Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

(повна назва факультету)

(повна назва кафедри)

1 **КВАЛІФІКАЦІЙНА РОБОТА**

на здобуття освітнього ступеня

(назва освітнього ступеня)

на тему:

Виконав(ла): студент(ка) _____ курсу, групи _____
спеціальності

(шифр і назва спеціальності)

(підпис) (прізвище та ініціали)

Керівник
(підпис) (прізвище та ініціали)

Нормоконтроль
(підпис) (прізвище та ініціали)

Завідувач кафедри
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

Тернопіль
20___

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет _____
(повна назва факультету)

Кафедра _____
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)
« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня _____
(назва освітнього ступеня)

за спеціальністю _____
(шифр і назва спеціальності)

студенту _____
(прізвище, ім'я, по батькові)

1. Тема роботи _____

Керівник роботи _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 20___ року № _____

2. Термін подання студентом завершеної роботи _____

3. Вихідні дані до роботи _____

4. Зміст роботи (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

2 КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	3 Примітка

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Обсяг звіту - 60 сторінок, кількість частин звіту - 4 розділи, кількість ілюстрацій - 44, кількість таблиць - 1, кількість джерел по переліку посилань - 14.

Мета проекту: створити клієнт-серверний додаток для настільної рольової гри Dungeons & Dragons, який дозволяє гравцям та майстрам гри взаємодіяти в реальному часі, управляти персонажами, проводити бої та кампанії. Використовуючи Socket для мережевої комунікації та Xamarin.Forms для створення кросплатформного інтерфейсу користувача, розробка охоплює створення стабільного та ефективного сервера, а також зручного клієнтського додатку. Система базується на інтерактивному взаємодії між гравцями, автоматизації процесів гри та забезпечення глибокого занурення в ігровий світ. Додаток розв'язує проблему координації та управління ігровими сесіями D&D, забезпечуючи зручність в організації ігор та підвищуючи загальну доступність та привабливість настільної рольової гри.

Ключові слова: C#, Xamarin, Клієнт-сервер, D&D, sequence diagram, use-case diagram, class diagram, MessagingCenter, binding.

ANNOTATION

The report comprises 60 pages, consists of 4 sections, includes 44 illustrations, 1 table, and references 14 sources in the bibliography.

The project's objective is to develop a client-server application for the tabletop role-playing game Dungeons & Dragons, which allows players and game masters to interact in real-time, manage characters, conduct battles, and campaigns. Utilizing Socket for network communication and Xamarin.Forms for creating a cross-platform user interface, the development encompasses building a stable and efficient server, as well as a user-friendly client application. The system is based on interactive player engagement, automation of game processes, and ensuring a deep immersion into the gaming world. The application addresses the issue of coordinating and managing D&D game sessions, enhancing the convenience of organizing games and increasing the overall accessibility and appeal of the tabletop role-playing game.

Keywords: C#, Xamarin, Client-server, D&D, sequence diagram, use-case diagram, class diagram, MessagingCenter, binding.

ЗМІСТ

ВСТУП 4
1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ 6
1.1 Огляд існуючих зразків програмного забезпечення 6
1.2 Визначення критеріїв до розроблюваного програмного продукту 13
1.3 Середовище та мова програмування 14
2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ДОДАТКУ 16
2.1 Клієнт-серверна архітектура 16
2.2 Діаграма прецедентів 17
2.3 Діаграма класів 19
2.4 Діаграма послідовності 25
3 КОНСТРУЮВАННЯ КЛІЄНТ СОКЕТНОГО ДОДАТКУ 27
3.1 Розгляд процесу дії сервера 27
3.2 Опис коду клієнта 29
3.3 Опис MessagingCenter 30
3.4 Опис переходів між сторінками 32
3.5 Опис CombatPage 33
3.6 CreateCharacter 35
3.7 DiceRollPage 37
3.8 TradingPage 38
3.9 Тестування 39
4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТАЦІЯХ 44
4.1 Охорона праці 44
4.2 Безпека в надзвичайних ситуаціях 47
ВИСНОВОК 52
ПЕРЕЛІК ЛІТЕРАТУРИ 53
ДОДАТКИ 55
Додаток А 56
Додаток Б ..... **Помилка! Закладку не визначено.**

ВСТУП

З моменту свого виникнення Dungeons&Dragons пройшла значний розвиток і стала культурним явищем, впливаючи на різноманітні форми розваг і сприяючи розквіту живої глобальної спільноти гравців. Згадки про гру можна знайти в літературі, фільмах, телесеріалах і відеоіграх. Крім самої настільної гри, бренд Dungeons&Dragons розширився на інші медіа форми, такі як книги, фільми, відеоігри та серіали. Це стало підтвердженням широкого впливу гри на сучасну поп-культуру і визначило його як важливий елемент розваг та розвитку творчих здібностей. Завдяки своїй унікальній концепції, Dungeons&Dragons ставить акцент на творчість, стратегію та взаємодію гравців. Гра відома своєю здатністю об'єднувати гравців, створюючи командний дух та розвиваючи навички співпраці. Гравці утворюють міцні зв'язки, спільно розплітаючи історії та стикаючись з викликами. Гра надає платформу для з'єднання друзів і поділу творчих вражень.

Дана робота є значущою у зв'язку із стрімким розвитком галузі розробки мобільних додатків та використання технологій, таких як Xamarin та сокет-з'єднання, в контексті створення додатку для гравців настільно-рольової гри Dungeons&Dragons. Особливо у зв'язку із глобальною пандемією коронавірусу, яка вплинула на всі сфери життя, включаючи традиційні форми розваг, можливість грати в очному форматі стала значно обмеженою. Цей напрям дозволяє забезпечити високий рівень інтерактивності, зручності для користувачів під час гри та забезпечити можливість продовжувати гру в умовах соціального дистанціювання та обмежень на збори в групах.

Метою даної роботи є вивчення можливостей інтеграції технологій Xamarin та Socket-з'єднань для створення мобільного додатку, призначеного для гравців у Dungeons&Dragons. Дослідження охоплює вивчення можливостей Xamarin для кросплатформенної розробки, а також реалізацію механізмів синхронізації через сокети для спільної гри гравців у режимі реального часу.

Об'єктом дослідження є розробка android-додатку для гравців настільно-рольової гри Dungeons&Dragons з використанням xamarin та socket, який базується на технологіях Xamarin та Socket-з'єднань. Предметом дослідження є вивчення можливостей цих технологій для забезпечення плавної інтеракції гравців у грі.

Ціль даної роботи - створення мобільного додатка, який використовує переваги Xamarin для кросплатформенної сумісності та сокет-з'єднань для забезпечення спільної гри в Dungeons&Dragons. Результатом роботи має бути готовий до використання Dungeons&Dragons додаток з можливістю взаємодії гравців та обміну даними у режимі реального часу.

Кваліфікаційна робота студента магістра спеціальності «121 Інженерія програмного забезпечення» є завершальним етапом його навчання на магістерському рівні і є обов'язковою складовою частиною процесу отримання вищої освіти у галузі програмної інженерії. Ця робота має на меті виявлення та демонстрацію глибоких знань студента у вибраній галузі, а також здатності застосовувати ці знання на практиці.

4 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

Темою кваліфікаційної роботи є розробка системи управління настільною рольовою грою. Орієнтиром у даній сфері є настільна рольова гра Dungeons&Dragons, в якій учасники грають роль уявних персонажів у фантазійному світі. Гра складається з елементів імпровізації, співпраці та пригод, які вирішуються за допомогою правил, кубиків, фантазії та уяви. У Dungeons&Dragons майстер гри, відомий як майстер гри, відповідає за створення світу, сценарію та розробку пригод, в яких беруть участь гравці. Інші учасники гри керують своїми уявними персонажами, вибираючи їхні дії та реагуючи на ситуації, що створює майстер гри [1].

