної системи для здійснення поповнення особистого балансу гравцями Minecraft сервера 38

2.4.5 SKRIPT та особливості написання плагінів 38

2.4.6 Вибір ядра для клієнтської частини 39

2.5 Вибір середовища для розробки 41

2.5.1. SublimeText 41

2.5.2 IntelliJ IDEA 41

2.6 Розробка веб-застосунку для підтримки Minecraft сервера 42

2.6.1 Написання Frontend частини веб-застосунку 42
2.6.2 Написання Backend частини веб-застосунку 43
2.6.3 Написання плагінів для роботи системи оплати 44
2.7 Реалізація серверної частини та встановлення веб застосунку на хостинг 46
2.7.1 Порядок встановлення веб застосунку на сервер 46
2.7.2 Розробка бази даних MySQL 47
2.7.3 Отримання SSL сертифікату 48
2.7.4 Вибір доменного імені та прив'язка системи 49
2.8 Встановлення веб-застосунку та ігрової частини Minecraft сервера на хостинг. Запуск системи 51
3 ТЕСТУВАННЯ ТА ЗАПУСК МЕРЕЖІ МАЙНКРАФТ СЕРВЕРІВ 53
3.1 Тестування функціоналу веб-застосунку 53
3.2 Тестування "профілю гравця" 54
3.3 Публічний тест 56
ВИСНОВКИ 58
ПЕРЕЛІК ПОСИЛАНЬ 59
Додаток А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація) 60
Додаток Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ 67

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DNS – Domain name server

FTP – протокол для віддаленого завантаження файлів

SSH – протокол для віддаленого управління комп’ютером/сервером

SSL – Secure Sockets Layer, криптографічний протокол для встановлення безпечного з’єднання

БД – база даних

Коїн – внутрішньоігрова валюта

ПЗ – програмне забезпечення

СУБД – система управління базами даних

ФОП –фізична особа-підприємець

Хаб – місце збору гравців

ВСТУП

Успіх будь-якого ігрового серверу значною мірою залежить від розуміння гравцем його інтерфейсу та зручності виконання доступних функцій. Ніша Minecraft серверів та веб-застосунків для їх підтримки має велику кількість додатків, однак більшість з них не забезпечують повною мірою необхідні функціональні можливості. Також проблемним питанням є те, що більшість гравців з України продовжують грати на серверах країни-агресора, адже через надмірний вплив російського інфопростору на українських гравців склалось враження, що українські сервери не потрібні. Створення українських аналогів є надзвичайно актуальним, бо необхідно зменшити російський вплив на українське суспільство, а привернути увагу до вітчизняного продукту можна лише якістю та зручністю.

Об’єктом дослідження є процес взаємодії користувача з Minecraft сервером та веб-застосунком для виначення шляху оптимальної реалізації готового продукту.

Предметом дослідження є програмне забезпечення для підтримки Minecraft серверів.

Метою роботи є спрощення взаємодії користувача з сервером Minecraft за рахунок удосконалення системи донату та підвищення зручності особистого кабінету, що реалізовані з використанням плагінів на основі Skript.

Для досягнення поставленої мети було сформульовано нижченаведені задачі:

– провести аналіз ринку Minecraft серверів та веб-застосунків для підтримки Minecraft серверів, визначити вимоги та вподобання гравців;

– визначити функціональні та нефункціональні вимоги до сервера та веб-застосунку;

– провести аналіз засобів для розробки програмного забезпечення;

– розробити архітектуру веб-застосунку;

– розробити модуль системи оплати;

– розробити та реалізувати архітектуру бази даних для функціонування системи оплати та збереження інформації про гравців;

– розробити модуль, що реалізує особистий кабінет (профіль) гравця;

– розгорнути та протестувати розроблену систему.

Мережа ігрових серверів – це першочергово логіка переходу гравця з одного серверу на інший в рамках єдиного сервера, що відповідає за загальні дані та має роль хабу. Веб-застосунок в свою чергу виконує роль рекламної сторінки та має функціонал поповнення внутрішньоігрового балансу гравця, що дає змогу отримувати кошти за проведену роботу та масштабуватись. Розроблену систему було розгорнуто та виконано заходи з її підтримки, що дало змогу зацікавити українських гравців вітчизняним продуктом та викликати інтерес у медійному просторі. Завдяки цьому в подальшому є можливість залучити більшу команду розробників та створювати унікальні ігрові режими, якими можна наповнювати сервер.

1 ДОСЛІДЖЕННЯ НІШІ ІГРОВИХ СЕРВЕРІВ MINECRAFT ТА ПОСТАНОВКА ЗАДАЧІ ЩОДО ЇХ ВИРІШЕННЯ

1.1 Аналіз ринку Minecraft серверів

Можна виділити кілька окремих напрямкі0в Minecraft серверів за їх спеціалізацією.

1. Приватні сервери – відбирають людей за їх якостями або за певну плату.

2. Публічні сервери – сервери, на які може зайти будь-який користувач.

3. Особисті сервери – зазвичай, створює одна людина на своєму ПК без жодних налаштувань для компанії друзів, або ж, пограти 1-2 вечори.

В роботі проаналізовано ринок українських серверів. Їх перелік є в реєстрі Організація Українських Майнкрафтерів (ОУМ), екранні форма якого наведена на рис 1.1. Реєстр відображає лише публічні та приватні сервери. На рис 1.2 демонструється відображення серверів у катологзі на прикладі сервера «UA POLIT». [12]

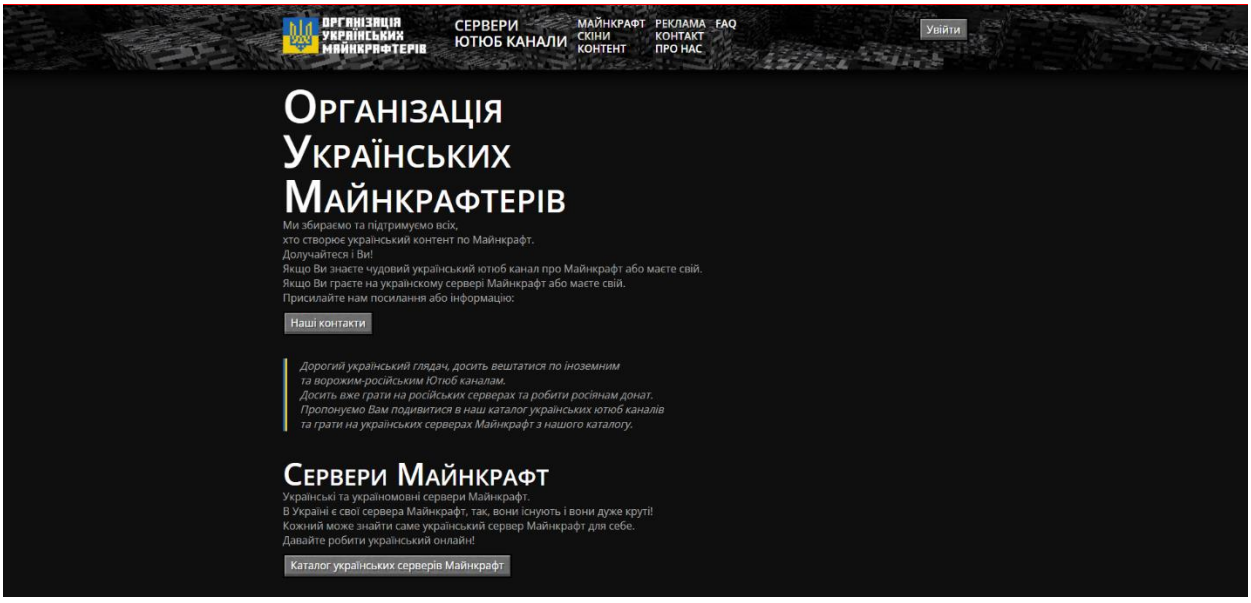


Рисунок 1.1 - Реєстр Організації Українських Майнкрафтерів

До огляду було взято сервер UA POLIT (рис 1.2). На разі це найпопулярніший український сервер, оскільки його тематика близька більшості гравців.

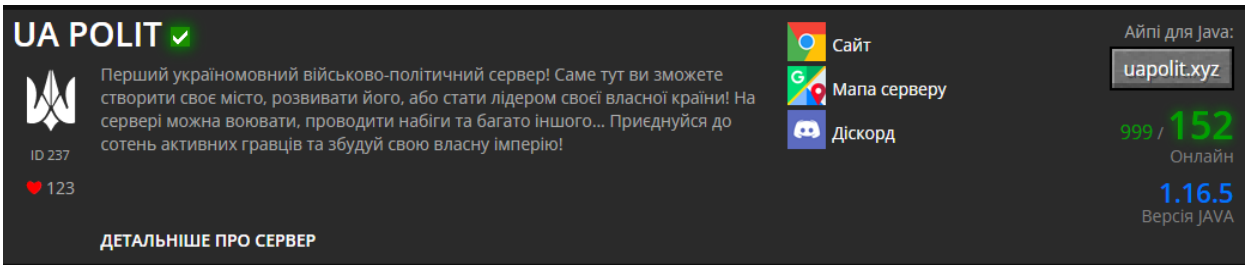


Рисунок 1.2 – Сервер UA POLIT

Саме цей сервер зміг виділитись тим, що найпершим дав гравцям змогу створювати власні віртуальні держави, міста, об’єднуватись, розвиватись або ж воювати. Якщо порівнювати з іншими серверами, то політична тематика гравцям дуже подобається, саме тому на ньому зберігається високий онлайн. Онлайн – це якісний показник, що демонструє зацікавленість аудиторії в даному сервері. Він залежить від якості сервера, регіону на який розрахований, до прикладу США чи Україна та типу сервера. Приватний він чи публічний. Для кожного сервера в залежності від тематики, кількість людей в онлайні може бути різною. Так, для приватного сервера рівень в 100 гравців на добу буде високим. Для сервера з міні-іграми та режимами нормальні показники починаються від 1000, залежно від кількості режимів та лобі, які наявні та в залежності від їх наповнення. Якщо є попит на якийсь конкретний режим, то в лобі буде створюватись черга, що дає зрозуміти про необхідність збільшення кількості ігор певної тематики. Проте є й інші сервери, які також можна бачити на цьому сайті. Зазвичай вони однотипні, пропонують звичайну гру в режимі виживання без жодних особливих ознак, що могли б їх виділити серед інших. Відповідно вони не можуть залучити достатню авдиторію, тому онлайн там менший, або взагалі відсутній. Це дає розуміння про проблеми таких серверів.

1. Гравцям це не цікаво.

2. В серверів є певні внутрішні недоліки, котрі можуть бути пов’язані з адміністрацією або ж розробкою, котрі в свою чергу впливають на бажання грати. Це можуть бути критичні баги або ж погана робота зі спільнотою гравців.

Виходячи з аналізу та опитувань виходить, що зацікавленість в первному

сервері залежить від технічних особливостей, що прямо впливають на ігровий досвід, в тому числі надійність і безвідмовність, а також соціальна робота з гравцями. Сервер – це єдиний організм, симбіоз гравців з адміністрацією. [1]

Коли сервер стає популярним через вище названі причини а також має закладену спроможність до масштабування, то в такому випадку він стає проєктом. Сервер стає проєктом рівно тоді, коли кількість гравців на ньому стабільна, та коли цей самий сервер включає в себе декілька режимів для виживання, або ж міні-ігор, що дає йому статус «Проєкт» або ж «Мережа ігрових серверів». Кількість гравців в онлайні першочергово залежить від тематики та ексклюзивних функцій, що сервер може дати. Якщо певний режим зацікавив гравця, то рівень онлайну буде рости, адже нові гравці будуть затримуватись а не виходити одразу після тестування.

На рис.1.3 зображені сервери, що є однотипними, вони нічим не виділяються з поміж інших, також, опираючись на опитування гравців, оскільки на цих серверах є безліч технічних проблем, хоч і на деяких з них є по кілька режимів – гравці не мають бажання там грати саме через відсутність особливостей.

Вибір тематики майбутнього серверу не менш важливий і залежить від потенційної авдиторії. Таким чином можна виділити певні категорії гравців та посортувати їх за віком. 10-15 років, 16-19 та 20+. Часто вони є несумісні, адже більшість з категорії 10-15 ніколи не оберуть сервер для будівництва, натомість будуть не проти пограти в щось більш казуальне, до прикладу «Анархія» чи класичне серверне виживання з донат привілеями.

Категорії 10-15 більш притаманна ідея класичних серверів, де витративши невеликі кошти, через привілеї та ігрові скіни можливо певним чином виділитись перед іншими, адже за проведеним аналізом, психологія 10-15 річних гравців працює саме так. Для них важливо бути кращими за інших, то ж такі сервери можуть бути досить прибуткові, проте статистично на таких серверах онлайн має триматись в кілька тисяч гравців, адже більшість з них – не є платіжеспроможньою авдиторією. Ідеальний варіант – публічний сервер з платним додатковим контентом.

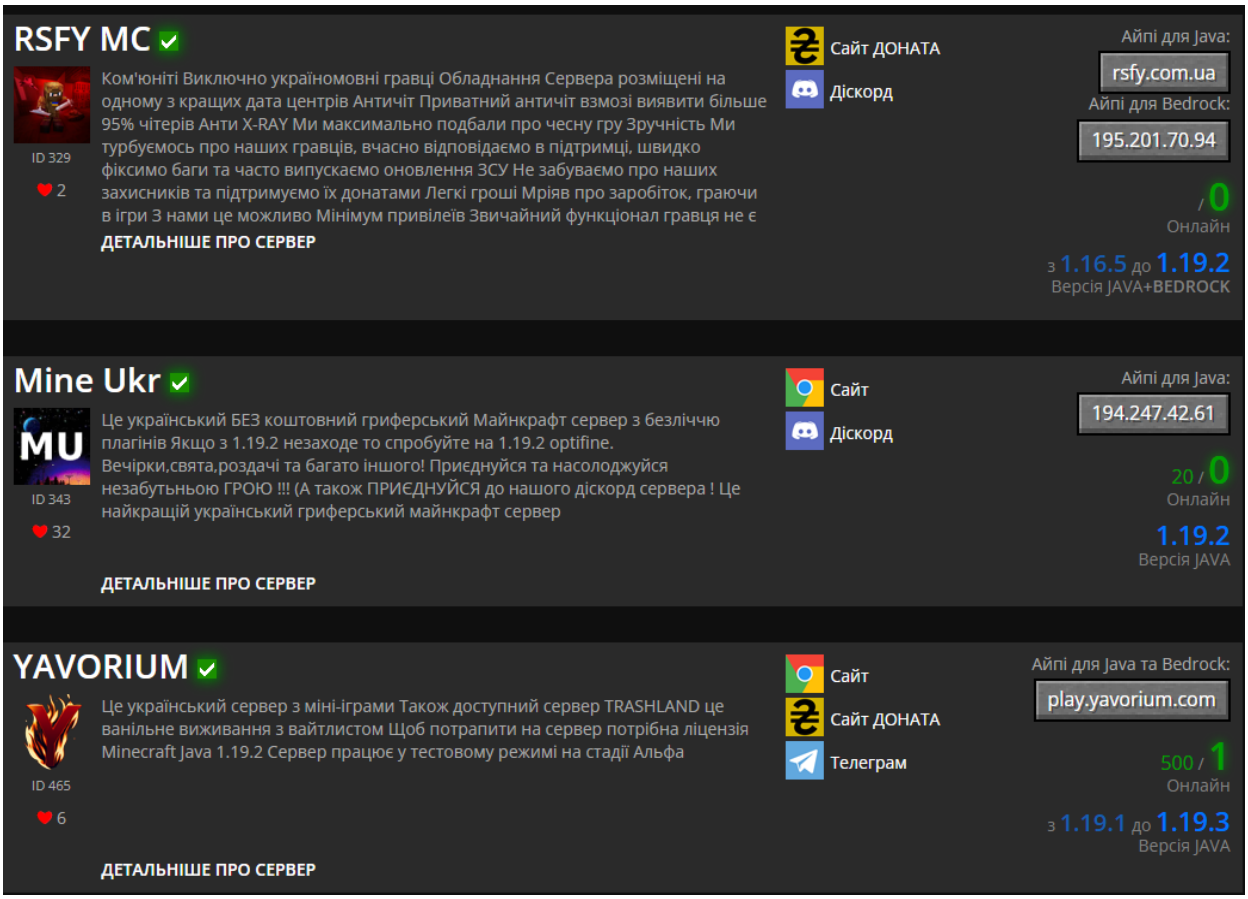


Рисунок 1.3 - Однотипні сервери

Таким чином, певна кількість гравців робить серверу рекламу своєю присутністю, частина з них – купує платний контент. На приватних серверах грають більш дорослі гравці (переважно від 16 років) за разову плату або ж підписку відчувають себе більш комфортно в більш закритій спільноті. Обидва варіанти мають свої переваги та недоліки. В публічних – має бути високий онлайн, щоб отримати прибуток. В приватних – більший прибуток за рахунок меншого онлайну, але таких гравців шукати складніше. Загалом може здатись, що ідея приватних серверів – краща, адже ресурсу на їх утримання треба менше, а платіжеспроможність гравців – більша, але це не завжди так. До кожного треба знаходити власний підхід, і якщо «школярі» в перспективі можуть принести прибутку більший, то й ресурсу на утримання сервера треба буде витрачати більше. Не факт, що кожен з них захоче щось придбати для себе, а от дорослі гравці навпаки, приносять стабільний прибуток і одразу. Якщо брати сервер по підписці

за можливість входу, то це стабільний прибуток. Якщо ж брати класичне виживання, то частіше там буде високий онлайн, але мінімум транзакцій. І якщо обирати між цими двома варіантами, то, вочевидь, збирати дорослу авдиторію завжди набагато краще.