Основна мета гри - це спільне створення оповіді через пригоди та розвиток персонажів шляхом взаємодії та вирішення завдань, викликів та загадок. У Dungeons&Dragons використовуються кістки для випадкових кидків, що додає несподіваність та напругу до гри. Гра сприяє творчому мисленню, співпраці, вирішенню проблем та розвитку уяви та соціальних навичок.

Проте новачки часто зустрічають труднощі у процесі гри через повільне освоєння механік гри та велику увагу, яку вони залучають. Тому на сьогоднішній день існують онлайн сервіси, які беруть на себе рутину ведення записів та розрахунків дій гравців, надаючи більше простору та часу для насолоди кожним моментом пригоди.

4.1 Огляд існуючих зразків програмного забезпечення

4.1.1 D&D Beyond

D&D Beyond, мобільна версія популярного цифрового інструменту для гри Dungeons & Dragons (рис 1.1), є справжнім бенкетом для шанувальників рольових

ігор. Ця платформа забезпечує швидкий доступ до величезного арсеналу ресурсів Dungeons&Dragons, включаючи правила, напрямки, ігрові інструменти, і навіть засоби для створення персонажів.

D&D Beyond представляє собою набір цифрових інструментів та онлайн-колекцію книг, доступних для придбання та використання в межах Dungeons&Dragons 5e. Початково розроблений у 2017 році компанією Curse LLC, пізніше у 2018 році він був придбаний Fandom Inc., яка протягом чотирьох років активно розвивала його функціонал [2].

Використовуючи мобільну версію D&D Beyond, гравці та ігрові майстри можуть легко організовувати свої ігрові сесії. Це стає можливим завдяки інтеграції з обліковим записом користувача на сайті D&D Beyond, де зберігаються всі дані про персонажів, набори кампаній та ігрові матеріали. Мобільна версія дозволяє зберігати важливу інформацію офлайн, що є надзвичайно корисним для гри в місцях з обмеженим доступом до Інтернету.

Один з ключових елементів D&D Beyond — це інструмент для створення персонажів, який відіграє важливу роль у грі. Він дозволяє гравцям крок за кроком створити унікального персонажа, вибираючи расу, клас, здібності, заклинання та багато іншого. Це процес не просто технічний, але й творчий, оскільки гравці можуть втілити свої фантазії у життя.

Наступною перевагою мобільної версії є доступ до цифрових книг Dungeons&Dragons. Гравці можуть купувати та завантажувати офіційні видання, такі як "Player's Handbook" чи "Monster Manual", які надають глибоке занурення в світ Dungeons&Dragons. Ці ресурси є незамінними для розуміння глибини та складності гри.

Мобільна версія D&D Beyond також включає різноманітні інструменти для ігрових майстрів, допомагаючи їм у підготовці та веденні кампаній. Зручність управління персонажами, відстеження статистики та доступ до великої кількості ігрових матеріалів значно полегшують проведення ігрових сесій.

Рисунок 4.1.1 Логотип D&D Beyond

Можливості гравця безкоштовної версії програми:

1 Створення до шести персонажів безкоштовно дозволяє як новачкам, так і досвідченим гравцям розвивати та випробовувати нові концепції персонажів.

2 Online-таблиця персонажів для гри на планшеті, телефоні або ноутбуці. Ця можливість надає зручний доступ до:

– Відстеження стану персонажів, включаючи здоров’я, слоти для заклинань, очки харизми тощо.

– Швидкий доступ до текстів заклинань і здібностей з одного кліка, що дозволяє швидко ознайомитися з їх ефектами та роботою.

– Функції короткого та тривалого відпочинку допомагають повернутися до належних рівнів та швидше здійснювати оновлення.

3 Оновлення при переході на новий рівень - інструмент спрощує процес оновлення персонажів.

4 Легке роздрукування аркуша гравця дозволяє тим, хто надає перевагу паперовому формату, легко мати доступ до аркушів гравця, що відповідають стандартам довідника гравця.

Можливості DM безкоштовної версії програми:

1 Дозволяє зберегти вісім кампаній Encounter Builder

2 Campaign Manager, що відповідає за організацію кампаній та запрошення гравців для приєднання.

3 Homebrew Creations - набір інструментів, що допомагає створювати власні елементи:

– Створення власних магічних предметів, пов'язаних зі заклинаннями, які вони виконують.

– Розробка власної зброї з крутими назвами, бонусами до шкоди та навіть магічними властивостями.

– Створення власних монстрів - клонування та модифікація існуючих або створення з нуля.

4 Encounter Builder дозволяє додавати різноманітних монстрів до зустрічей, надаючи їм назви. Він також порівнює вашу зустріч із здібностями вашої групи, що дає уявлення про складність зіткнення та кількість отриманих XP. Наприклад, якщо ви плануєте додати жуків та гоблінів у зіткнення, це попередить вас про його складність для вашої групи першого рівня або його легкість для групи дванадцятого рівня.

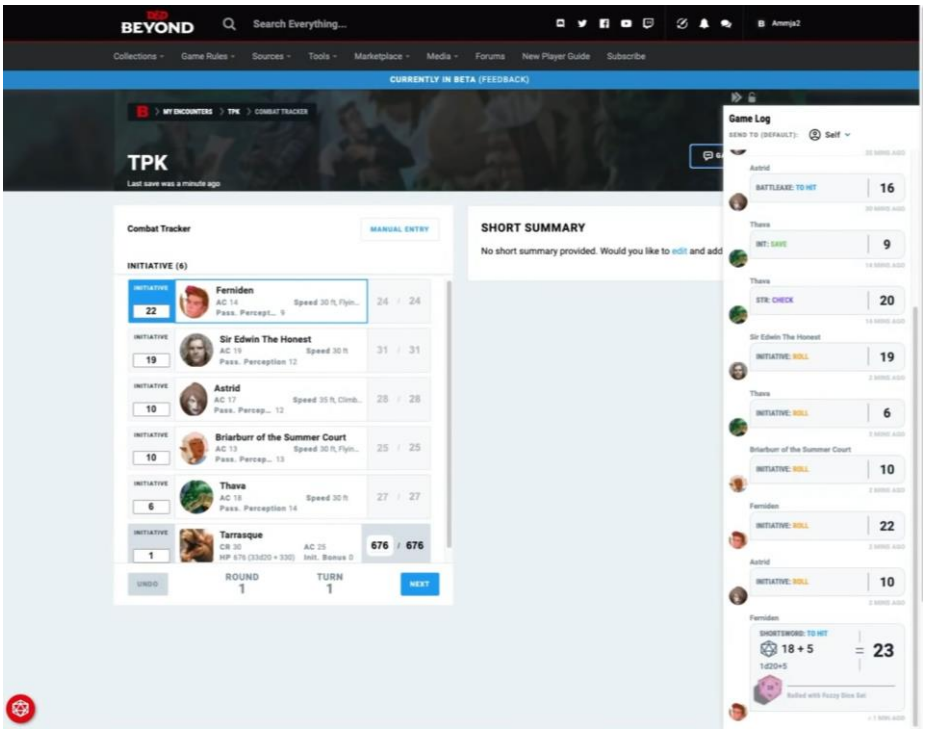


Рисунок 4.1.2 - Combat Tracker

5 Combat Tracker (рис. 1.1.2) слугує для відстеження порядку ходу та статистики монстрів під час бою. Ви можете використовувати зіткнення, створені за допомогою Encounter Builder, або швидко налаштувати бій. Особливо корисно, що він автоматично кидає ініціативу для монстрів, тим часом як гравці кидають свою ініціативу. В безкоштовній версії є обмеження на 8 зіткнень.

Платна версія знімає всі обмеження та надає можливість для Dangeon Master поширювати куплені доповнення для групи гравців, використовувати в своїх іграх карти та хмарне сховище на 10Gb за 4.58 долара США в місяць.

4.1.2 Fight Club 5th Edition

"Fight Club 5th Edition" (рис. 1.1.3) є надзвичайно корисним мобільним додатком для гравців та ігрових майстрів, що захоплюються п'ятою редакцією настільної рольової гри Dungeons&Dragons [3]. Цей додаток відрізняється своєю гнучкістю та зручністю, оскільки надає користувачам інструменти для створення та управління персонажами, а також засоби для ефективної організації ігрових сесій.

Рисунок 4.1.3 - Логотип Fight Club 5th Edition

Основною функцією "Fight Club 5th Edition" є можливість створення деталізованих персонажів. Гравці можуть вибирати різноманітні раси, класи, здібності, заклинання, та інше, що дозволяє глибоко персоналізувати своїх героїв.

Це не тільки сприяє зануренню в ігровий світ, але й робить кожного персонажа унікальним і запам'ятовуємим.

Додаток також містить різноманітні інструменти для управління інвентарем та заклинаннями, що є важливою частиною гри D&D. Він дозволяє легко відстежувати оснащення персонажів, включаючи зброю, броню, магічні предмети, та інші ресурси, які гравці можуть використовувати під час пригод.

"Fight Club 5th Edition" також пропонує функції для ведення бою. З його допомогою можна легко відслідковувати ініціативу, здоров'я, та інші важливі параметри, що спрощує проведення бойових епізодів та забезпечує плавність ігрового процесу.

Однією з унікальних особливостей цього додатку є його здатність імпортувати та експортувати дані персонажів. Це робить можливим швидке ділення інформацією з іншими гравцями або ігровими майстрами, сприяючи ефективній колаборації та обміну ігровими даними.

Крім того, "Fight Club 5th Edition" надає доступ до різноманітних ресурсів, таких як ігрові правила та посібники, що допомагає новачкам швидше освоїти гру, а досвідченим гравцям — поглибити свої знання.

У цілому, "Fight Club 5th Edition" є незамінним помічником для любителів D&D. Він не тільки полегшує управління персонажами та ігровими сесіями, але й збагачує ігровий досвід, роблячи його більш гладким та захоплюючим.

Fight Club 5-го видання працює як універсальний цифровий аркуш персонажа, надаючи все необхідне для участі у грі Dungeons & Dragons 5-го видання. Будь-яка деталь, зазвичай записана на аркуші персонажа 5-го видання, може бути введена безпосередньо в Fight Club. Він керує основними даними про вашого персонажа (наприклад, показники характеристик, навичок, вмінь) і автоматично рахує всі пов'язані статистики [3].

4.1.3 Game Master 5th Edition

"Game Master 5th Edition" (рис. 1.1.3)– це мобільний додаток, створений спеціально для ігрових майстрів, які проводять сесії в рамках п'ятої редакції настільної рольової гри "Dungeons & Dragons" [4]. Цей інструмент є надзвичайно корисним для організації та управління ігровими кампаніями, забезпечуючи гладке та зручне ведення гри.

Рисунок 4.1.4 - Логотип Game Master 5th Edition

Однією з основних характеристик додатку є його здатність допомагати ігровим майстрам у створенні та веденні ігрових сесій. "Game Master 5th Edition" надає можливість створювати детальні сценарії, включаючи місцевості, персонажів не-гравців (NPCs), та зустрічі з монстрами. Це дозволяє майстрам заздалегідь підготуватися до різних ігрових ситуацій, забезпечуючи плавність та гнучкість ведення кампаній.

Додаток також включає в себе інструменти для легкого ведення бою. Він дозволяє швидко організовувати бойові зустрічі, відстежувати ініціативу та здоров'я учасників, а також управляти заклинаннями та здібностями монстрів. Це забезпечує ігровому майстру зручний спосіб управління бойовими сценами, що є ключовою частиною багатьох Dungeons&Dragons кампаній.

Крім того, "Game Master 5th Edition" допомагає в організації та відстеженні інвентарю персонажів та NPC, що включає зброю, броню, магічні предмети, та

інші ресурси. Ця функціональність дозволяє майстру тримати все під контролем, забезпечуючи точність та узгодженість у грі.

Додаток також надає доступ до різноманітних ресурсів, таких як правила гри, посібники, та карти, що робить його ідеальним інструментом для планування та ведення ігрових сесій. Ці ресурси допомагають ігровому майстру збагатити світ кампанії та створити глибокі та захоплюючі пригоди. Також ви можете експортувати та імпортувати дані між пристроями, спрощуючи обмін інформацією про ваші сесії гри.

Додатки Game Master 5th Edition і Fight Club 5th Edition розроблені компанією Lion's Den і накладають обмеження для безкоштовного використання кількістю створених персонажів та ігрових кампаній кількістю в 1 штуку. Вартість повного доступу складає 2.99 долара США для кожного додатку окремо.

4.2 Визначення критеріїв до розроблюваного програмного продукту

При оцінці сучасного ринку цифрових версій настільних рольових ігор, були визначені ключові критерії для створення оптимальної платформи, що відповідала б потребам користувачів. Серед них:

- Зручний користувацький інтерфейс – це базовий, але критичний аспект. Зрозумілість, інтуїтивність та легкість використання інтерфейсу дозволить користувачам швидко освоїти платформу і отримати задоволення від гри.
- Створення та ведення блокноту персонажа – це інструмент для зберігання та оновлення інформації про персонажів, який дозволяє зручно відслідковувати їх характеристики та розвиток.
- Автоматичний розрахунок результату кинутих кубів – надає зручність у проведенні ігрових моментів, сприяє швидкості та точності у визначенні результатів дій.

• Відстеження порядку ходу персонажів під час бою – допомагає у збереженні порядку та впорядкуванні інформації під час активних сцен у грі.

• Створення персонажів та предметів від імені майстра гри – це важлива функція, що дозволяє майстру гри впливати на гру та надавати унікальні предмети та персонажів для гравців.

У доповнення до цього, вважаю, що такі функції, як автоматичне оновлення інформації для майстра гри та гравців, імпорт та експорт доповнень для рольових ігор з інших додатків та використання хмарного сховища для синхронізації профілю на різних пристроях, є важливими для забезпечення зручності та доступності, дозволяючи користувачам бути підготовленими та організованими навіть на різних пристроях або платформах.

4.3 Середовище та мова програмування

Xamarin – це потужний інструмент для розробки мобільних додатків, який дозволяє розробникам створювати нативні додатки для різних платформ, таких як iOS, Android і Windows, використовуючи єдину кодову базу на мові C#. Xamarin був створений компанією Xamarin Inc., яку пізніше придбала Microsoft [5].

Xamarin використовує Mono, версію .NET Framework, яка забезпечує сумісність між різними платформами. Це дозволяє використовувати повний набір функцій мови C# і бібліотек .NET, а також інтегруватися з різними інструментами та бібліотеками .NET. Також не менш важливим є його інтеграція з Visual Studio, що робить процес розробки плавним і зручним, особливо для тих, хто вже знайомий з середовищем Microsoft.

В цілому, Xamarin відкриває широкі можливості для розробників, які хочуть створювати високоякісні, нативні мобільні додатки, економлячи час і ресурси завдяки перевикористанню коду.

Сокет – це концептуальний термін в області комп'ютерних мереж та програмування, який відноситься до кінцевої точки у мережевому з'єднанні [6]. Сокети є основними елементами для побудови мережевого взаємодії між різними пристроями та дозволяють програмам відправляти та отримувати дані через мережу.

У контексті мережевого програмування, сокет – це інтерфейс (API), який забезпечує можливість програмам встановлювати з'єднання для обміну даними. Кожен сокет асоціюється з певним портом на хості, дозволяючи таким чином мережевому трафіку бути спрямованим до конкретної програми.

Сокети є фундаментальними для створення різноманітних мережевих застосунків, від простих чат-клієнтів до складних розподілених систем. Вони забезпечують низькорівневий механізм взаємодії між комп'ютерами в мережі, дозволяючи програмному забезпеченню обмінюватися даними незалежно від апаратного забезпечення чи платформи.