На прикладі з UA POLIT можна бачити досить високий показник онлайну (рис.1.4), проте, проаналізувавши монетизацію, можна зрозуміти, що будь-який приватний сервер з правильною технічною реалізацією вподобань гравців принесе більше, адже сервер UA POLIT розрахований на широку авдиторію та має мінімальний поріг входу, зайти на нього можна безкоштовно. Це означає, що адміністрації потрібно більше, а за цілодобовий нагляд за дотриманням правил доведеться додатково платити. Це не є безпечно. Адже доведеться вводити певні обмеження, такі як «приват території» гравцями та донат привілеї. Це застаріла система Pay To Win, яка за гроші надає можливість використовувати певні команди, котрими при неправильному розподілі прав за невелику суму можна знищити сервер загалом. Єдиним виходом є верифікація людей, що купують собі такі права, але це першочергово знижує прибутковість сервера. Частіше верифікації просто не існує, натомість ціни на такі привілеї непомірно високі, купують їх не стабільно, то ж, дуже складно прорахувати бюджет. Саме тому, вихдячи з аналізу ринку, варіант з щомісячною підпискою та меншим онлайном – кращий за публічний сервер з продажем донат привілеїв та/або косметики. Косметика на таких серверах може бути лише додатковою опцією, робити її основним способом монетизації не маючи онлайну в кілька тисяч чоловік – не раціонально. Слід також зауважити, що чим більше гравців планується пускати на сервер – тим більше ресурсу буде потрібно на його утримання. В середньому утримання 100 гравців в онлайні (по потужностям) коштує близько 40-60$, залежно від обраного хостингу.

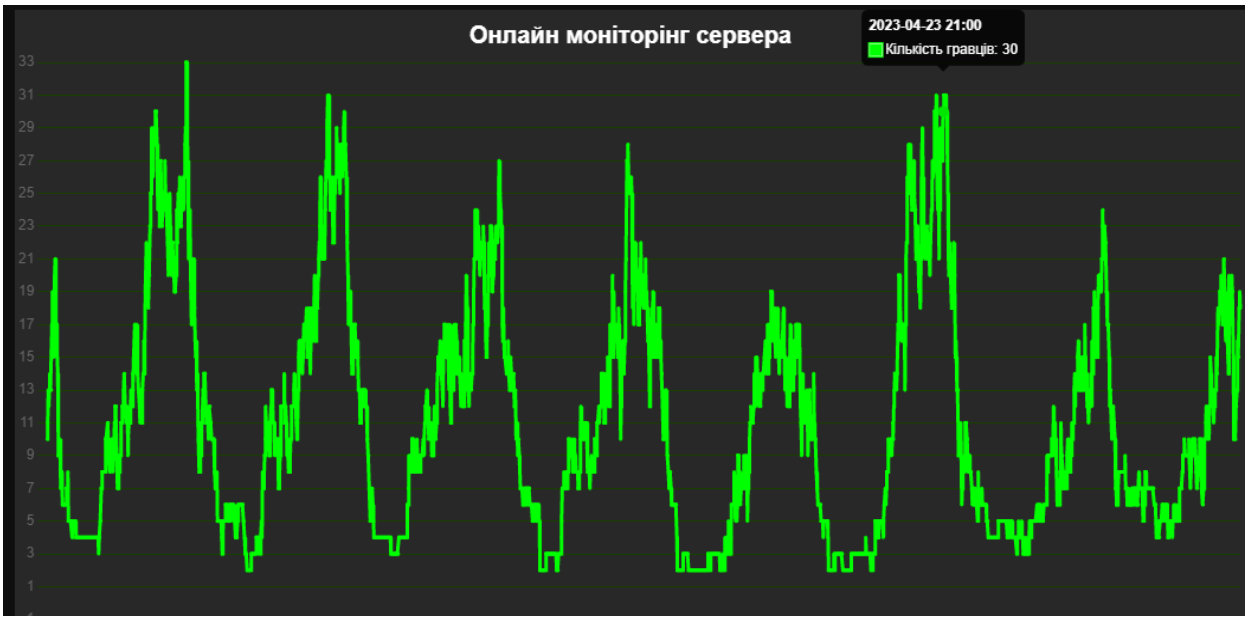


Рисунок 1.4 – Рівень онлайну на UA POLIT

Варто зазначити, що для гравців важливо, аби сервер був стабільним, щоб уникнути неприємних ситуацій, коли було витрачено кошти, а сервер з часом закривається по невідомій причині. Тому важливо зберігати стабільність роботи сервера, адже не може бути мови про перспективність, якими б цікавими не були ідеї закладені на початку, якщо сервер буде не стабільним, не коректно працювати та не виконувати належно ті функції, що в нього закладено. Перспективним Minecraft-проєктом може бути як вузьконаправлений приватний сервер, так і публічний. Проте важливо уникнути проблем зі складністю підтримки, для цього потрібно вміти реалізувати необхідні функції на досить актуальній, але стабільній версії, адже якщо мати справу з такою складною грою як Minecraft, де будь-яку сторонню роботу по покращенню проводять різноманітні групи гравців, не можна бути впевненим, що, до прикладу, вийшла версія 1.20, і в той же момент вийде серверне ядро. Так, офіційне ядро є, але через те, що воно завжди застаріле ще на момент публікації та не підтримується багатьма плагінами, певні групи гравців пишуть покращені версії серверного ядра та підтримують його. Найпопулярнішими ядрами є Paper, Spigot та Bukkit. Відповідно внутрішньоігрові плагіни, зазвичай, підтримують лише одне з запропонованого списку ядер. Через що розробка сервера на актуальній версії стає надто складною, адже часто треба

чекати ядро та актуальні плагіни для оптимізації, після чого реалізовувати власні технічні задачі.

На жаль, більшість українських гравців грають на серверах країни агресора, таких як «CRISTALIX». Першочергово там грають через те, що на сервері доступно більше 15 видів різноманітних міні-ігор. Це дає гравцям можливість знайти себе та пограти в режим, котрий їм найбільше симпатизує. До того ж, «Кристалікс» є досить стабільним сервером. Середній онлайн на ньому тримається в межах від 3000 до 5000 тисяч гравців одночасно, що є досить високий показник серед серверів, котрі не орієнтовані на західну авдиторію. Така велика авдиторія дозволяє зробити акцент на продажі косметики та певних «класів». Зі сторони гравця їх веб-застосунок є досить зручним в користуванні, хоча містить досить багато архаїчних моментів, таких як «особистий кабінет».

В сучасних реаліях веб застосунок для підтримки Minecraft сервера потрібен лише для того, аби мати змогу поповнити баланс на сервері та знайти потрібні посилання на «Діскорд», «Телеграм» канали, тощо.

Пік розвитку Minecraft «проєкта» серверів досягається тоді, коли у сервера з’являється власний лаунчер або ж авдиторія стає невід’ємною частиною його життя. Якщо порівняти «UA POLIT» та «CRISTALIX», можна побачити, що у першого веб-застосунок (сайт) відсутній взагалі. В другого він є і надає можливість переглядати кількість онлайну, завантажити власний лайунчер, поповнити баланс та увійти до особистого кабінету, де можна виконувати такі дії як: змінювати скін, нікнейм, пароль. Це не є зручно, адеже на разі існують технології, які надають гравцям можливість зробити це прямо в грі не відволікаючись та не відкриваючи нічого додатково. Зазвичай це досягається задопомогою написання власних плагінів або ж налаштування вже існуючих.

Чим більше якісних режимів на сервері – тим краще. В цьому випадку варто звернути увагу на те, що на «UA POLIT» є лише один режим, але він унікальний. Є й аналогічні сервери, проте через погану технічну реалізацю гравці зтикаються з великою кількістю багів. Якість, масштабування та спеціалізованість сервера залежать виключно від фінансування та професійності команди розробників.

Найбільш потужним у сфері Minecraft серверів є HYPIXEL (рис. 1.5). Це канадський сервер, що було запущено в 2013 році. На разі його середній онлайн становить близько 50 000 гравців одночасно. Це є дуже високим показником, але задля утримання такої кількості гравців треба мати власне ПЗ та ядро з переписаними алгоритмами роботи, враховуючи технічні особливості гри. З технічної точки зору він є надзвичайно складним. Кожна дрібниця на HYPIXEL зроблена вручну великою командою розробників. Майже всі плагіни на сервері, навіть досить класичні, написані вручну.

Рисунок 1.5 – веб-застосунок HYPIXEL

Проаналізувавши HYPIXEL, варто зазначити, що цей сервер приваблює гравців цікавістю та простотою. Перш за все він стабільний, регулярно оновлюється, має багато додаткового контенту та зручну для користувача систему донату на випадок, якщо той захоче придбати якийсь платний контент.

Веб-застосунок HYPIXEL є досить зручним з точки зору користувача, але має

багато речей, які складні для підтримки зі сторони розробника. До прикладу локальні форуми на разі переходять в Discord канали, як для зручності користування так і модерації [11]

Також існує 2 основні методи продажу платного контенту. Це є або внутрішня валюта, яку треба купувати через веб-застосунок за певним курсом, або ж купівля певного товару через веб-застосунок, котрий потім через визначену команду передається на акаунт гравця по ігровому нікнейму. Використовувати перший варіант значно простіше, актуальніше і безпечніше, адже веб-застонунки, як, фактично, додаткова опція з кількома ключовими моментами для підтримки сервера, перетворювались на, фактично, онлайн-магазини, в котрих іноді, навіть, забували додати кнопку пошуку, через що знайти щось потрібне було надто складно при великій кількості товару. Коли ж ти маєш внутрішні кошти, то процеси покупки спрощується в рази, адже відпадає необхідність займатись підтримкою веб-застосунку, сильно простіша і зрозуміліша версія доступна прямо в грі, котру може підтримувати, до прикладу, розробник, що спеціалізується виключно на плагінах.

1.2 Аналіз вимог та вподобань гравців

Робота зі спільнотою гри Minecraft є досить складною, адже вкрай необхідно вміти правильно працювати з нею, та на основі цього провести аналіз вимог та вподобань гравців, щоб мати можливість сформувати функціональні та не функціональні вимоги щодо розробки. Найкращим аналізом є опитування та мітинги в діскорді. Таким чином було проведено кілька опитувань щодо ключових аспектів сервера та веб-застосунку, а саме:

1. Яким має бути сервер? (рис. 1.6)
2. Веб-застосунок (сайт) сервера має містити такі функції: (рис. 1.7)
3. Чи користувалися б ви особистим кабінетом на постійній основі? (рис. 1.8)
4. Особистий кабінет краще мати на сайті чи в грі? (рис. 1.9)