5 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ДОДАТКУ

5.1 Клієнт-серверна архітектура

Клієнт-серверна архітектура (рис 2.1.1) представляє собою модель комп'ютерної мережі, де декілька клієнтських пристроїв запитують і отримують ресурси та сервіси від основного, централізованого сервера. Це може відбуватися через локальну мережу або через Інтернет. Клієнти використовують спеціальні програми для з'єднання з сервером, який, у свою чергу, займається обробкою запитів та управлінням даними.

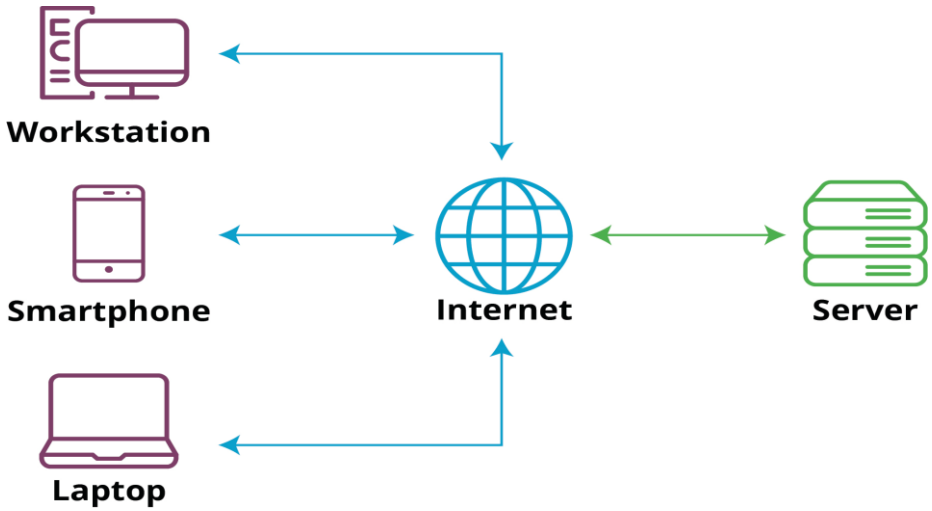


Рисунок 5.1.1 - Діаграма клієнт-серверного зв'язку

У клієнт-серверній мережі, один або кілька комп'ютерів виконують роль сервера, у той час як інші пристрої мережі, такі як комп'ютери або мобільні пристрої, діють як клієнти, звертаючись до сервера за різними ресурсами. Сервери керують доступом до апаратного і програмного забезпечення, забезпечуючи централізоване місце для програм, файлів та інших ресурсів. Наприклад, сервер може зберігати важливу базу даних клієнтів, до якої співробітники можуть мати доступ через мережу.

Існують також спеціалізовані види серверів, які виконують конкретні завдання. Наприклад, файловий сервер займається зберіганням і управлінням

файлами, сервер друку управляє друком документів і принтерами, сервер бази даних обробляє запити до баз даних, мережевий сервер керує мережевими з'єднаннями, а веб-сервер обслуговує запити на веб-сторінки. Клієнт-серверні мережі зазвичай забезпечують ефективний спосіб підключення великої кількості комп'ютерів і часто потребують адміністрації з боку мережевого адміністратора через їхню складність і масштаб.

Одним з ключових елементів цієї архітектури є модульність. Сервери можуть бути спеціалізовані для різних завдань, що дозволяє збільшити ефективність та оптимізувати ресурси. Клієнт-серверні системи можна легко масштабувати, додаючи більше серверів або оновлюючи існуючі для підтримки більшої кількості клієнтів або для обробки більш складних завдань. Що є особливо корисно для бізнесів, які ростуть і потребують збільшення їхньої IT-інфраструктури. Не менш важливим, є зручність управління. Завдяки централізації ресурсів, адміністраторам легше управляти мережею, оновлювати системи, впроваджувати оновлення безпеки та виконувати резервне копіювання.

Клієнт-серверна архітектура є не тільки фундаментом для багатьох сучасних мережевих систем, але й гнучким рішенням, яке може задовольнити широкий спектр потреб і вимог.

5.2 Діаграма прецедентів

Діаграма прецедентів, відома також як діаграма випадків використання, є ключовим інструментом у методології об'єктно-орієнтованого аналізу та проектування, особливо в рамках Unified Modeling Language. Ця діаграма відіграє важливу роль у визначенні та документуванні функціональних вимог до системи, відображаючи взаємодію між користувачами або іншими системами і системою для досягнення конкретних цілей [7].

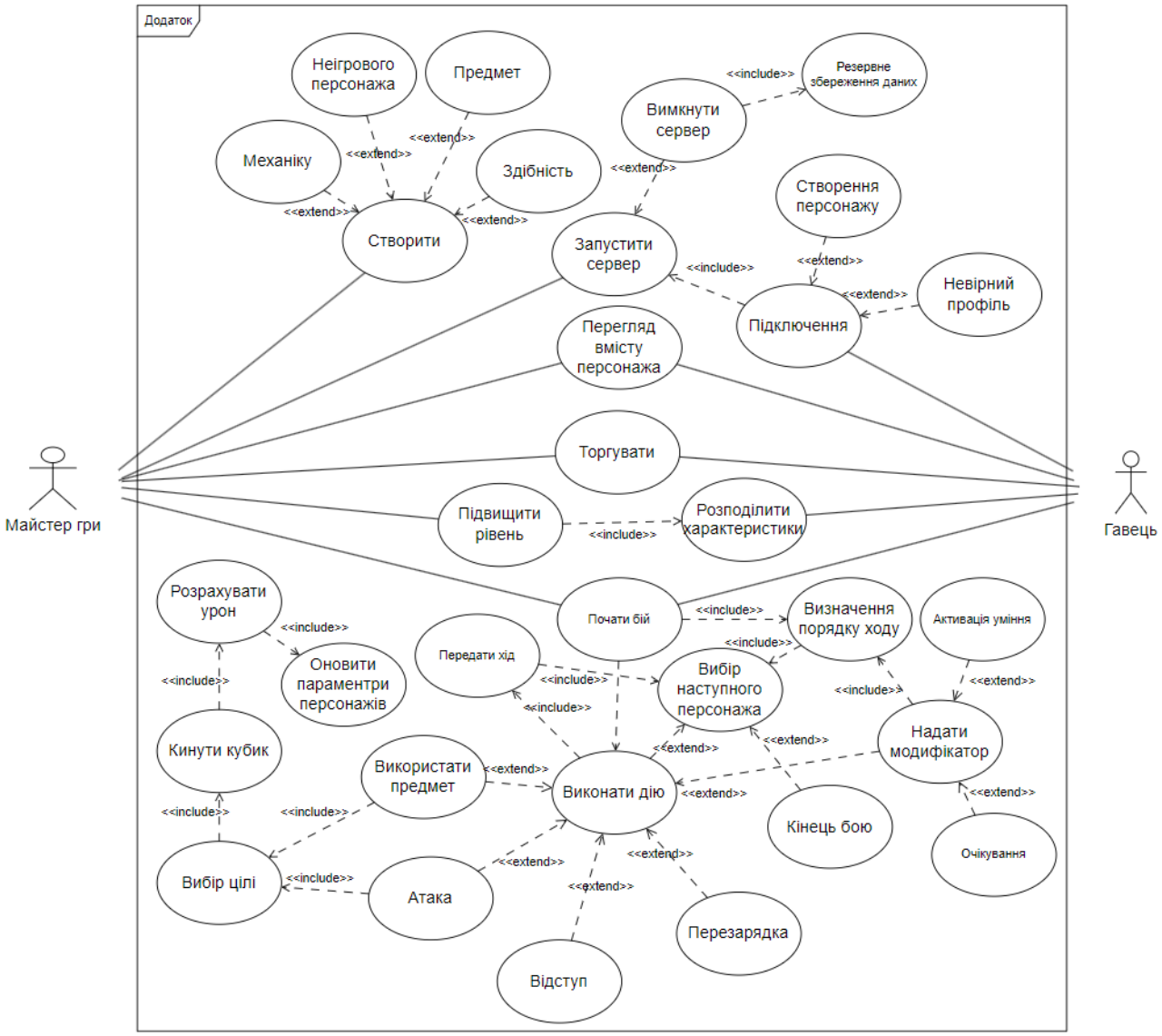


Рисунок 5.2.1 - діаграма прецедентів

Діаграма випадків використання представляє собою засіб для опису функцій або сервісів, які система пропонує своїм користувачам або інтерфейсується з іншими системами. Цей інструмент дозволяє виявити і зрозуміти вимоги та очікування користувачів від системи, виокремлюючи конкретні задачі, які система має виконувати. Він спрощує уявлення про те, які дії можуть здійснювати користувачі або інші системи у взаємодії з даною системою, ілюструючи взаємозв'язки між системою та її користувачами.

Цей вид діаграми також виявляє потенціал для взаємодії з зовнішніми ентітетами, такими як різноманітні системи або ролі користувачів, та дозволяє визначити, як ці елементи інтегруються та взаємодіють з основною системою.

Завдяки діаграмі випадків використання стає зрозуміліше, які конкретні функції чи операції системи є необхідними для задоволення потреб користувачів. Вона допомагає уточнити та відобразити специфічні дії та процеси, які система повинна підтримувати, щоб ефективно функціонувати та відповідати вимогам користувачів.

Опис діаграми прецедентів рис 2.2.1 для бою гравця та майстра гри:

1. Майстер гри або гравець починає бій.
2. Відбувається визначення порядку ходу діючих персонажів.
3. Активному користувачу пропонують вибирати одну з дій до моменту виснаження балів дій:
    a. відступити – персонаж витрачає свої бали дій для втечі;
    b. використати предмет, атакувати персонаж атакує суперника або використовує предмет на себе;
        - вибравши дію необхідно вибрати ціль взаємодії;
        - для перевірки на успіх необхідно кинути кубик;
        - отриманий результат обчислюється на основі характеристик иа навичок персонажа, цілі та предмета чи зброї;
        - характеристики обох сторін оновлюються згідно з результатом.
    c. перезарядитись;
    d. активувати уміння, зачекати.
4. незалежно від вибору, користувач вимушений передати хід наступному персонажу згідно порядку ходу;
5. бій відбувається до моменту відсутності активних учасників однієї зі сторін.

5.3 Діаграма класів

У міжгалузевому інженерії, діаграма класів (рис 3.3), яка є елементом мови моделювання UML (Unified Modeling Language), є статичною діаграмою

структури, яка описує конфігурацію системи. Вона розкриває класи системи, їх атрибути, операції (або методи) та зв'язки між об'єктами [8].

Для початку побудови діаграми класів необхідно розробити класи з методами та атрибутами.

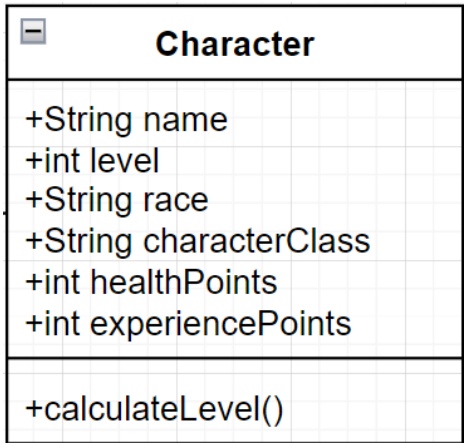


Рисунок 5.3.1 – Клас Character

Клас Character (рис. 2.3.1) - Представляє персонажа в грі з атрибутами, такими як ім'я, рівень, раса, клас персонажа, кількість очок здоров'я та досвіду. Включає методи, такі як calculateLevel().

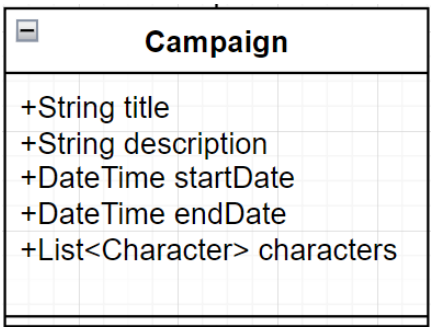


Рисунок 5.3.2 – Клас Campaign

Клас Campaign (рис. 2.3.2) - Представляє кампанію з атрибутами, такими як заголовок, опис, дати початку та завершення, і список персонажів. Діє як контейнер для кількох персонажів.

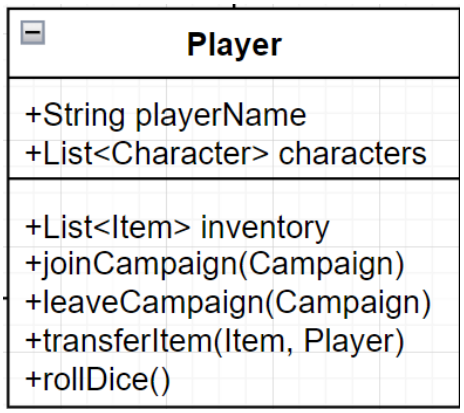


Рисунок 5.3.3 – Клас Player

Клас Player (рис. 2.3.3) - Представляє гравця в грі. Включає атрибути, такі як ім'я гравця, список персонажів та інвентар предметів. Основні методи включають joinCampaign(), leaveCampaign(), transferItem() та rollDice(). Має взаємозв'язок з класами Персонажа та Предмета, і використовує класи Dice та GameConnection.

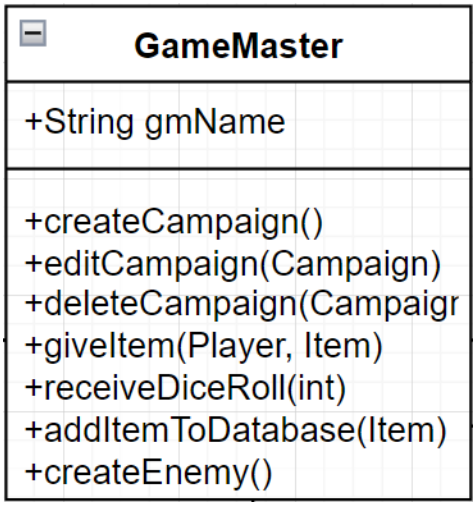


Рисунок 5.3.4 – Клас GameMaster

Клас GameMaster (рис. 2.3.4) - Представляє ведучого гри з методами для створення, редагування та видалення кампаній, видавання предметів гравцям, отримання результатів кидка костей, додавання предметів до бази даних та створення ворогів. Керує кампаніями та має можливість оновлювати DatabaseManager та створювати Ворогів.

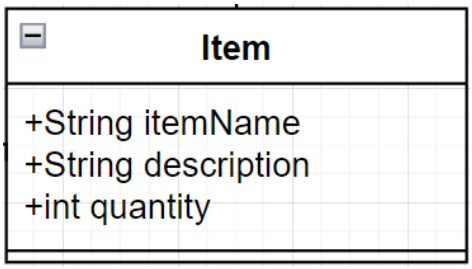


Рисунок 5.3.5 – Клас Item

Клас Item (рис. 2.3.5) - Представляє предмети в грі з атрибутами, такими як назва предмета, опис та кількість.

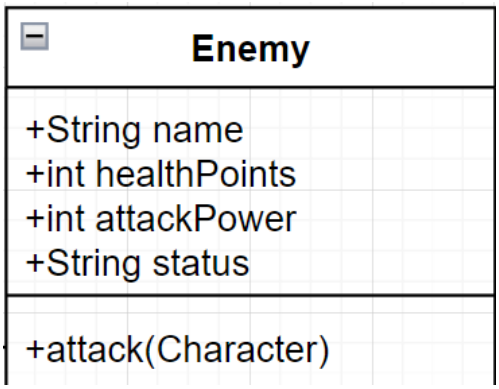


Рисунок 5.3.6 – Клас Enemy

Клас Enemy (рис. 2.3.6) - Представляє ворожого персонажа з атрибутами, такими як ім'я, кількість очок здоров'я, потужність атаки та стан. Включає метод для атаки.