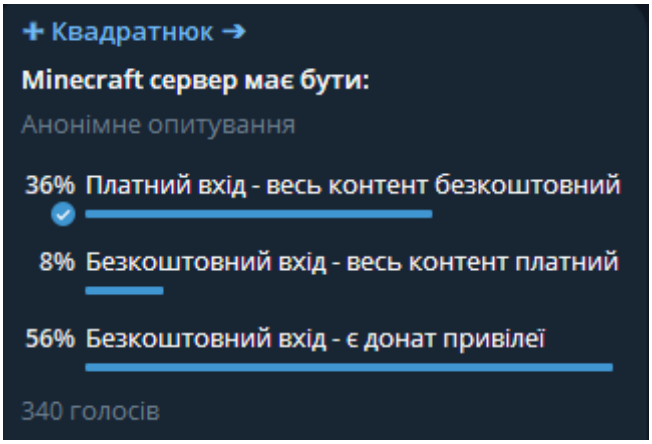


Рисунок 1.6 – Опитування 1

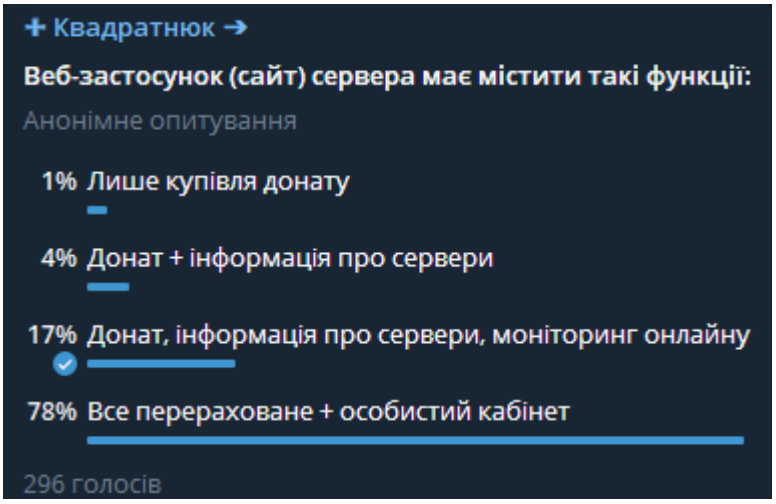


Рисунок 1.7 – Опитування 2

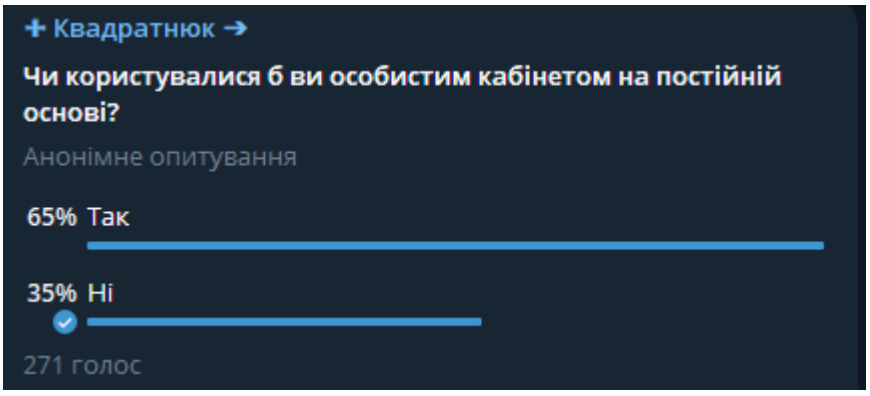


Рисунок 1.8 – Опитування 3

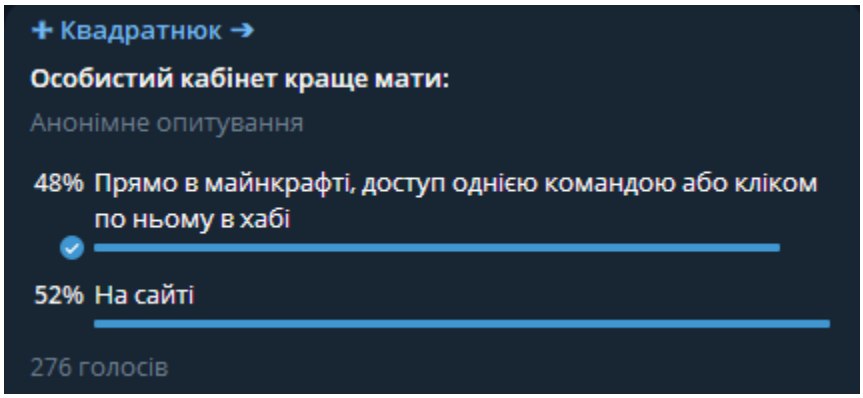


Рисунок 1.9 – Опитування 4

Таким чином, виходячи з першого опитування можна зробити висновок, що потрібно рухатись в сторону мережі серверів з кількома різними ідеями стосовно донату та свободи дій, оскільки спільнота розділилась, то задля охоплення більшої кількості гравців, необхідно робити одразу 2 сервери. Платний вхід – безкоштовний контент та безкоштовний вхід – донат привілеї. Виходячи з 2 опитування можна сказати, що для майбутніх гравців важливий пункт щодо донату, особистого кабінету, інформації та моніторингу. Проте знаючи, виходячи з аналізу, що на сайт гравці заходять лише для поповнення балансу або купівлі привілея, не буе правильним рішенням одразу робити особистий кабінет саме у веб-застосунку, тому в 3 та 4 опитуванні було уточнено цей момент. Думка гравців розділилась, що дає можливість обирати краще рішення на власний розсуд, або ж реалізувати обидва варіанти при масштабуванні, адже на відміну від першого, де важливо було знати де саме гравці бажають грати, 4 опитування, щодо кабінету, є опціональним. В даному випадку вірним рішенням буде робити особистий кабінет через гру за допомогою плагінів, якщо ж мова йде про сервер з модифікаціями, власними акаунтами та лаунчером – тоді потрібно робити Особистий кабінет саме на у веб-застосунку та дублювати його в лаунчері, задля безперешкодного встановлення ігрового скіна та нікнейму.

1.3. Особливості роботи зі спільнотою гравців Minecraft

Гравців Minecraft, як було згадано раніше, можна розділити на кілька

категорій. Як показує досвід людей, що тримають свої сервери по цій грі – не завжди всі гравці можуть бути між собою сумісними. В даному випадку складно притримуватись думки окремої категорії. До прикладу, якщо одна група бажає сервер безкоштовний, але з донат привілеями, а друга сервер з платним входом, але без руйнування ігрового балансу через ті самі привілеї, то вочевидь зібрати їх на одному сервері не вийде, адже він не буде подобатись як першій так і другій групі. Вірним рішенням може бути створення саме мережі серверів, де завдяки хабу буде розділення на кілька окремих серверів, проте зі спільною статистикою та досягненнями. Таким чином стає можливим збільшення онлайну та потенційного прибутку проєкту, адже вдається без компромісів дати кожній групі гравців те, що вони прагнуть найбільше. Так, виходячи з проведених опитувань щодо вимог та вподобань гравців стає зрозуміло, що в багатьох аспектах думки авдиторії є максимально суперечливими. В цьому полягає складність роботи зі спільнотою, адже перед тим, як почати розробку сервера та веб-застосунку – необхідно зрозуміти, якого саме формату має бути майбутній проєкт аби стати успішним. Не потрібно паразитувати саме на «українській» тематиці, адже те, що сервер є українським не робить його ідеальним автоматично. Не буде вірним рішенням назвати сервер «паляниця», та встановити на нього український ресурспак. Це більше маркетинг, але сервер має першочергово бути досконалим з технічного боку.

Оскільки стабільність сервера формує саме спільнота людей, що там грають, розробнику необхідно прислухатись не лише до більшості, а й до вийняткових ідей чи скарг. Найкращим способом дізнатись думку спільноти є опитування в телеграмі, на ютубі чи в Діскорді. Також джерелом інформації можуть бути мітинги, де кожен гравець може висловити свою думку щодо певного оновлення чи свої побажання щодо наступних. Лояльність спільноти сприяє збільшенню онлайну та кількисті гравців, що захочуть придбати платний контент, аби підтримати розвиток проєкту. Гравців потрібно заохочувати різноманітними винагородами, до прикладу медальками чи тимчасово надавати певні привілеї. Ці винагороди можуть бути надані після отримання гравцем певного призового місця

або ж, до прикладу, великої кількості награних годин.

1.4. Технічні особливості та типи веб-застосунків для Minecraft серверів

Веб-застосунок для підтримки серверу Minecraft може використовуватись для публічного просування серверу, створення унікальних можливостей для гравців або безпечної обробки платежів для оплати внутрішньоігрових товарів. В залежності від цілей створення веб-застосунку можуть використовуватись різні методи реалізації поставлених задач. Існують насупні типи веб-застосунків для підтримки Minecraft серверів.

Веб-форум – використовується для комунікації гравців між собою та зв’язку з адміністрацією Minecraft серверу. Є досить застарілим типом, оскільки на сьогоднішній день більшість комунікації відбувається на спеціалізованих Discord серверах.

Landing-page – використовується виключно для просування серверу, і часто не несе жодного технічного навантаження, окрім заохочення людей приєднатися до проєкту, таким чином зручно рекламувати сервер, лишаючи єдине посилання на сайт з вичерпною інформацією.

Веб-магазин – використовується для купівлі гравцями внутрішньоігрових предметів. Є одним з найскадніших в реалізації, оскільки потребує взаємодії з платіжними системами, що часто є великою проблемою для розробників та користувачів через складність в додаванні нових товарів та необхідності створення великої кількості мерчантів.

За приклад буде взято веб-застосунок Minecraft серверу “Craftoriya”, над котрим було проведено роботу під час дослідження. У даному випадку комбінуються особливості Landing-page та веб-магазину, оскільки веб-застосунок не має великого функціоналу взаємодії з Minecraft сервером, проте має можливість купівлі внутрішньоігрової валюти “CraftoriyaCoins”. У якості мови написання backend частини веб-застосунку було обрано JavaScript з використанням Node.js. Першочергово перевага була надана цій мові програмування через наявність

великої кількості документації FondyApi. В свою чергу, платіжна система Fondy була обрана через простоту оформлення Мерчанту і відсутність обов'язкової вимоги щодо оформлення ФОП.

Першочерговою особливістю є обов'язкова ітнеграція API, за допомогою цієї технології можлива реалізація системи оплати через платіжну систему та передача даних на серверну частину для обробки інформації за допомогою коду та конвертацію реальної валюти і ігрову донат-валюту. З серверного боку має бути написано плагін, який буде працювати з API. Оскільки на сайті є оплата, важливо подбати про безпеку. Мова йде про SSL сертифікат, котрий має бути обов'язково. Також наявність SSL сертифікату - базова вимога будь-якої платіжної системи, в даному випадку це Fondy, без сертифікату сайт не пройде валідацію. Задля успішного проходження валідації необхідно скласти договір публічної оферти та розмістити його на сайті. Важливо відстежувати транзакції аби уникнути проблем в майбутньому. Так, відстежування буде відбуватись як в особистому кабінеті Fondy так і по записам в базу даних. На випадок збою необхідно розмістити на головній сторінці посилання на гарячий канал підтримки, аби можна було зв'язатись з адміністрацією сервера та повідомити про збій. Програмно необхідно реалізувати захист від хибних платежів шляхом налаштування перевірок. Веб-застосунок має бути простим та зрозумілим. Його головне завдання – надати можливість поповнення балансу гравця та дізнаватись інформацію, варто уникати перевантаженого інтерфейсу, зробити його оптимізованим. Веб-застосунок має бути інтегрованим з базою даних задля збереження інформації про гравців, додання відповідних записів щодо балансу.

1.5. Постановка задач дипломної роботи

Після проведеного аналізу Minecraft серверів, веб-застосунків для їх підтримки, вподобань та побажань гравців було виділено основні задачі бакалаврської роботи:

1. Аналіз ринку Minecraft серверів.

2. Дослідження веб-застосунків для підтримки Minecraft серверів.
3. Аналіз вимог та вподобань гравців.
4. Розробка архітектури веб-застосунку.
5. Розробка системи оплати.
6. Розробка архітектури бази даних.
7. Розробка архітектури клієнтської частини.
8. Розробка особистого кабінету (профілю) гравця.
9. Налаштування та запуск системи.

2 РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ MINECRAFT СЕРВЕРА

2.1 Розробка та моделювання вимог до системи

На основі проведеного аналізу було сформовано низженаведені функціональні вимоги.

Функціональні вимоги до веб-застосунку:

- прийом платежів;
- перевірка стану платежів;
- перевірка нікнейму гравця;
- наявність SSL сертифікату;
- наявність договору публічної оферти;
- перелік доступних режимів гри на сервері;
- засоби зворотнього зв’язку.

Функціональні вимоги до клієнтської частини:

- відображення нікнейму гравця;
- можливість перейти у веб-застосунок для поповнення балансу;
- функціональний внутрішньоігровий магазин;
- можливість переглядати баланс;
- можливість прив’язати та переглядати внутрішньоігрові винагороди

Було сформовано наступні нефункціональні вимоги:

- легкість використання системи (для розробника);
- зрозумілість веб-застосунку та профіля (для користувача);
- надійність;
- швидкість;
- масштабованість;
- автоматизованість;
- зручна підтримка.

Для зручності розробки та формування загального уявлення про систему було створено діаграму використання системи (use case diagram). Ця діаграма визначає поведінку системи та взаємодію користувачів з нею.

В системі є три актори – це гравець, адміністратор та власник, де відповідно до даної системи варіанти використання обирає сам гравець. В свою чергу адміністратор та власник системи потрібні для її підтримки, що зображено окремо. Діаграма прецедентів користувача Гравець зображена на (рис 2.1).

Рисунок 2.1 – Діаграма варіантів використання актора Гравець

Основними діями, що доступні для гравця є:

- можливість зареєструватись за відсутності ліцензійної копії гри;
- можливість авторизуватись, якщо гравець вже зареєстрований (у разі наявності ліцензійної копії гри, всі дії відбуваються автоматично);
- обрати певний режим гра та долучитись до нього, це може бути як

класичний режим так і міні-гра. на діаграмі вони зображені окремо, адже в міні-ігор є власний хаб;

- використання магазину для покупки платного контенту;
- перегляд досягнень, винагород та їх редагування;
- перевіряти баланс свого профілю.

В свою чергу основними діями для адміністратора (рис 2.2) є:

- редагування міні ігор за побажаннями гравців;
- видавати попередження за порушення правил;
- блокування гравців за порушення правил;
- запуск спеціальних івентів (подій).

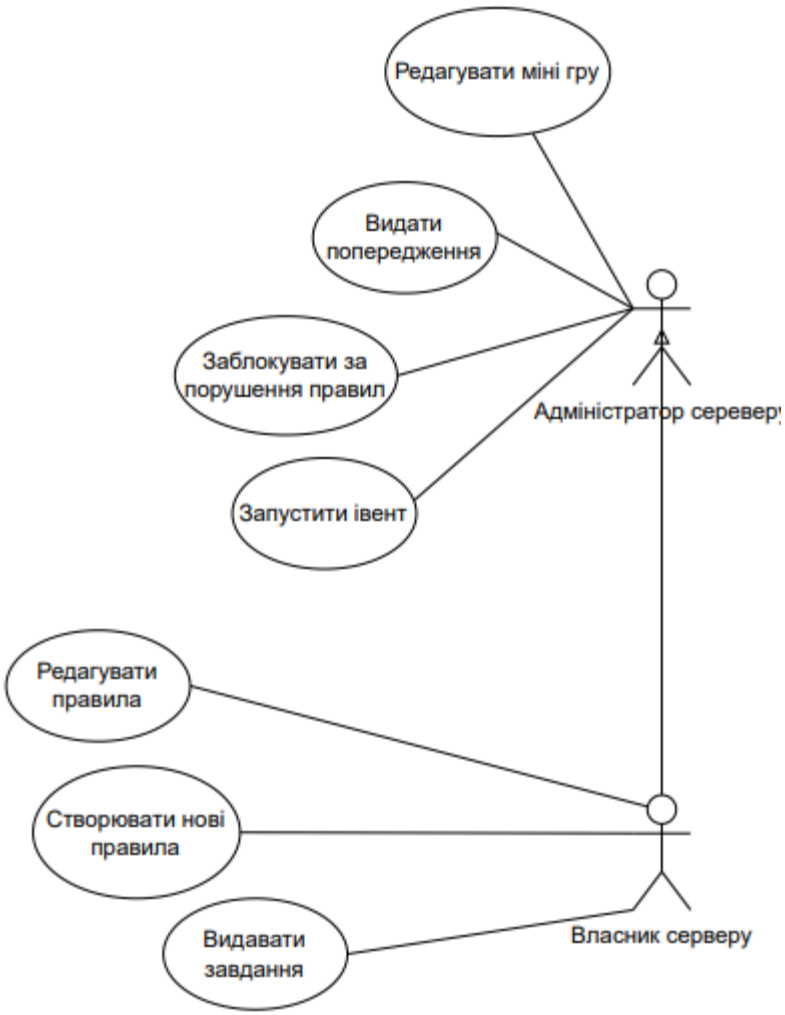


Рисунок 2.2 - Діаграма варіантів використання акторів, Адміністратор серверу та Власник серверу

Власник має всі ті ж можливості, що і гравець та адміністратор та додатково власні:

- можливість редагувати правила;
- створювати нові правила;
- видавати завдання.

2.2 Робробка структури бази даних

Було розроблено структуру БД на основі діаграми структури даних (DSD), вона зображена на (рис.2.3). Це діаграма концептуальної моделі даних, яка документує сутності та їх зв’язки, а також певні обмеження. В конкретному випадку в Info зберігається основна спільна інформація про гравців. В player_stats – статистика кожного гравця. player_stats може бути декілька під кожен сервер окремо, адже на кожному сервері гравці мають окрему статистику. В player_passes зберігається інформація щодо придбаних гравцями перепусток на кожен окремий платний сервер. event_leveling – зберігає інформацію про кількість пройдених гравцем івентів на певному сервері. На кожному івенти можуть убти окремими і під кожен такий сервер важливо зберігати дані окремо для зручності. Таким чином, від кількості пройдених івентів буде залежати рівень винагороди, що отримує гравець в кінці сезону. Відповідно player_medals – це інформація про медалі гравців, адже медалі - це фактично система винагород, яку було вирішено зробити окремо та розвивати як незалежну механіку. Окремо, в таблиці medals зберігається інформація про кожну доступну винагороду на сервері, в подальшому ця інформація використовується для ідентифікації медалі та її присвоєння гравцю. Таблиця name_color використовується як додаток до магазину, що відповідає за платний контент, а саме – можливість придбання кольорових нікнеймів.

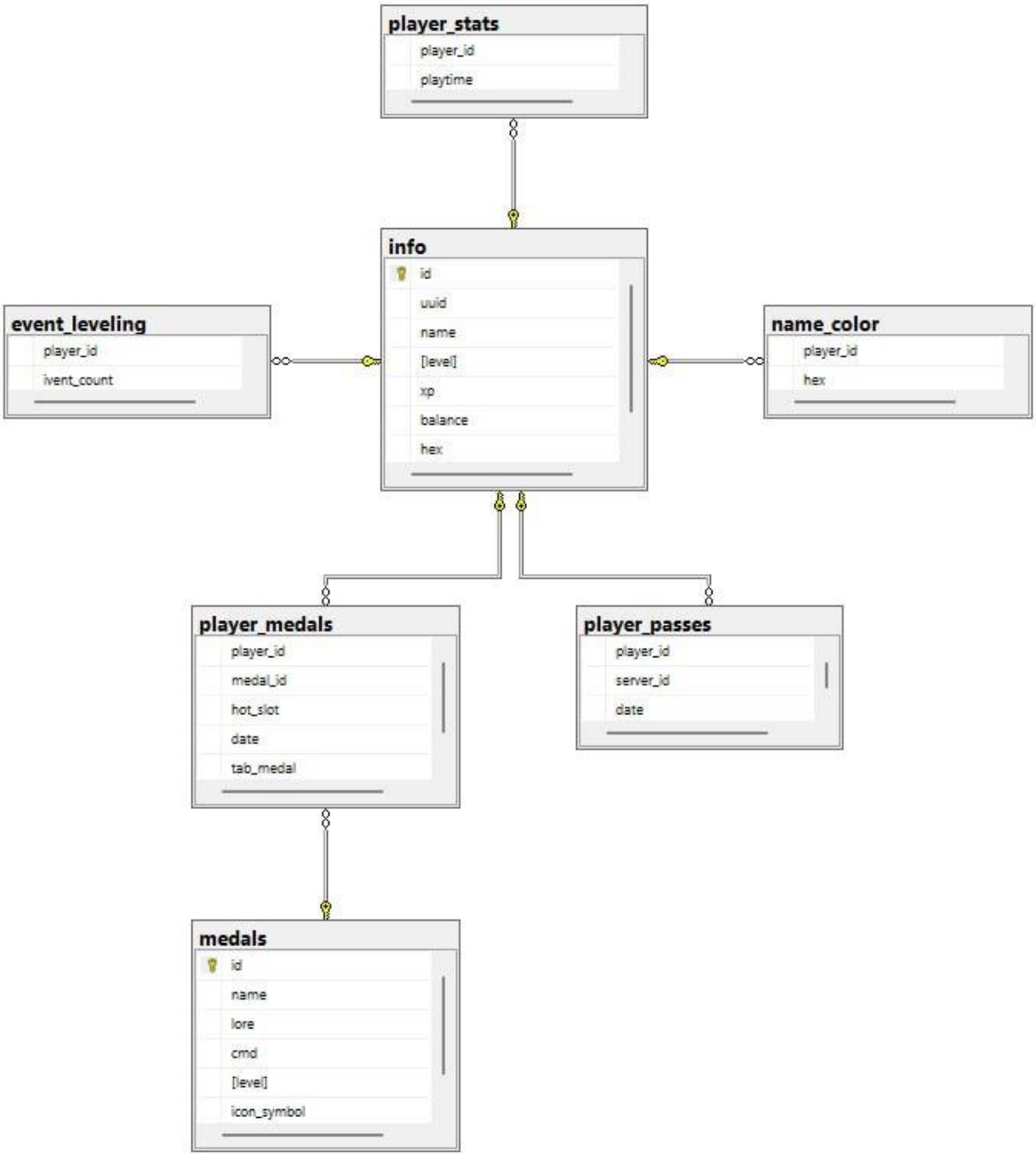


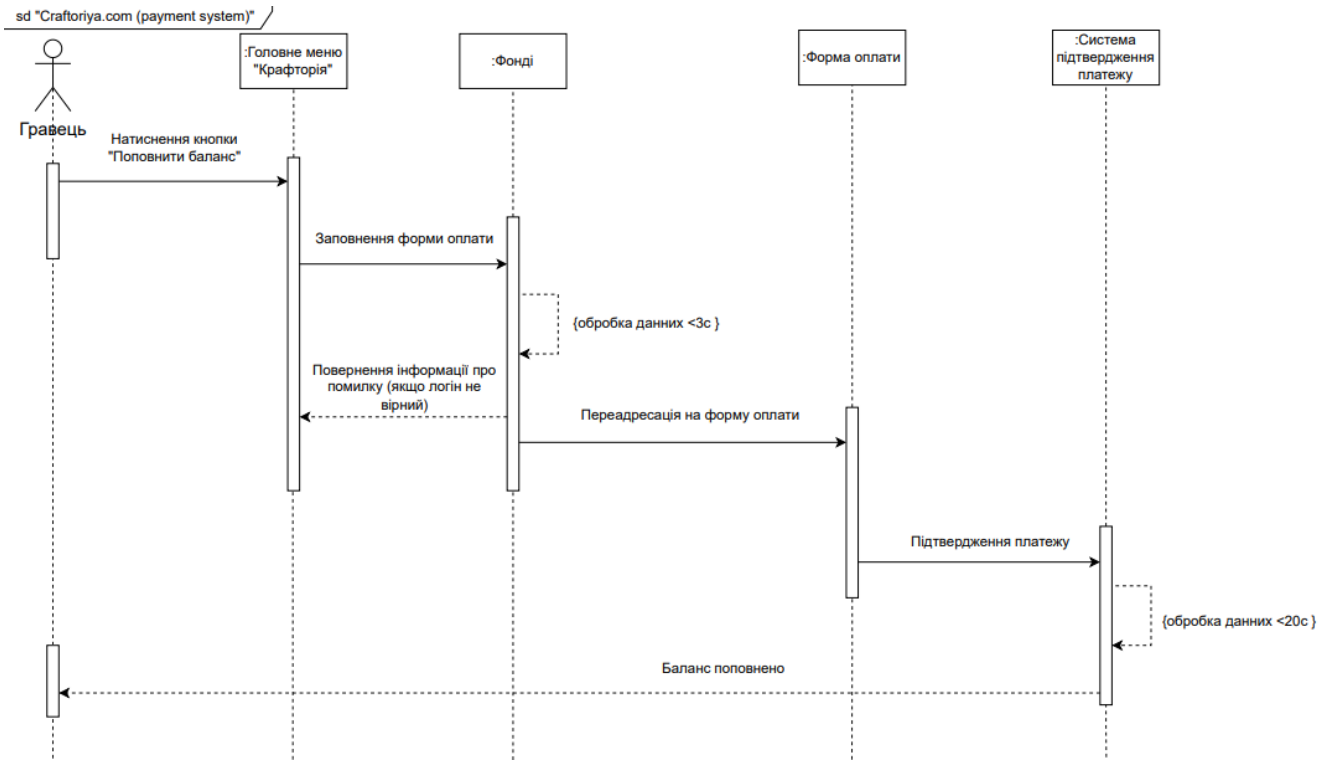
Рисунок 2.3 - Діаграма БД

2.3 Розробка алгоритмів роботи основних функцій та архітектури системи

Було розроблено алгоритми роботи основних функцій, що описуються діаграмою послідовності та діаграмою діяльності (рис 2.4) та (рис. 2.5) відповідно. Конкретно ці діаграми відображають роботу безпосередньо веб-застосунку, а саме

те, як працює система оплати.

Спочатку гравець натискає на кнопку «поповнити баланс», що знаходиться на головній сторінці, після чого заповнює форму оплати, де необхідно вказати ігровий нікнейм для розуміння системою, якому саме гравцю необхідно нарахувати кошти. Також гравець лишає пошту для зворотнього зв’язку і бажану суму до поповнення. Після цього інформація обробляється та надсилається форма оплати від фонді, у разі неправильно вказаного нікнейму (якщо він не заходив на сервер принаймі один раз), сервіс видасть помилку. В формі оплати необхідно вказати дані банківської карти та пошту для надчилання системою чеку про оплату. Після чого відбувається підтвердження в банківському застосунку, дані надсилаються на сервер, обробляються та баланс гравця поповнюється на вказану ним суму.



Рискнок 2.4 - Діаграма послідовності

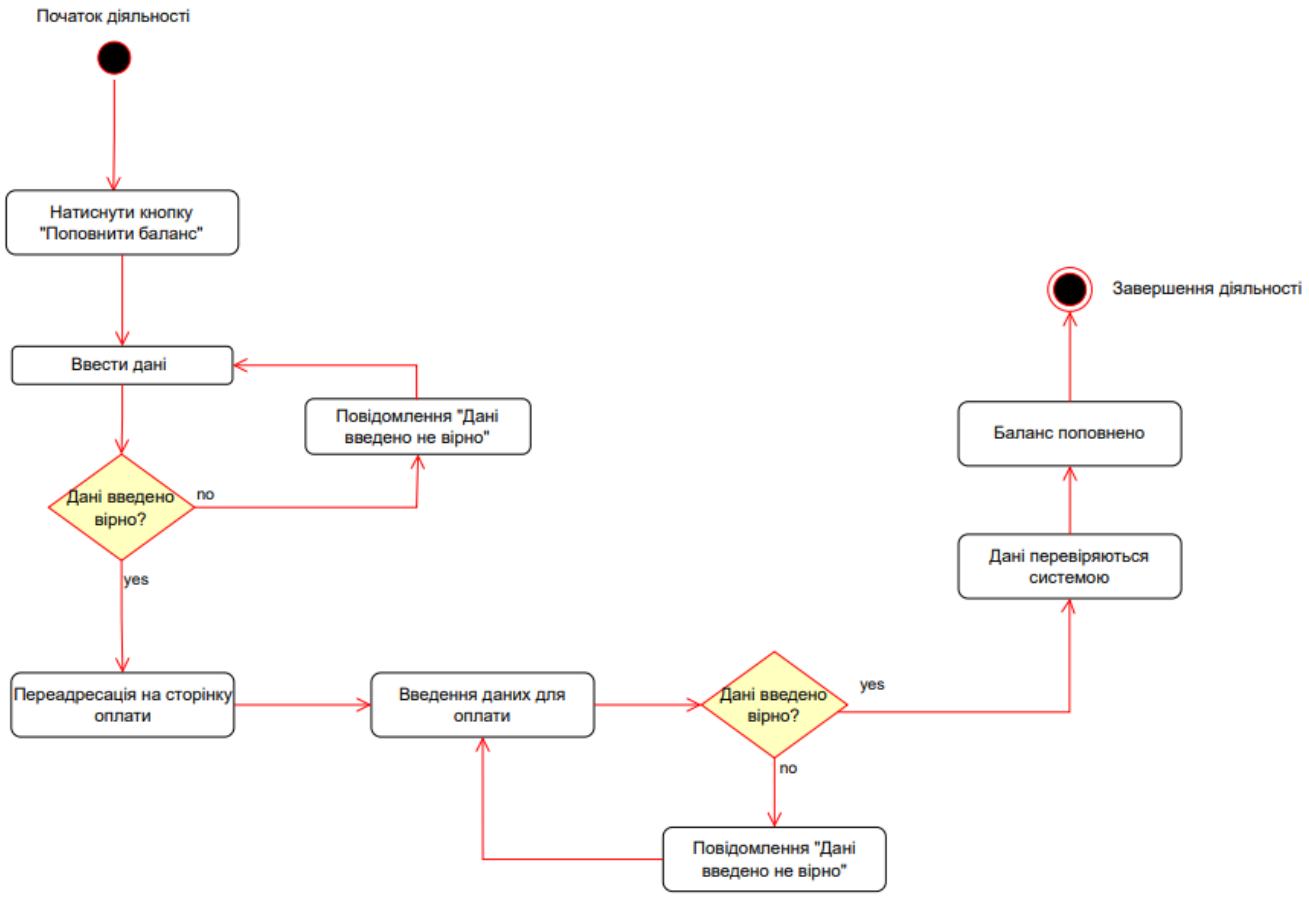


Рисунок 2.5 - Діаграма діяльності

Архітектура системи представлена на рисунку 2.6 і демонструє основні компоненти, необхідні для її функціонування, та їх зв’язок.

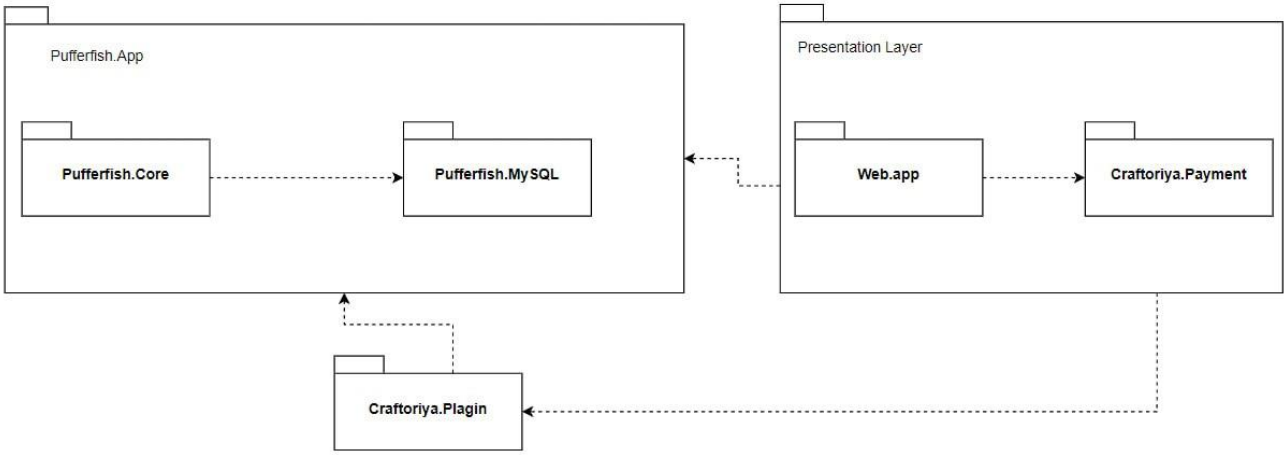


Рисунок 2.6 – Архітектура системи

2.4 Обгрунтування вибору технологій для розробки програмного забезпечення

2.4.1 Java та Kotlin

Java є однією з основних мов програмування в корпоративному середовищі, де вона використовується для розробки веб-застосунків, баз даних, електронної комерції та інших систем. Java є мовою з високим рівнем абстракції, що дозволяє програмістам швидко та ефективно розробляти програмне забезпечення. Java має велику кількість стандартних бібліотек, що значно полегшує розробку програмного забезпечення, та переносимість, завдяки якій програмне забезпечення, написане на Java, може працювати на різних платформах без змін. Крім того, Java є мовою програмування з відкритим кодом, що дозволяє програмістам використовувати безліч готових бібліотек та фреймворків для розробки своїх проектів.