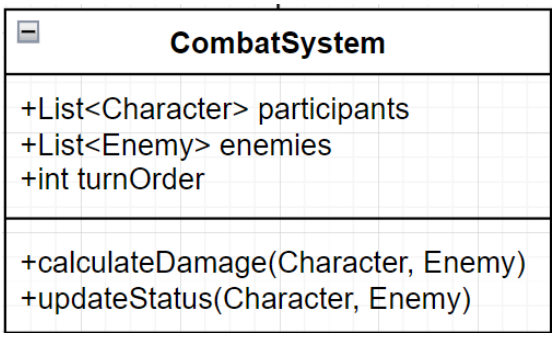


Рисунок 5.3.7 – Клас CombatSystem

Клас CombatSystem (рис. 2.3.7) - Керує боєм в грі зі списком учасників - персонажів та ворогів та порядком ходу. Включає методи для розрахунку пошкоджень та оновлення стану.

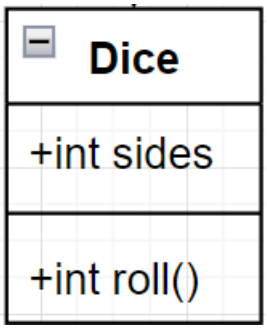


Рисунок 5.3.8 – Клас Dice

Клас Dice (рис. 2.3.8) який представляє кість у грі та включає метод для її кидання, приймає типи кубів та методи їх кидання. Створений щоб зробити процес більш комфортним для гравця та зручним у використанні.

Класи мають певний рівень взаємодії один з одним, виявлене у відносинах, що представлені у діаграмах класів. Ці зв'язки включають різні типи логічних зв'язків. Зв’язки між класами наступні:

1. Player – Character: Кожен гравець прямо пов'язаний зі своїм персонажем у грі. Це означає, що гравець керує діями та рішеннями свого персонажа.
2. GameMaster – Campaign: Майстер гри відповідає за розробку та ведення ігрової кампанії, створюючи сценарій, сюжет та задачі для гравців.
3. Campaign – Character: Кожна кампанія містить багато персонажів, які беруть участь у сюжетних подіях та завданнях, створених майстром гри.
4. Player – Item: Гравці володіють різними предметами (зброя, зілля, екіпірування тощо), які використовуються їхніми персонажами під час гри.
5. GameMaster – DatabaseManager: Майстер гри використовує менеджера бази даних для зберігання та організації інформації про кампанію, персонажів та інші ігрові елементи.
6. GameMaster – Enemy: Майстер гри відповідає за створення ворогів та антагоністів у кампанії, з якими зустрічаються гравці.

7. CombatSystem – Character: Бойова система безпосередньо пов'язана з персонажами, визначаючи правила та механізми бою, в яких беруть участь персонажі.
8. CombatSystem – Enemy: Так само, як і з персонажами, бойова система регулює взаємодію та бої між персонажами гравців та ворогами.
9. Player – Dice: Гравці використовують гральні кістки для визначення результатів рішень та дій їхніх персонажів.
10. GameMaster – Dice: Майстер гри також використовує кістки для визначення результатів та наслідків подій у кампанії.
11. Player – GameConnection: Гравці використовують ігровий зв'язок для входу в ігрову сесію та взаємодії з іншими учасниками.
12. GameMaster – GameConnection: Майстер гри керує ігровим зв'язком, контролюючи події, що відбуваються в ігровому середовищі, та координуючи взаємодію між гравцями.

Для обраних класів та їх відносин побудував діаграму класів рис 2.3.9

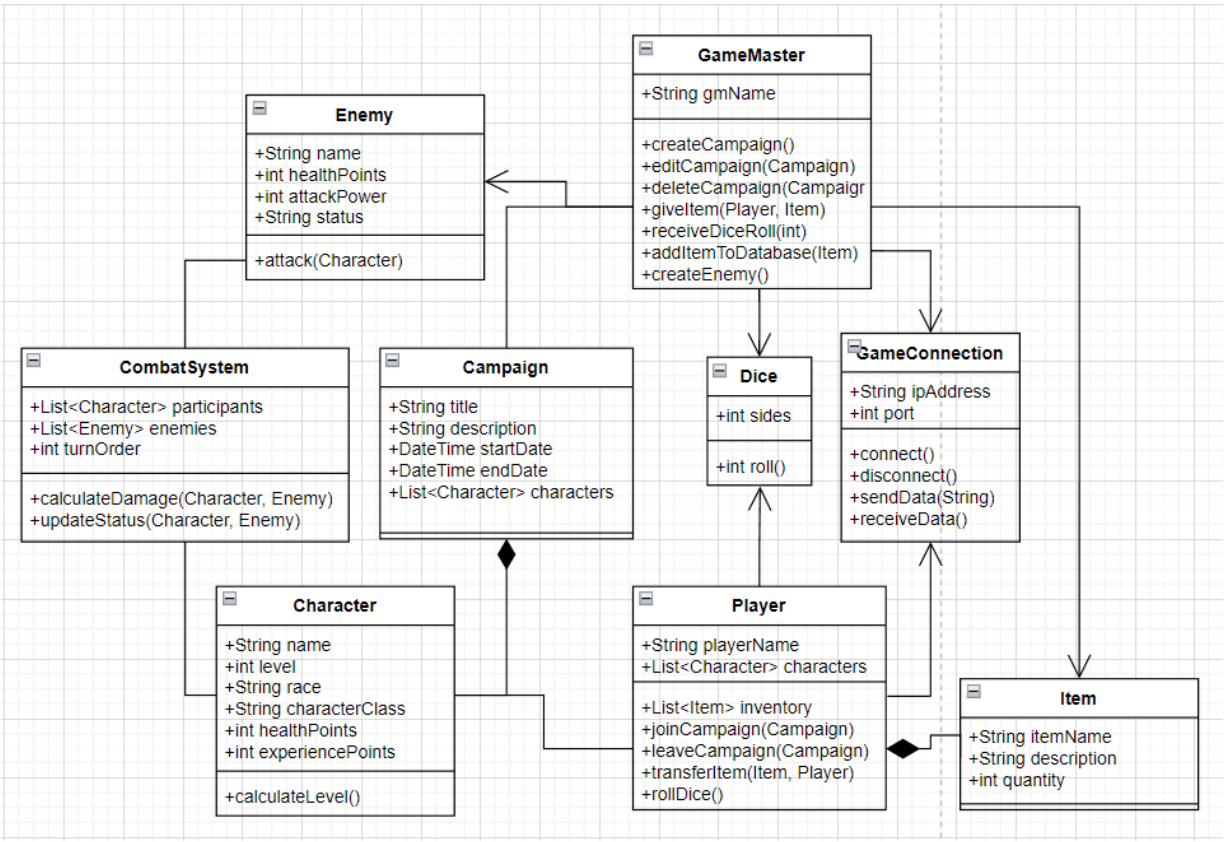


Рисунок 5.3.9 – Діаграма класів розроблюваного програмного продукту

5.4 Діаграма послідовності

Діаграми послідовності UML (рис. 2.4.1) є зображеннями, що показують процес виконання операцій. Вони наочно відображають, як об'єкти взаємодіють у колаборативному контексті. Ці діаграми базуються на часі, демонструючи послідовність взаємодій, візуально упорядковуючи надіслані повідомлення та їх часові рамки вздовж вертикальної вісі, що відображає час [9].