Kotlin, з іншого боку, була розроблена з метою покращення розробки програмного забезпечення для платформи Java, зокрема для Android. Kotlin може бути використана для розробки мобільних застосунків, веб-застосунків та серверних програм. Kotlin також має свої переваги. Ця мова програмування має простий синтаксис, що дозволяє програмістам швидко розробляти код. Kotlin також має безліч корисних функцій, таких як null-безпека, функціональна програмування та інтероперабельність з Java. Більше того, Kotlin має безкоштовний, відкритий код та хорошу підтримку у середовищах розробки, та в спільноті розробників.

Безпосередньо для розробки було опрано ці дві мови тому, що сам Minecraft написаний на Java, відповідно правильно та зручно буде працювати саме з цією мовою, на ній написано більшість плагінів. Kotlin є альтернативою, адже плагіни, що написані на Kotlin, сумісні з Java плагінами. Використання двох мов дозволить при подальшому масштабуванні сервера залучити до проєкту більше розробників.

2.4.2 HTML,CSS, JavaScript

HTML (HyperText Markup Language) - це мова розмітки, яка

використовується для створення веб-сторінок. HTML не є мовою програмування, а використовується для розміщення елементів інтерфейсу користувача на вебсайті. Основними елементами розмітки в HTML є теги. Окрім тегів, HTML використовує атрибути, які вказують додаткову інформацію про елемент.

CSS (Cascading Style Sheets) - це мова стилів, яка використовується для визначення вигляду та форматування елементів веб-сторінок. CSS використовується разом з HTML, щоб забезпечити відображення вмісту сторінки у браузері. CSS дозволяє визначати вигляд елементів на сторінці, редагувати їх кольори, шрифти, фонові зображення, розміри та відстані між ними і багато іншого. Це дозволяє розробникам створювати гарний і привітний користувацький інтерфейс. CSS складається з правил, в свою чергу правила складаються з селекторів та стилів. Селектори вказують, які правила застосовуються до конкретних елементів на веб-сторінці а стилі задають вигляд елементів на сторінці.

JavaScript – це скриптова, об’єктно-орієнтована мова програмування що може використовуватись для розробки веб-застосунків як на серверній так і на клієнтській частині. На сьогоднішній день JavaScript є сучасною і однією з найпопулярніших мов програмування. Хоч JavaScript і має ознаки ООП, але має деякі відмінності від інших об’єктно-орієнтованих мов через наявність концепції наслідування прототипів. Не дивлячись на те що JavaScript має С-подібний синтаксис, існують такі корінні відмінності:

- наявність інтероспекції – інтероспекція дає можливість аналізувати функції і використовується для аналізу коду, механізм прототипів;
- механізм прототипів – завдяки механізму прототипів об’єкти JavaScript можуть наслідувати властивості один одного;
- присутня обробка винятків – при виникненні помилки або іншого винятку, відбувається передача подальшого виконання програми визначеному обробнику винятка;
- наявність анонімних та лямбда функцій – анонімні функції не мають назви або інших індикаторів, лямбда функція є анонімною, але має кардинально інший синтаксис ;

- функції є об’єктами першого класу – існує можливість передачі функцій у якості аргументів іншим функціям а також запису їх до змінних;

Через велику кількість бібліотек та фреймворків, дана мова використовується в багатьох сферах розробки застосунків. Такі бібліотеки як React, Angular або Vue.js дозволяють розробляти прогресивні веб-застосунки. В свою чергу Node.js дозволяє використовувати JavaScript на боці серверу, та спрощує написання коду, через відсутність потреби використання різних мов на front та back частині веб-застосунку. Також варто зазначити такі фреймворки, як React Native, що дозволяє створювати мобільні застосунки з використанням JavaScript, та Electron, що дозволяє створювати стаціонарні застосунки для настільних ПК. [2]

2.4.3 Технологія API для взаємодії веб-застосунку з ігровою частиною Minecraft сервера

З’єднання веб-застосунку з серверною частиною в додатку реалізовано за допомогою технології API. API спрощує процес програмування завдяки тому, що надає тільки об’єкти або ж дії, котрі потрібні розробнику для реалізації визначених функцій. Так як для користувача кнопка, скажімо, оплати, поповнює баланс його персонажа в грі без потреби знати, які процеси там відбуваються, так для розробника є, фактично, лише функція, по переміщення файлів з точки А в точку Б без необхідності замислюватись над тим, що відбувається між надсиланням та прийомом того ж файлу. Фактично API використовує запити та відповіді.

Архітектуру API зазвичай пояснюють з точки зору клієнта та сервера. Застосунок, що надсилає запит є клієнтом, а застосунок, що надсилає відповідь – сервером. Існує 4 варіанти взаємодії з API.

SOAP API – протокол доступу до об’єкта. Тут клієнт з сервером обмінюються інформацією за допомогою XML. На разі ця технологія застаріла.

RPC API – система віддаленого виклику. Клієнт виконує функцію на сервері а той в свою чергу надсилає клієнту результат.

Websocket API – сучасна розробка, що використовує об’єкти JSON для

передачі даних. Підтримує зворотній зв'язок між клієнтським застосунком та сервером. Сервер може надсилати повідомлення зворотнього виклику, що робить його дуже ефективним. Кращий за REST API.

REST API – на сьогодні це найбільш зручні та гнучкі API-інтерфейси. Клієнт може надсилати запити на сервер у вигляді даних. Сервер використовуватиме їх для запуску внутрішніх функцій та повертатиме дані клієнту. На рис. 2.7 відображено баланс гравця в його профілі. [7]

Рисунок 2.7 - Баланс гравця в профілі

Наступний вигляд має база даних, котра містить інформацію передану за допомогою API (рис. 2.8). В конкретному випадку колонка balance. Там можна бачити кількість реальних гривень, що гравець задонатив на сервер. Оскільки в даному випадку 1 гривня = 1 ігровий коїн, то число 55 і є реальним балансом гравця, за котрий він може придбати перепустку з безкоштовного на приватний сервер, косметику абощо.

```
MariaDB [playersinfo]> select * from info where name = "Kvadratnyk";
+----+--------------------------------------+------------+-------+------+---------+--------+
| id | uuid                                 | name       | level | xp   | balance | hex    |
+----+--------------------------------------+------------+-------+------+---------+--------+
|  8 | 1b334ba3-1e1f-43a1-bd97-e6a1b628d20a | Kvadratnyk |     4 |    0 |   55.00 | 33F9FF |
+----+--------------------------------------+------------+-------+------+---------+--------+
1 row in set (0.002 sec)
```

Рисунок 2.8 - Баланс гравця в базі

Далі йде демонстрація безпосередньо підключення системи. Таким чином лишається налаштувати прийом callback від сайту та повернути з використанням

коду. Після чого треба налаштувати перевірки.

Технологія API необхідна для прив’язки веб-застосунку до гри та можливості автоматичного нарахування, в даному випадку, коштів на внутрішньоігровий профіль гравця. Таким чином за допомогою API відбувається запис до бази даних та нарахування коштів гравцю. Система є перспективною, адже є універсальною, її можна використовувати з будь-яким сервером чи платіжною системою. [13]

2.4.4 Вибір платіжної системи для здійснення поповнення особистого балансу гравцями Minecraft сервера

При виборі системи для здійснення оплати при поповнені внутрішньоігрового балансу, було розглянуто багато доступних варіантів, таких як NovaPay, Portmone, LiqPay та Fondy. Більшість з доступних варіантів потребували реєстації ФОП в державному реєстрі підприємців, що було недоцільно, враховуючи майже нульові прибутки серверу на момент запуску. Перевагу було надано платіжному сервісу Fondy. Вирішальними факторами стали відсутність необхідності реєстрації ФОП. Додатковим плюсом була власна API з простою документацією для Node.JS, що дозволило пришвидшити процес розробки та підключення системи оплати до веб-застосунку.

2.4.5 SKRIPT та особливості написання плагінів

Skript – це інтерпретована мова програмування, створена саме для простої роботи та швидкої розробки плагінів Minecraft, котрі можна перезавантажувати без повного перезапуску самого серверу. Він дозволяє швидко вносити зміни та швидко розробляти з нуля досить складні механіки. Це повноцінна, хоча й спрощена мова програмування. В ній можна так само працювати з базами данних, створювати методи, викликати їх та виконувати математичні операції. Він надзвичайно зручний ще тому, що в нього є величезна кількість аддонів, а також велике ком’юніті, та документація. За допомогою SKRIPT було реалізовані деякі меню та проведена робота з базою данних і данними гравців.

Фактично цей плагін дозволяв писати плагіни так само як і в окремому

програмному забезпеченні, такому як Intellij IDEA, але спростивши цю процедуру. Незважаючи на це, він не обмежував розробників. Розробники могли працювати через нього з базами данних, пакетами, чанками, та навіть напряму виконувати всередині java код. Все це було модульно, зручно в розробці і збільшувало продуктивність. [3]

2.4.6 Вибір ядра для клієнтської частини

З сучасних та зручних готових інструментів для створення безпосередньо сервера майнкрафт є кілька варіантів. Більшість з них базуються на коді серверного ядра Bukkit. Ядро Bukkit на поточний момент застаріло як морально, так і з технічної сторони. Воно дозволяло просто та зручно запустити власний сервер використовуючи .bat чи .sh скрипти. Також воно надавало досить просунутий для свого часу API для розробки плагінів під цю платформу на такій мові як Java. Ядро було досить легко конфігурувати, проте було малопродуктивним. Проблема більшості ядер серверів майнкрафт, котра й досі не зникла повністю — це обробка інформації в одному потоці, що через великі та складні в обчисленні задачі заморожувало весь основний потік серверу на час свого обчислення. Також конкретно Bukkit не мав готових інструментів для роботи з яким би то не було проксі. Його можна було використовувати як юніт в проксі мережі, але це було досить незручно, небезпечно, та слабко піддавалось налаштуванню.

Наступним ядром, котре також досі існує є Spigot. Spigot заснований на коді Bukkit, та вперше зайнявся оптимізацією роботи серверу. На цьому ядрі набагато швидше реалізована обробка ігрових npc, та різних механік світу. Ще тут є більш обширна конфігурація, що дозволяє налаштувати ядро конкретно під вашу задачу, щоб зменшити споживання ресурсів CPU, та RAM. З технологій тут з`явилась підтримка роботи на пряму з проксі ядром BungeeCord, котрий дозволяв пакетно з'єднувати сервери в один, переспрямовуючи пакети гравця на потрібний нам сервер. [4]

Через деякий час, приблизно з версії 1.8 з’явилось, та почало стрімко набирати популярності ядро Paper. Разом з Paper зароджувалось дуже багато так

званих “форків” попередніх 2х ядер, але вони не здобули такої популярності, та не були достатньо якісними. Розробники ж Paper’а почали активно доповнювати API ядра, а також дуже добре попрацювати над оптимізацією, та новими технологіями. Був повністю переписаний двигун обробки світла, та поведінки червоного каменю (складний для обчислення елемент в грі). Завантаження нових чанків отримало частково багатопоточну реалізацію, з’явилась буквально найбільша конфігурація серед всіх форків, а також безпечний та швидкий проксі Velocity, з котрим мав змогу працювати лише Paper, та форки, що були написані на його основі. Так, як це відкрите програмне забезпечення — то на його основі виходило дуже багато ядер зі своїми змінами, котрі не рідко в результаті наслідувались самим Paper’ом. [5]

Одним з найкращих, та найстабільніших його форків стало ядро Pufferfish. Pufferfish ще швидший та має додаткові налаштування. Його головна перевага – швидкість роботи та функціонал. Paper, та його форки повністю зворотньо сумісні з плагінами під Bukkit, але через часту зміну деяких методів, видалення застарілих, та додавання нових - старі плагіни з часом переставали працювати на нових версіях (в залежності від своєї складності), а плагіни, що були написані під нові версії з використанням нових методів, класів чи інших нововведень Paper’а – були повністю несумісні з старішими версіями. Також проблем додавало написання плагіна для кожної найменшої зміни механік гри та повне перевстановлення для внесення змін у вже існуючий плагін.

В таблиці 2.1 наведено порівняння ядер за їх основними параметрами.

Таблиця 2.1 – Порівняння ядер за основними їх параметрами

Характеристики	**Bukkit**	**Spigot**	**Paper**	**Pufferfish**
Розповсюдженість	+	+	+	-
Структуризація	-	+	+	+
Зручність	-	-	+	+
Робота з проксі	-	+	+	+
Робота з швидким проксі	-	-	+	+
Часткова багатопоточна обробка	-	-	+	+
Оптимізація обробки світла	-	-	+	+
Оптимізація редстоуну	-	-	+	+
Додаткові виправлення	-	-	-	+

2.5 Вибір середовища для розробки

2.5.1. SublimeText

Sublime Text – це текстовий редактор створений за допомогою мови програмування Python. Він є одним з найпопулярніших інструментів для розробки, оскільки може підсвічувати синтаксис більш ніж 50 мов програмування, включаючи JavaScript, HTML та CSS. Його особливостями є швидкість роботи, кросплатформеність та простота інтерфейсу користувача. Оскільки Sublime Text є простим текстовим редактором, він не має багатьох функцій які пропонують інші редактори коду, такі як Visual Studio Code або IntelliJ IDEA, одим з цих мінусів є майже повна відсутність розширень та плагінів через те що Sublime Text не є проєктом з відкритим вихідним кодом. При написанні fronend частини веб-застосунку, перевагу було надано саме Sublime Text, через швидкодію та відсутність потреби відладки коду у випадку написання fronend. [8]

2.5.2 IntelliJ IDEA

IntelliJ IDEA - це надзвичайно потужна інтегрована середовища розробки (IDE) для мов програмування Java та Kotlin. Ця програма дозволяє розробникам швидко та ефективно створювати високоякісний код. Одна з головних причин використання IntelliJ IDEA полягає в тому, що вона має безліч корисних функцій,

які полегшують життя програмістів. Наприклад, вона надає готові шаблони та бібліотеки, що значно економить час під час розробки проектів. Також IntelliJ IDEA має багато функцій для автоматизації рутинної роботи, які дозволяють програмістам швидше та якісніше розробляти проєкти.

Окрім того, IntelliJ IDEA має безліч зручностей для розробників мови програмування Java та Kotlin. Ці зручності дозволяють розробникам легко орієнтуватися в коді, швидко знаходити помилки та діагностичні повідомлення, а також вбудований дебагер, що допомагає знаходити та виправляти помилки.

Ще в IntelliJ IDEA є простий спосіб під’єднання до системи контролю версій, такої як github. Це дозволяє розробляти великий код зручно великою кількістю людей одночасно. Також велика кількість плагінів, та, наявність режиму колаборації, що впливає на вибір програмістів в сторону цього IDE.

Загалом, IntelliJ IDEA є однією з найкращих IDE для розробки програм на Java та Kotlin, завдяки її потужним функціям, зручному інтерфейсу та багатофункціональністю. Вона допомагає програмістам розробляти якісний та ефективний код та економити час при розробці проєктів. [9]

2.6 Розробка веб-застосунку для підтримки Minecraft сервера

2.6.1 Написання Frontend частини веб-застосунку

Frontend сайту - це те, що першим справляє враження про продукт у нового користувача, саме тому надзвичайно важливо приділити цій частині розробки якомога більше часу. Перше - це база з HTML + CSS. Ними пишеться основна структура, і навіть деякі інтерактивні елементи. На першому кроці CSS файл під’єднується в документ. На основній сторінці сайту (самих сторінок декілька) реалізована кнопка з адресою серверу. В розробці фронту сайту також необхідне використання JS, для створення анімацій, переходів та інтерактиву сторінки, котрий не буде ніяк пов’язаний з логікою backend частини. При натисканні на кнопку відбувається запис адреси серверу в буфер обміну користувача використовуючи спеціальний метод .clipboard в JS. В змінну записується поточний

час для подальших обчислень. При самому натисканні перевіряється, чи пройшло з моменту останнього натиску більше трьох секунд віднімаючи від поточного часу час, який записаний в змінній в мілісекундах. Якщо ж так - то міняється вмст кнопки на “Скопійовано!”, та все повертається до початкового виду, як було за 3000 мілісекунд. При появі будь-якої помилки при спробі копіювання буде видано помилку, що повідомить користувача.

Список контейнери, що містять в собі всі актуальні під-сервера проєкту, наведено в лістингу програмного коду (Додаток Б). Тут гравець може швидко ознайомитись з основною потрібною інформацією по цим серверам, та побачити невеличку ілюстрацію того, що на ньому відбуватиметься.

2.6.2 Написання Backend частини веб-застосунку

Метою Backend частини веб-застосунку є встановлення додатку, який буде працювати на веб-сервері, обробляти запити користувачів та зберігати/зчитувати дані з баз даних.

Спочатку відбувається підключення необхідних залежностей (express, mysql, cookie-parser, fs, path, sha1, CloudIpsp, https). express - це фреймворк для створення веб-додатків на Node.js; mysql - модуль для роботи з MySQL-базами даних; cookie-parser - модуль для роботи з кукісами (cookies) користувачів; fs - модуль для роботи з файловою системою; path - модуль для роботи зі шляхами до файлів; sha1 - модуль для хешування даних за алгоритмом SHA-1; CloudIpsp - SDK для інтеграції платіжної системи FONDY; https - модуль для роботи з HTTPS-з'єднаннями. Далі створюється екземпляр фреймворку express та змінні merchant_id та secretKey для інтеграції з платіжною системою FONDY. Після цього функція filedir(filename) отримує назву файлу та повертає його вміст, зчитуючи його з файлової системи та перетворюючи його в рядок.

Наступним кроком є конфігурація для підключення до баз даних playersinfo та donates. Параметри конфігурацій включають адресу хоста, ім'я користувача, пароль, назву бази даних та ліміт підключень. Далі використовується модуль mysql для створення пулів з'єднань для кожної бази даних. Крім того, є обробники

помилок підключення до баз даних playersinfo та donates відповідно. Вони виводять повідомлення про помилку, якщо підключення не вдалося.

Для функціонування додатку використовується підключення модулів: express, mysql, cookie-parser, fs, path та cloudipsp-node-js-sdk. [10]

Об'єкт globalvars зберіає глобальні змінні, такі як статус електронної пошти, сума, гравець, запит на SQL, SQL 2, статус промокоду та глобальне значення cookie. Об'єкт tempDataStore використовується для зберігання тимчасових даних. Методи app.use() застосовуються для підключення middleware, таких як cookieParser(), express.static(), express.json(). Також використовується метод app.get() для обробки GET запитів на різні шляхи, наприклад, кореневий шлях /, /payment, /coins, /oferta та /rules. Після цього методу використовуються виклики функцій, що відправляють відповідні файли клієнту.

Обробник app.post() застосовується для запиту на перевірку наявності нікнейму в базі даних. Використовуючи бібліотеку mysql, виконується запит до бази даних на отримання даних про гравця з вказаним нікнеймом. Результат запиту повертається клієнту у відповіді на POST запит.

2.6.3 Написання плагінів для роботи системи оплати

Меню в Minecraft працює через відкривання гравцю скрині з потрібними предметами в конкретних комірках, а після, підв’язування їх під конкретні дії на стороні плагіна. Так само працює система оплати.

При натисканні гравцем на предмет його голови у гарячих клавішах, виконується перевірка, чи не пройшла перезарядка. Якщо ні, то програється потрібний звук та меню не відкривається. Якщо так, то програється ішний звук, це означає що все пройшло успішно та для гравця виконується функція яка відкриває йому скриню з потрібно розташованими предметами (рис. 2.9).

Рисунок 2.9 – Демонстрація магазину в профілі

Далі, при натисканні на потрібну комірку, гравцю відкривається інша скриня, яка вже і відповідає за оплату. Коли гравець знаходиться в меню “Магазин”, він може обрати що він хоче придбати. Наприклад, це можуть бути різні косметичні предмети, певні бонуси та можливість потрапити на закриті сервери, які відкриваються лише гравцям що придбали перепуску (рис 2.10).

Рисунок 2.10 – Демонстрація вмісту магазину

Якщо гравець обирає предмет “Придбати перепустку” (в даному випадку – це перша кнопка з білим папірцем), система перевіряє, чи не придбав він вже перепустку, щоб не списувати оплату двічі.

Якщо у гравця немає перепустки, система через запит до бази даних визначає кількість валюти гравця та, якщо вона більше або дорівнює ціні перепустки, списує цю саму суму, та гравець після підтвердження оплати отримує доступ до потрібного серверу.

Якщо у гравця не вистачає валюти для купівлі перепустки, йому пропонується її придбати на веб-сайті, щоб в подальшому використати її в придбанні перепустки, певних бонусів, або косметичних предметів.

2.7 Реалізація серверної частини та встановлення веб застосунку на хостинг

2.7.1 Порядок встановлення веб застосунку на сервер

Основні кроки для встановлення веб застосунку на сервер для подальшого його запуску в загальних формах наведено нижче.

1. Встановлюється серверне ПЗ. Якщо немає серверного програмного забезпечення, то спочатку потрібно встановити його на сервер. Деякі популярні серверні ПЗ включають в себе Apache, Nginx, IIS тощо. В залежності від обраного ПЗ, процедура встановлення може відрізнятись.

2. Встановлення бази даних. Якщо застосунок використовує базу даних, то потрібно встановити сервер баз них, наприклад, MySQL, PostgreSQL, MongoDB тощо. Це дозволить зберігати та організовувати дані, необхідні для роботи застосунку.

3. Налаштування файрволу. Файрвол - це програмне забезпечення, що контролює доступ до сервера через Інтернет. Далі необхідно налаштувати файрвол так, щоб забезпечити доступ до веб-сайту, але при цьому зберігти його безпеку.

4. Завантаження коду застосунку на сервер. Потрібно використовувати протокол SSH або FTP для завантаження файлів.

5. Налаштування веб-серверу. Це треба зробити так, щоб веб-сервер відповідав на запити користувачів та відображав веб-сайт. Це можна зробити, наприклад, за допомогою файлу конфігурації Apache або Nginx.

6. Перевірка роботи застосунку.

Щодо вибору хостингу, найбільш вдалим варіантом є Hetzner через їхню відмінну репутацію в галузі та високу якість послуг. Hetzner є одним з провідних хостинг-провайдерів в Європі та пропонує широкий спектр послуг, включаючи віртуальні та фізичні сервери, хмарні обчислення та інші. Одна з найбільших переваг Hetzner - це їхні низькі ціни порівняно з іншими провайдерами хостингу. Крім того, Hetzner пропонує дуже широкі можливості вибору обладнання і операційної системи, що дозволяє мені налаштувати мої сервери саме так, як

потрібно в конкретному випадку. Це дуже важливо, оскільки необхідно забезпечити максимальну продуктивність веб-сайтів і заощадити гроші на необхідному обладнанні. Хоча є деякі альтернативи для Hetzner, проте їхня спеціалізація на хостингу і високій якості обслуговування роблять їх досить вдалим рішенням. Деякі інші провайдери хостингу, які можуть бути розглянуті, включають OVH, DigitalOcean, Amazon Web Services, Google Cloud Platform та інших. Хоча ці альтернативи можуть бути більш відомі та мають свої переваги, Hetzner є найкращим варіантом для вирішення поставлених завдань. Hetzner надають все необхідне, щоб забезпечити максимальну продуктивність і надійність веб-застосунку та сервера, що дозволяє спокійно працювати над прєектами та не хвилюватись про проблеми з боку хоста. В даному випадку використано тариф з процессором ryzen7 8ядер, 16 потоків, 64гб оперативної пам'яті, 2тб місця на SSD дисках, та інтернет карта 1гбіт. На хет Hetzner знері досить зручно створити, та налаштувати саму систему, та потрібні порти. Бісля базового встановлення не виникає зайвих проблем з підтвердженнями і тд. Можна приєднуватись до лінуксу по ssh, генерувати ключі, та переналаштовувати програмну частину. [6]

2.7.2 Розробка бази даних MySQL

MySQL - це одна з найпопулярніших систем управління базами даних (СУБД), що використовуються в сучасному програмуванні. Одна з головних переваг MySQL полягає в тому, що вона є безкоштовною та з відкритим вихідним кодом, що забезпечує її доступність для широкого кола користувачів та розробників. MySQL є реляційною СУБД, що означає, що дані зберігаються у вигляді таблиць зі стовпцями та рядками. Вона дозволяє користувачам зберігати, оновлювати, видаляти та вибирати дані з бази даних з використанням SQL (Structured Query Language), що є стандартною мовою запитів до СУБД. SQL дозволяє користувачам створювати складні запити, щоб знайти потрібні дані з бази даних. Одна з ключових особливостей MySQL - це її швидкодія. Вона може обробляти великі обсяги даних за дуже короткий час. Також MySQL має високу надійність, стійкість до помилок та високу масштабованість, що дозволяє

використовувати її для різноманітних проектів, від невеликих сайтів до великих підприємств. В додатку, що розроблено в даній роботі, використовується форк MySQL, MariaDB. ця СУБД дещо швидша за MySQL і повністю відкрита. Не зважаючи на це вона повністю зворотньо-сумісна з MySQL. Для роботи з MariaDB ще більш ефективно можна використовувати кешування, та так звані пули підключень. Для цього реалізований власний драйвер, що в Kotlin дозволяє звертатись до баз данних як до об'єктів, в котрих викликаються методи. В середині цей драйвер використовує Hikari pool для відкриття з'єднання. При відсутності змін, аби не діставати інформацію безпосередньо з БД, використовується саме кешування. Це прискорює швидкість доступу, розвантажує диск, та подовжує його ресурс. З мінусів – великий кеш займатиме багато оперативної пам'яті. Самі запити по замовчуванню стандартним SQL драйвером відбуваються синхронно, тобто в одному основному потоці. Це досить погано для швидкодії. Основий процесс Minecraft серверу првацює в одному потоці, і це створює проблему. Саме тому все описане вище потрібно запускати асинхронно. Завдяки бібліотеці CompletableFuture стає можливим відкрити новий пул підключень hikari, перестворити об'єктні запити до БД в SQL сирі запити, дістати чи якось модифікувати інформацію в БД, та кешувати її для перевикористання. Через конфігуруєму кількість часу пул закривається, та при потребі перевідкривається. [10]

2.7.3 Отримання SSL сертифікату

SSL-сертифікати є незамінним інструментом для захисту конфіденційності інформації в Інтернеті. Вони дозволяють шифрувати передачу даних між веб-сайтом та користувачем, забезпечуючи захист від різних видів атак, включаючи перехоплення даних та внесення змін до них. Одним із популярних постачальників SSL-сертифікатів є CloudFlare. У цій дипломній роботі розглянуто процес отримання SSL-сертифіката на платформі CloudFlare.

CloudFlare - це один з провідних постачальників послуг CDN (Content Delivery Network) та захисту веб-сайтів від DDoS-атак. Платформа пропонує

широкий спектр інструментів, що дозволяють забезпечити безпеку веб-сайту, включаючи SSL-сертифікати.

Для отримання SSL-сертифіката на платформі CloudFlare необхідно виконати наступні кроки:

- зареєструватися на платформі CloudFlare;
- додати веб-сайт до CloudFlare;
- встановити SSL-сертифікат на платформі CloudFlare;
- встановити SSL-сертифікат на сервері.

Після успішної реєстрації необхідно додати веб-сайт, на якому необхідно встановити SSL-сертифікат, до платформи CloudFlare. Для цього необхідно додати DNS-записи для веб-сайту на платформі CloudFlare.

Після додавання веб-сайту до платформи CloudFlare можна перейти до встановлення SSL-сертифіката. Для цього необхідно перейти до вкладки "SSL/TLS" у панелі керування платформою CloudFlare та обрати тип сертифіката, який бажано встановити. На платформі CloudFlare є декілька типів SSL-сертифікатів, таких як Flexible, Full, Full (strict) та Origin CA. Кожен тип сертифіката має свої особливості та рівень безпеки, тому необхідно обрати такий тип, який найкраще відповідає поставленим потребам.

Після встановлення SSL-сертифіката на платформі CloudFlare необхідно встановити його на веб-сервері. Цей процес може відрізнятися в залежності від типу сервера та операційної системи. Зазвичай, для встановлення SSL-сертифіката необхідно завантажити файл сертифіката та приватного ключа на сервер та налаштувати веб-сервер на використання цих файлів.

2.7.4 Вибір доменного імені та прив'язка системи

Доменні імена працюють на nameservers від Cloudflare, завдяки чому можна керувати ними зі сторінки самого Cloudflare (рис 2.11).

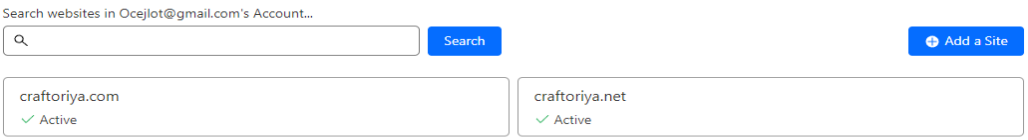


Рисунок 2.11 – Сторінка керування доменними іменами

Це дуже зручно, так як він має дуже багато інструментів для роботи з доменами, SSL сертифікатами, проксюванням, котре приховує IP адрес хостингу, та зручним налаштуванням переспрямовування користувачів, завдяки чому можливо використовувати один й той самий домен для переспрямування гравців як на сайт, так і безпосередньо на сервер майнкрафт, навіть якщо вони будуть знаходитись на різних ip адресах.

Налаштування DNS записів (з прихованою конфіденційною інформацією показано на рис 2.12.

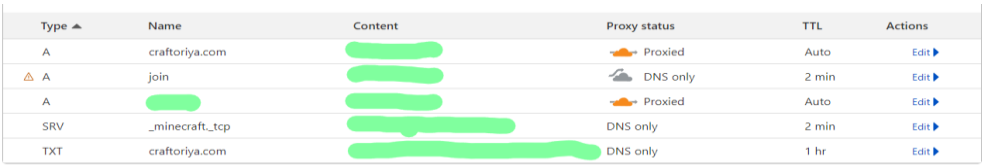


Рисунок 2.12 – Сторінка налаштування DNS

Отримання SSL сертифікату зараз - це дуже проста задача. Завдяки Cloudflare можна легко отримати безкоштовний сертифікат терміном дії 15 років, та вибрати один з 4х видів безпеки підключення до сайту. Перший - відсутність захисту, а останній - використання ключа для підтвердження підключень. Самий безпечний, і саме він використовується в конкретному випадку. Інші 2 варіанти їх комбінації (рис 2.13).