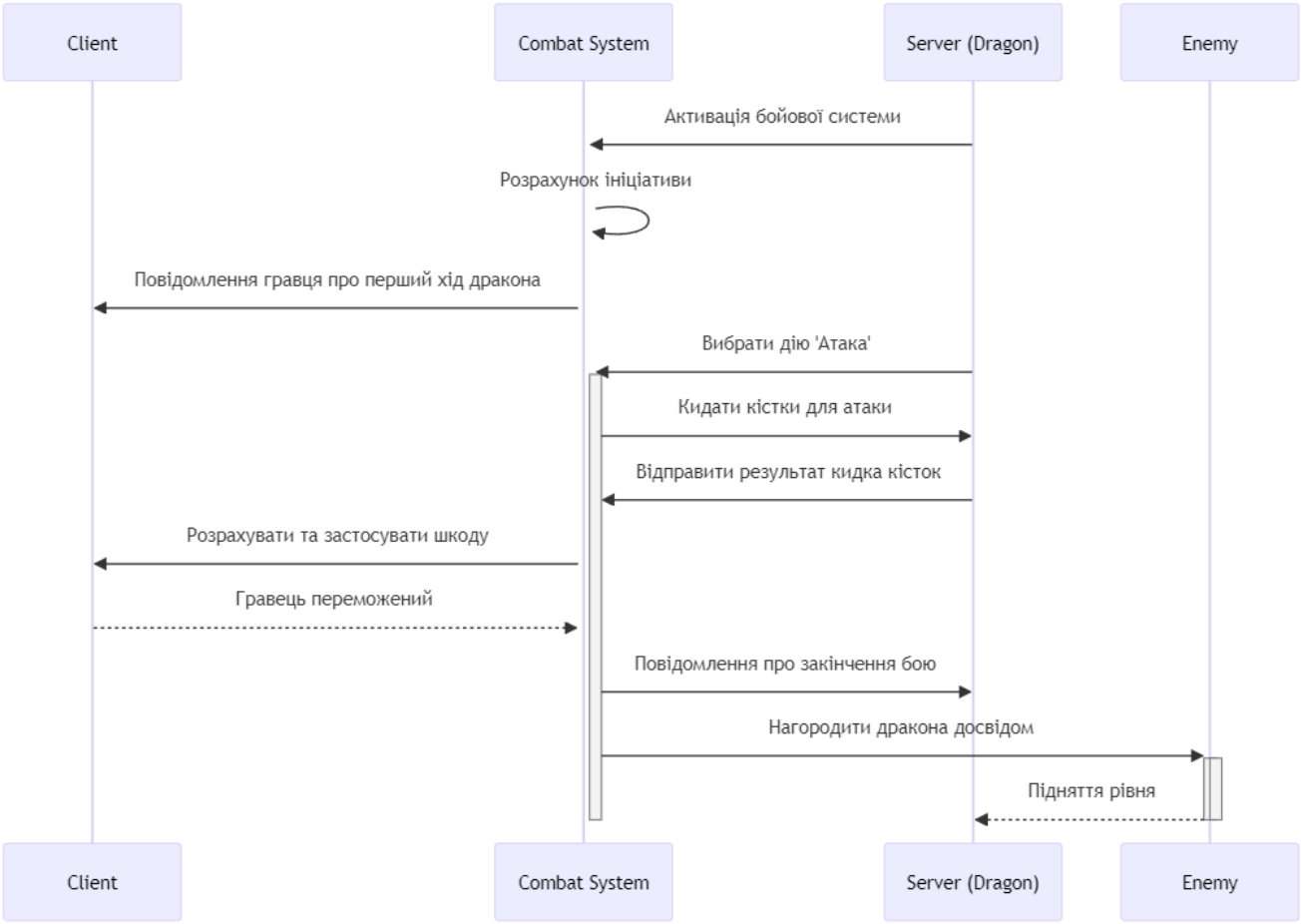


Рисунок 5.4.1 – Діаграма послідовності сутички персонажа з сильним суперником

Відтворив діаграму послідовності для проведення бою гравця низького рівня з драконом зі сторони ігрового майстра на рис. 2.4.1 Згідно діаграми бачимо що дракон напав на гравця, розпочав бій і почав роботу класу Combat. Клас Combat розрахував ініціативу гравця нижчою ніж ініціативу дракона і надав

дракону перший хід одночасно попередивши грався що відбулося. Ігровий майстер вибрав дію «Атака» і кинув кубики, результат кидка передався класу Combat де він розрахував силу атаки. Передав пошкодження гравцю, котрий не мав можливості пережити отриману шкоду і загинув. Повернувши дані клас Combat повідомив сторони про завершення бою та нагородив дракона досвідом, чим активував клас Leveling System і підняв собі рівень.

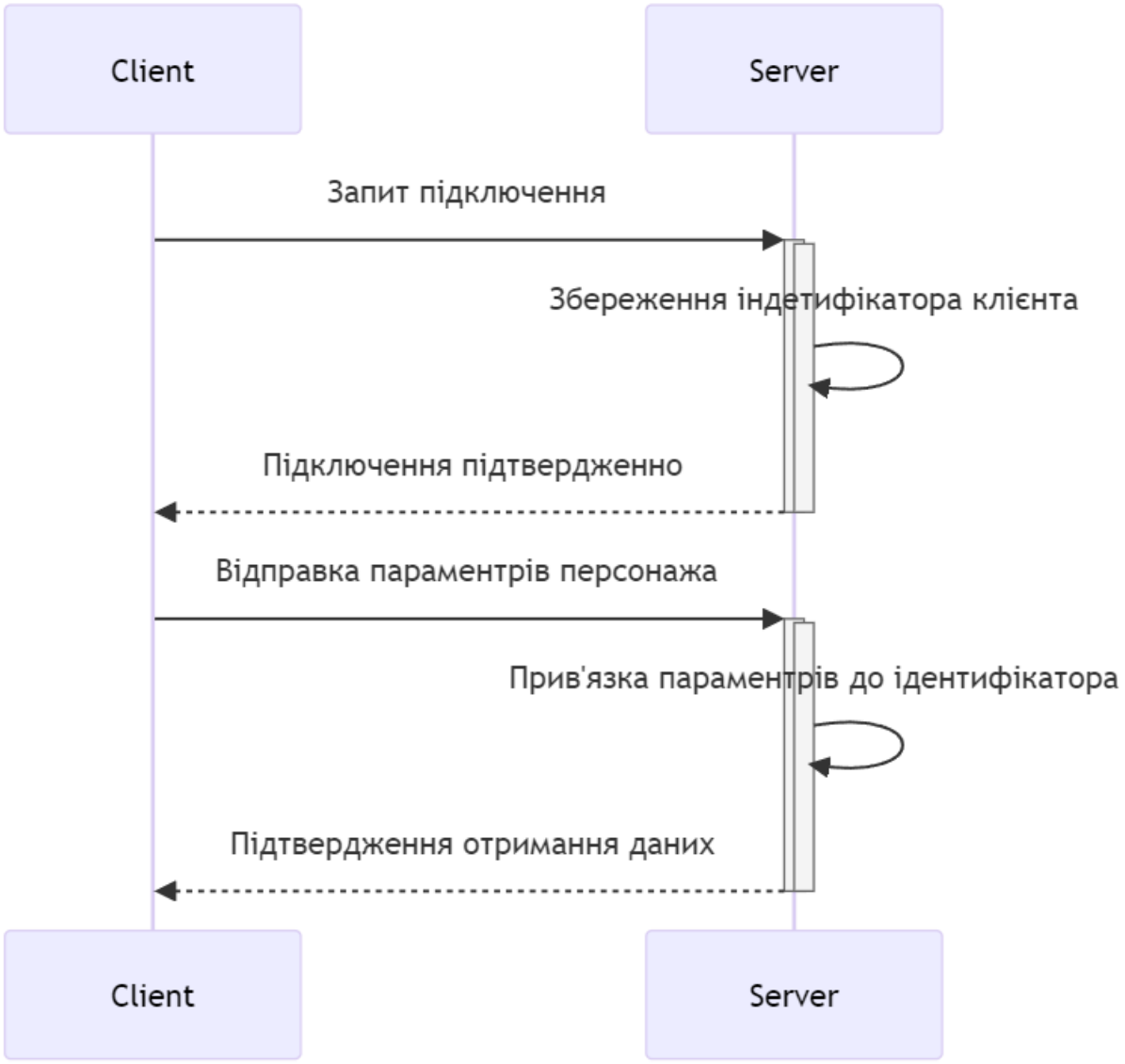


Рисунок 5.4.2 – Діаграма послідовності взаємодії клієнта з сервером

Діаграма послідовності (рис. 2.4.2) відображає процес реєстрації клієнта на сервері та подальшої передачі даних про ігрового персонажа від клієнта.