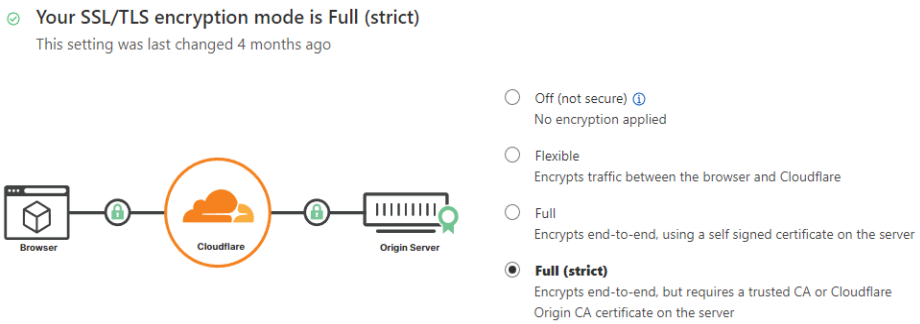


Рисунок 2.13 – Сторінка вибору SSL сертифікату

2.8 Встановлення веб-застосунку та ігрової частини Minecraft сервера на хостинг. Запуск системи

Налаштування сервера Minecraft на Linux є відносно простим процесом. Одним з перших кроків є встановлення Java на сервері. Java Adoptium є одим з найкращих форків, котрий вміє працювати з garbage collector'ом shenandoah.

Щоб встановити Java Adoptium на Linux, спочатку потрібно завантажити відповідний архів з веб-сайту https://adoptium.net/.

Після завантаження архіву, розпакувати його до потрібної директорії за допомогою команди tar: `tar xzf <назва_архіву>.tar.gz -C <шлях_до_директорії>`

Після розпакування архіву необхідно встановити символічне посилання на java в системну директорію /usr/bin, щоб була можливість використовувати її з будь-якої директорії. Для цього треба виконати команду ln з відповідними параметрами: `ln -s /usr/local/java/<назва_директорії>/bin/java /usr/bin/java`

Після встановлення Java треба завантажити сервер Minecraft. Було використано ядро одне з списку форків, описаного вище по документу. Наприклад Paper. Необхідно виконати команду "wget" в терміналі, щоб завантажити файл сервера на Linux-сервер. Після завантаження файлу треба розпакувати його в каталозі, який було обрано заздалегіть.

На цьому етапі можна відредагувати налаштування сервера за допомогою файлу "server.properties". В цьому файлі відбувається зміна таких параметрів, як максимальна кількість гравців, режим гри та інші налаштування.

Запуск сервера - останній крок у встановленні сервера Minecraft на Linux. Це можна зробити шляхом виконання команди "java -Xmx<Кількість озу> -Xms<Кількість озу> -XX:+UseShenandoahGC -jar minecraft_server.jar " у терміналі. Ця команда запустить сервер Minecraft з налаштуваннями,що були вказані раніше.

Налаштування серверу для роботи в інтернеті описано далі. Налаштування фаерволу проводиться через firewalld. Фаервол firewalld є стандартним засобом захисту Linux-систем від зовнішніх зловмисників та неправильної конфігурації мережі. Для його використання необхідно встановити й налаштувати сервер. Одним з варіантів захисту сервера може бути використання проксі-сервера. В даному випадку використано проксі-сервер Nginx.

Після налаштування файерволу firewalld можна перейти до налаштування проксі-сервера nginx. Проксі-сервер може бути використаний для проксювання трафіку між клієнтом та сервером. Це може бути використано для балансування навантаження на декілька серверів, або для забезпечення доступу до різних служб зі звичайного порту HTTP.

Налаштування параметрів Nginx передбачають визначення порту, на якому працює сервер, та кореневої директорії, де зберігаються файли веб-сайту.

Наступним кроком є налаштування віртуального хоста. Це дозволяє встановити правила маршрутизації, які визначають, який запит повинен бути переданий до якого сервера.

3 ТЕСТУВАННЯ ТА ЗАПУСК МЕРЕЖІ МАЙНКРАФТ СЕРВЕРІВ

3.1 Тестування функціоналу веб-застосунку

Основна функція веб-застосунку – поповнення балансу гравця. Відповідно для того, щоб поповнити баланс, необхідно зайти на Craftoriya.com. Вид веб-застосунок мережі майнкрафт-серверів, коли користувач потрапляє на головну сторінку, представлено на (рис. 3.1).

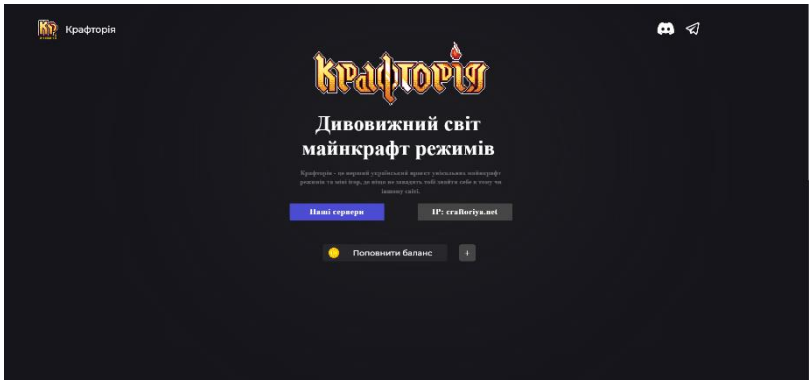


Рисунок 3.1 – Головна сторінка сервера «Craftoriya»

Нижче відображено сервери, їх опис та додаткову інформацію (рис. 3.2).

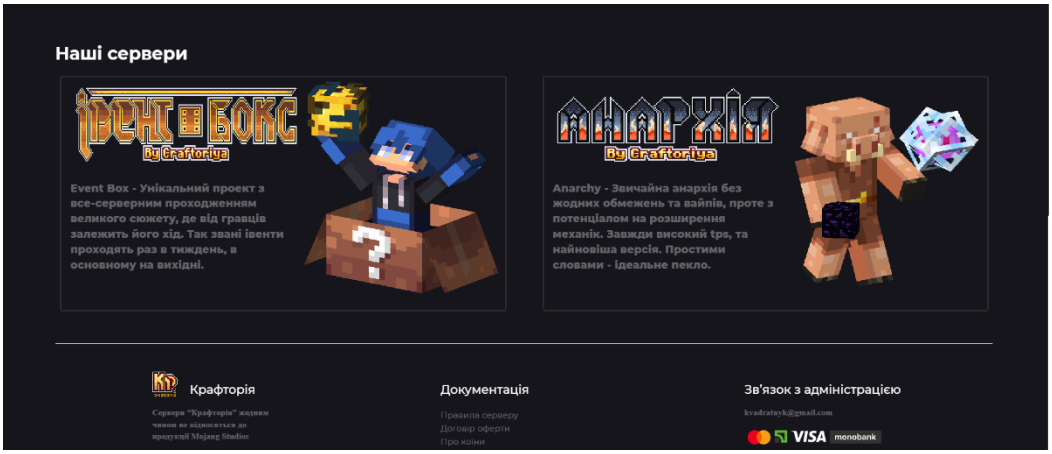


Рисунок 3.2 – Ознайомча частина веб-застосунку «Craftoriya»

Сторінка поповнення представлена на рис. 3.3. Щоб на неї потрапити та

поповнити баланс, необхідно натиснути на кнопку «Поповнити баланс». Після цього треба вказати свій нікнейм на сервері, пошту та бажану суму до поповнення, після чого буде здійснено оплату через Fondy. Надалі всі операції проводяться на сервері.

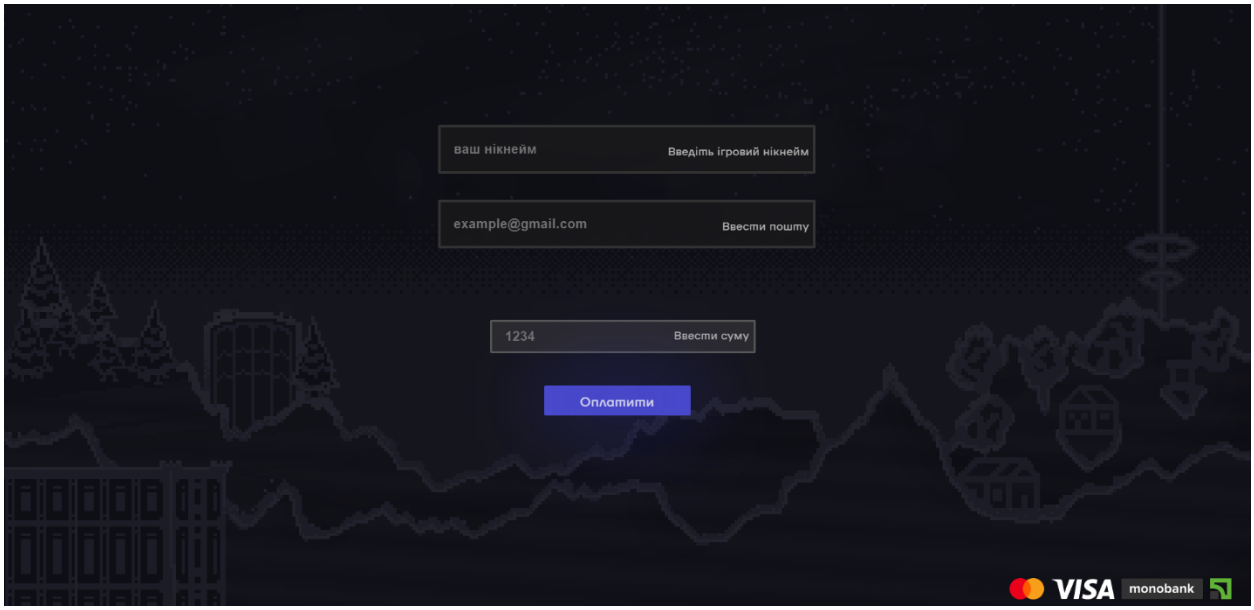


Рисунок 3.3 – Сторінка вводу даних веб-застосунку «Craftoriya»

3.2 Тестування "профілю гравця"

Профіль гравця включає в себе наступні елементи: Голова гравця, що відображає статистику при наведенні на неї. Винагороди гравця, список доступних режимів з платним допуском та пропозиція купити його, вбудований магазин а також значок «Коінів» - внутрішньоігрова валюта. При натисканні на нього з’являється інтерактивне вікно, де запропонує поповнити баланс. у випадку погодження, гравець перенаправляє на сайт, де можна безпосередньо поповнити баланс. Ця система повинна працювати без помилок, адже несправність даної механіки призводить до втрати гравців та прибутку. Далі відображено профіль гравця з усіма необхідними та зазначеними раніше елементами (Рис. 3.4).

Рисунок 3.4 – Профіль гравця на сервері «Craftoriya»

Далі зображено меню купівлі перепусток на платні сервери в магазині, по наведенню вказано ціну та баланс гравця (рис. 3.5).

Рисунок 3.5 – Слот купівлі перепустки на сервер

На рис 3.6 продемонстровано ще один, додатковий розділ магазину, де є можливість придбати для себе кольоровий нікнейм, вказано ціну, термін на котрий буде придбано косметику та баланс гравця. Так само як і при купівлі перепустки – якщо баланс достатній, то сума буде списана з внутрішнього балансу. Якщо ні – то буде запропоновано поповнити його через сайт (рис. 3.6).

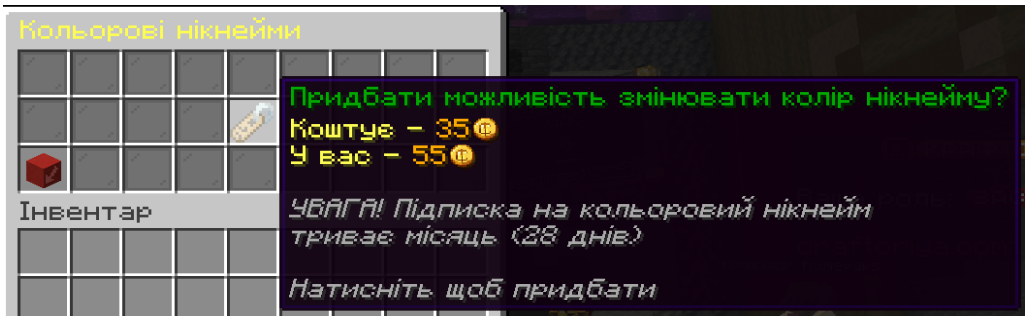


Рисунок 3.6 – Слот купівлі перепустки на сервер

3.3 Публічний тест

На момент написання дипломної роботи сервер Craftoriya працює вже три місяці, тут грають люди, поповнюють баланс, купують платний контент та приймають активну участь в житті сервера. Важливо цілодобово спостерігати за процесом, щоб у разі чого оперативно виправляти мложливі баги. На разі запущено Міні-ігри, де є кілька режимів, окремий сервер «Анархія» та сервер «ІвентБокс» (рис. 3.7).

Рисунок 3.7 – Хаб сервера «Craftoriya»

Гравці безпроблемно можуть зайти на сервер, обрати режим гри та провести на ньому час. Так само поповнити баланс, скористатись магазином чи переглянути свою статистику. За цей час критичних проблем виявлено не було. В першому відкритому тесті приймало участь близько 70 гравців (рис. 3.8).

Рисунок 3.8 – Хаб в момент відкриття сервера «Craftoriya»

ВИСНОВКИ

1. Проаналізовано нішу ігрових Minecraft-серверів та веб-застосунків для підтримки Minecraft-серверів. Визначено переваги та недоліки аналогів. На основі результатів опитування гравців Minecraft-серверів визначено їх вподобання.

2. Сформульовано функціональні та нефункціональні вимоги до сервера та веб-застосунку, які враховують переваги і недоліки аналогів та вподобання гравців.

3. Проведено аналіз засобів для розробки програмного забезпечення. Для розробки веб-застосунку було обрано Sublime Text, для написання плагінів IntelliJ IDEA, серверним ядром було обрано Pufferfish.

4 Розроблено архітектуру веб застосунку, що включає в себе базу даних, модулі для роботи системи оплати, набір плагінів для функціонування сервера та серверне ядро.

5. Розроблено модуль системи оплати через платіжну систему Fondy з використанням технології API.

6. Розроблено архітектуру бази даних для функціонування системи оплати та збереження інформації про гравців з використанням СУБД MySQL. База даних є розподіленою і забезпечує зберігання локальної інформації для кожного серверу окремо.

7. Розроблено профіль гравця та внутрішньоігровий магазин, завдяки чому можна виконувати доступні операції безпосередньо в грі.

8. Виконано розгортання та тестування системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Minecraft.net [Електронний ресурс]. Режим доступу до ресурсу: https://www.minecraft.net/en-us

2. Java.com [Електронний ресурс]. Режим доступу до ресурсу: https://www.java.com/en/

3. Skriptlang [Електронний ресурс]. Режим доступу до ресурсу: https://docs.skriptlang.org

4. Spigotmc [Електронний ресурс]. Режим доступу до ресурсу: https://www.spigotmc.org

5. Papermc [Електронний ресурс]. Режим доступу до ресурсу: https://papermc.io/

6. Hetzner [Електронний ресурс]. Режим доступу до ресурсу: https://www.hetzner.com

7. Amazon [Електронний ресурс]. Режим доступу до ресурсу: https://aws.amazon.com/api-gateway/?nc1=h_ls

8. SublimeText [Електронний ресурс]. Режим доступу до ресурсу: https://www.sublimetext.com/

9. Intellij IDEA [Електронний ресурс]. Режим доступу до ресурсу: https://www.jetbrains.com/

10. MySQL [Електронний ресурс]. Режим доступу до ресурсу: https://www.mysql.com

11. Hypixel [Електронний ресурс]. Режим доступу до ресурсу: https://hypixel.net/

12. Minecraft.org.ua [Електронний ресурс]. Режим доступу до ресурсу: https://minecraft.org.ua/minecraft-servers

13. Researchgate [Електронний ресурс]. Режим доступу до ресурсу: https://www.researchgate.net/publication/3895364_Analysis_and_testing_of_Web_applications

Додаток А.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ПІДТРИМКИ МЕРЕЖІ ІГРОВИХ СЕРВЕРІВ MINECRAFT З ВИКОРИСТАННЯМ ПЛАГІНІВ НА ОСНОВІ SKRIPT

Виконав студент 4 курсу
групи ПД 44
Посенко Данило Романович
Керівник роботи
К.т.н, доц, доцент кафедри ІПЗ Золотухіна Оксана Анатоліївна

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** спрощення взаємодії користувача з сервером Minecraft за рахунок удосконалення системи донату та підвищення зручності особистого кабінету, що реалізовані з використанням плагінів на основі Skript.
- **Об’єкт дослідження** процес взаємодії користувача з ігровим Minecraft сервером.
- **Предмет дослідження** програмне забезпечення для підтримки Minecraft серверів.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз ринку Minecraft серверів та веб-застосунків для підтримки Minecraft серверів, визначити вимоги та вподобання гравцівДослідження веб-застосунків для підтримки Minecraft серверів.
2. Визначити функціональні та нефункціональні вимоги до сервера та веб-застосункуРозробка архітектури веб-застосунку.
3. Провести аналіз засобів для розробки програмного забезпеченняРозробка архітектури бази даних.
4. Розробити архітектуру веб-застосунку.
5. Розробити модуль системи оплати.
6. Розробити та реалізувати архітектуру бази даних для функціонування системи оплати та збереження інформації про гравців.
7. Розробити модуль, що реалізує особистий кабінет (профіль) гравця.
8. Розгорнути та протестувати розроблену систему.

АНАЛІЗ АНАЛОГІВ

	Hypixel	Gummer Craft	UA Polit	Crafroriya
Веб-застосунок	+	+	-	+
Міні-ігри	+	+	-	+
Особистий кабінет	+	-	-	+
Надійність	+	-	+	+
Доступ без ліцензії	-	+	+	+

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1. Прийом платежів
2. Перевірка нікнейму гравця
3. Перелік доступних режимів гри на сервері
4. Функціональний внутрішньоігровий магазин
5. Можливість переглядати баланс
6. Наявність спеціальних винагород
7. Сумісність версій
8. Можливість переглядати баланс

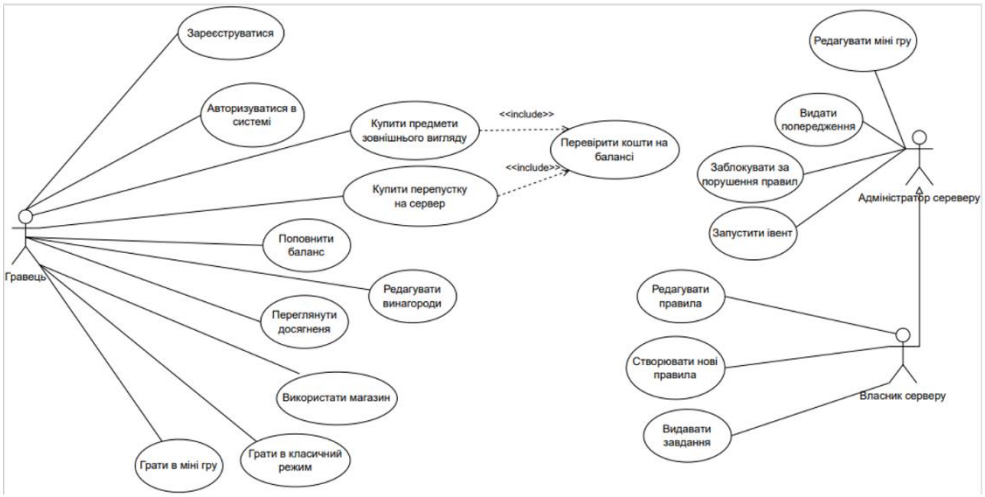
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Skript

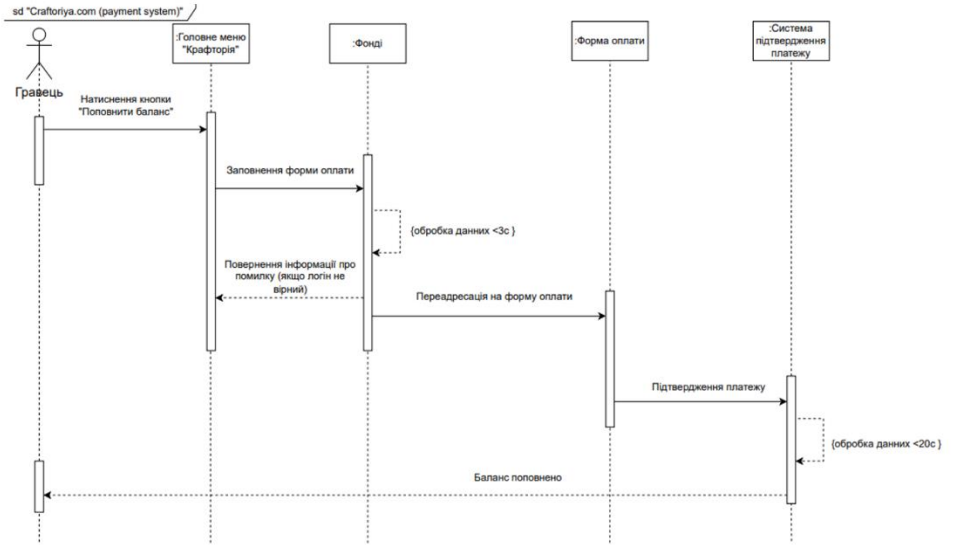
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



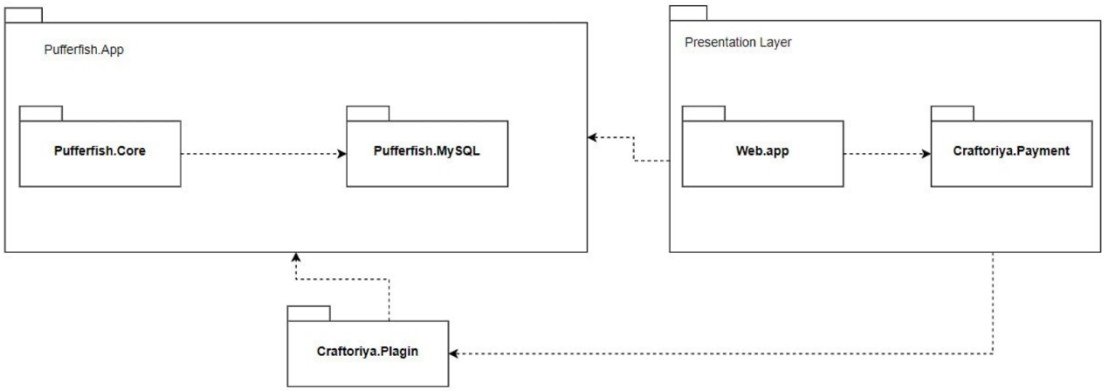
7

ДІАГРАМА ПОСЛІДОВНОСТІ ПОПОВНЕННЯ БАЛАНСУ



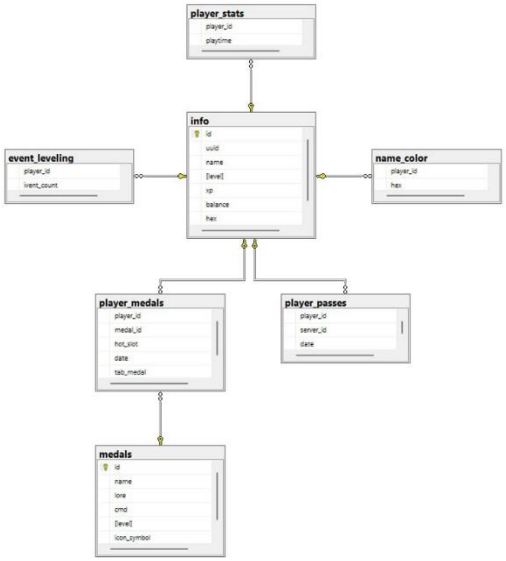
8

ДІАГРАМА АРХІТЕКТУРИ ВЕБ-ЗАСТОСУНКУ



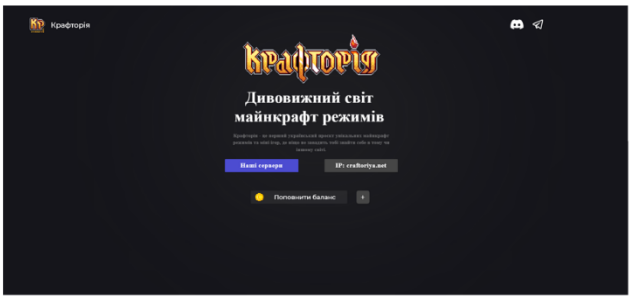
9

СХЕМА БД



10

ЕКРАННІ ФОРМИ

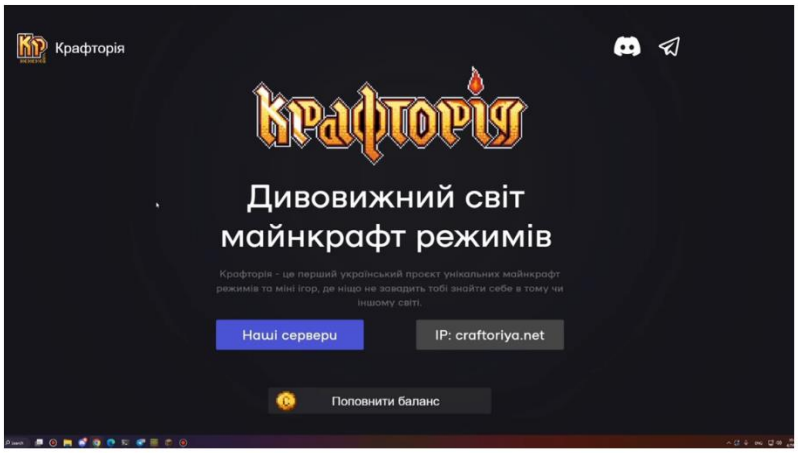


Головний екран веб-застосунку

Профіль гравця

11

ДЕМОНСТРАЦІЯ РОБОТИ СИСТЕМИ ОПЛАТИ



12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Посенко Д.Р. Розробка веб-застосунку для підтримки мережі ігрових серверів Minecraft з використанням плагінів на основі SKRIPT / Золотухіна О.А, Посенко Д.Р // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії, 01- 03 червня 2023р., ДУТ, м. Київ - К: ДУТ, 2023. Подано до друку.
2. Посенко Д.Р. Опис технології API/ Золотухіна О.А, Посенко Д.Р // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії, 01- 03 червня 2023р., ДУТ, м. Київ - К: ДУТ, 2023. Подано до друку.

ВИСНОВКИ

1. Проаналізовано нішу ігрових Minecraft-серверів та веб-застосунків для підтримки Minecraft-серверів. Визначено переваги та недоліки аналогів. На основі результатів опитування гравців Minecraft-серверів визначено їх вподобання.
2. Сформульовано функціональні та нефункціональні вимоги до сервера та веб-застосунку, які враховують переваги і недоліки аналогів та вподобання гравців.
3. Проведено аналіз засобів для розробки програмного забезпечення. Для розробки веб-застосунку було обрано Sublime Text, для написання плагінів IntelliJ IDEA, серверним ядром було обрано Pufferfish.
4. Розроблено архітектуру веб застосунку, що включає в себе базу даних, модулі для роботи системи оплати, набір плагінів для функціонування сервера та серверне ядро.
5. Розроблено систему оплати через платіжну систему Fondy з використанням технології API.
6. Розроблено архітектуру бази даних для функціонування системи оплати та збереження інформації про гравців з використанням СУБД MySQL. База даних є розподіленою і забезпечує зберігання локальної інформації для кожного серверу окремо..
7. Розроблено профіль гравця та внутрішньоігровий магазин, завдяки чому можна виконувати доступні операції безпосередньо в грі.
8. Виконано розгортання та тестування системи.

Додаток Б.
ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

 контейнери, що містять в собі всі актуальні під-сервера проєкту, наведено в лістингу програмного коду (Додаток Б).

```
        <div class="image_wrapper2" style="margin-left: -245px" id="serv1">
        <div class="text" style="
        text-align: left;width: 192px; margin-left: 10px; height:210px;font-family: Montserrat, sans-serif; font-size: 13px; padding: 15px">
            Event Box
            Унікальний проект з все-серверним проходженням великого сюжету, де від гравців залежить його хід. Так звані івенти проходять раз в тиждень, в основному на вихідні.
        </div>
        <button class="CLASS" style="font-family: Montserrat, sans-serif; " onclick="
let z = document.getElementById('serv1').style.opacity;
            document.getElementById('serv1').style.opacity = z==='0'?'1':'0';
    ">Назад</button>
    </div>
    <div class="image_wrapper" id="serv2_">
        <img src="../img/anarhy.png" style="width: 180px; height: 53px; margin-left: 30px; margin-top: 15px" class="image">
        <img src="../img/anarchy_screen.webp" style="width: 232px;height: 119px; background: #2C2C2C; margin-left: 4px;margin-top: 10px" class="image">
        <div class="text" style="height:35px;text-align: left;width: 192px; margin-left: 10px; font-family: Montserrat, sans-serif; font-size: 13px">
            Anarchy - стабільний,
            та красивий хаос
        </div>
```

```
<button class="CLASS" onclick="
let z = document.getElementById('serv2_').style.zIndex;
document.getElementById('serv2_').style.zIndex = z==='0'?'1':'0';
">Детальніше</button>
</div>
<div class="image_wrapper2" style="margin-left: -245px" id="serv2">
<div class="text" style="
text-align: left;width: 192px; margin-left: 10px; height:210px;font-family: Montserrat, sans-serif; font-size: 13px; padding: 15px">
Anarchy
Звичайна анархія без жодних обмежень та вайпів, проте з потенціалом на розширення механік. Завжди високий tps, та найновіша версія. Простими словами - ідеальне пекло.
</div>
```

Backend частина веб-застосунку. Встановлення додатку, який буде працювати на веб-сервері, обробляти запити користувачів та зберігати/зчитувати дані з баз даних.

```
const express = require('express');
const mysql = require('mysql');
const cookieParser = require('cookie-parser');
const fs = require('fs');
const path = require('path')
const sha1 = require('sha1');
const CloudIpsp = require('cloudipsp-node-js-sdk');
const app = express()
const https = require('https');
const { join } = require('path');
```

```
const port = ;
const merchant_id = ;
var filepath
const secretKey = ""

const fondy = new CloudIpsp(
    {
      merchantId: merchant_id,
      secretKey: secretKey
    }
  )
function filedir(filename) {
    filepath = fs.readFileSync(path.join(__dirname, "/public/", filename), "utf-8", (err,
data) => {
        if (err) throw err;
        console.log(data)
    })
    return(filepath)
}

const playersdbConfig = {
  host: 'localhost',
  user: 'server',
  password: '',
  database: 'playersinfo',
  connectionLimit: 20
};

const donationsdbConfig = {
  host: 'localhost',
```

```
  user: 'server',
  password: '',
  database: 'donates',
  connectionLimit: 20
};
const playersdbPool = mysql.createPool(playersdbConfig);
const donationsdbPool = mysql.createPool(donationsdbConfig);

playersdbPool.on('error', (err) => {
  console.error('Failed to connect to the players database:', err);
});

donationsdbPool.on('error', (err) => {
  console.error('Failed to connect to the donations database:', err);
});
```