6 КОНСТРУЮВАННЯ КЛІЄНТ СОКЕТНОГО ДОДАТКУ

6.1 Розгляд процесу дії сервера

Сервер виконує кілька ключових функцій:

1. Ініціалізація та Запуск Сервера

```
24      public SocketServer()
25      {
26          listener = new TcpListener(IPAddress.Any, 8080);
27      }
        2 references
28      public void Start()
29      {
30          isRunning = true;
31          listener.Start();
32          ListenForClients();
33      }
```

Рисунок 6.1.1 – Ініціалізація сервера

Спочатку сервер створює об'єкт TcpListener (рис. 3.1.1), вказуючи IP-адресу та порт, на якому він буде слухати вхідні з'єднання. Виклик Start() запускає процес прослуховування. Сервер запускає безкінечний цикл, що дозволяє постійно приймати нові з'єднання.

```
        1 reference
40      private async void ListenForClients()
41      {
42          try
43          {
44              while (isRunning)
45              {
46                  TcpClient client = await listener.AcceptTcpClientAsync();
47                  clients.Add(client);
48                  ProcessClient(client);
49              }
50          }
51          catch (Exception ex)
52          {
53              errors = ex.ToString();
54          }
55      }
```

Рисунок 6.1.2 – Початок прослуховування вхідних підключень

Коли клієнт намагається підключитися до сервера, TcpListener приймає це з'єднання за допомогою методу AcceptTcpClientAsync() (рис. 3.1.2). Це створює новий об'єкт TcpClient. Сервер зберігає об'єкт TcpClient у списку, що дозволяє легко керувати всіма підключеними клієнтами.

```
1 reference
57  private async void ProcessClient(TcpClient client)
58  {
59      var stream = client.GetStream();
60      byte[] buffer = new byte[1024];
61      int bytesRead;
62
63      try
64      {
65
66          while ((bytesRead = await stream.ReadAsync(buffer, 0, buffer.Length)) != 0)
67          {
68              // Отримання повідомлення від клієнта
69              reader = Encoding.UTF8.GetString(buffer, 0, bytesRead);
70              // Виконання команди
71              HandleServerCommand(reader);
72              // Відправка відповіді клієнту
73              writer = "Message received " + reader;
74              byte[] data = Encoding.UTF8.GetBytes(writer);
75              await stream.WriteAsync(data, 0, data.Length);
76          }
77      }
78      catch (Exception ex)
79      {
80          errors = ex.ToString();
81      }
82      finally
83      {
84          client.Close();
85      }
86  }
```

Рисунок 6.1.3 – Обробка вхідних повідомлень та відправка підтвердження

У методі ProcessClient (рис. 3.1.3), сервер читає дані, відправлені клієнтом, і може відправити відповідь. Сервер має можливість відправляти повідомлення окремим клієнтам. Він використовує індекс зі списку клієнтів для визначення, кому відправляти повідомлення.

6.2 Опис коду клієнта

```
1 reference
public async void ConnectToServer(string serverIp)
{
    try
    {
        isRunning = true;
        client = new TcpClient();
        await client.ConnectAsync(serverIp, 8080);
        stream = client.GetStream();

        ListenToServer();
    }
    catch (Exception ex)
    {
        errors = $"Connection error: {ex.Message}";
    }
}
```

Рисунок 6.2.1 – Метод підключення до сервера

Клієнт ініціалізує об'єкт TcpClient, вказуючи IP-адресу та порт сервера, до якого він хоче підключитися (рис 3.2.1). Це здійснюється через функцію ConnectToServer. Коли клієнт створюється, він автоматично намагається встановити з'єднання з сервером за переданим адресом. Після успішного підключення до сервера, клієнт виконує ListenToServer. Цей процес використовується для прийому даних.

```
private async void ListenToServer()
{
    try
    {
        byte[] buffer = new byte[1024];
        while (isRunning)
        {
            while (client.Connected)
            {
                int bytesRead = await stream.ReadAsync(buffer, 0, buffer.Length);
                if (bytesRead > 0)
                {
                    string response = Encoding.UTF8.GetString(buffer, 0, bytesRead);
                    Device.BeginInvokeOnMainThread(() =>
                    {
                        reader = response;
                        HandleServerCommand(response);
                    });
                }
            }
        }
    }
    catch (Exception ex)
    {
        Device.BeginInvokeOnMainThread(() =>
        {
            errors = $"Read error: {ex.Message}";
        });
    }
}
```

Рисунок 6.2.2 – Метод очікування повідомлення від сервера

Клієнт створює асинхронний потік (async), який безперервно слухає вхідні повідомлення від сервера. Цей метод безперервно чекає на вхідні повідомлення від сервера (рис. 3.2.2). Коли повідомлення надходять, він читає байти і перетворює їх назад у рядок, який відображається в консолі.

```
48      public async void Send(string message)
49      {
50          byte[] data = Encoding.UTF8.GetBytes(message);
51
52          await stream.WriteAsync(data, 0, data.Length);
53
54          // Очікування відповіді від сервера
55          byte[] buffer = new byte[1024];
56          int bytes = await stream.ReadAsync(buffer, 0, buffer.Length);
57          string response = Encoding.UTF8.GetString(buffer, 0, bytes);
58          HandleServerCommand(response);
59      }
```

Рисунок 6.2.3 – Метод відправлення команди

Клієнт може відправляти повідомлення на сервер, використовуючи метод Send (рис. 3.2.3). Він кодує рядок у байти і відправляє їх через WriteAsync. Одразу отримує відповідь від сервера і обробляє її.

6.3 Опис MessagingCenter

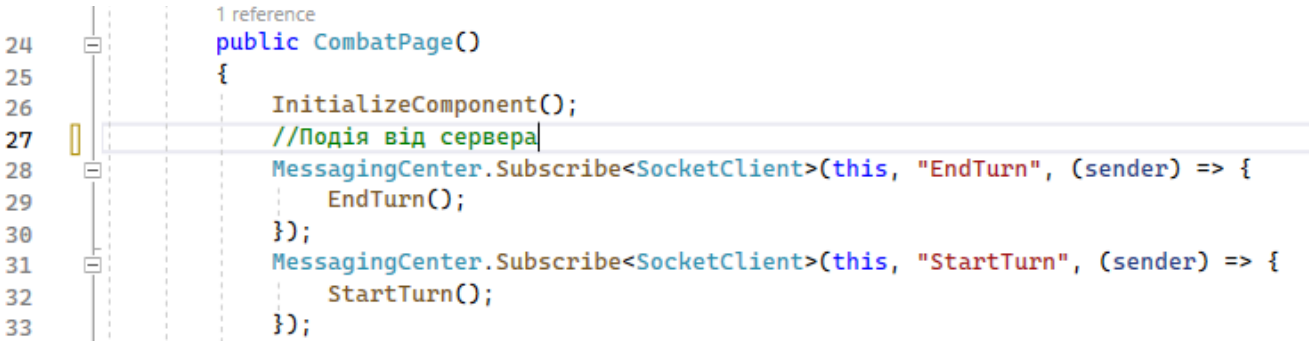
MessagingCenter у Xamarin.Forms — це механізм для обміну повідомленнями між різними частинами додатку без необхідності їх прямого зв'язування. Це корисно для зниження залежностей між компонентами та підвищення гнучкості коду. MessagingCenter має три основні компоненти:

Send (рис. 3.3.1) -Об'єкти, які ініціюють відправлення повідомлень. Коли потрібно сповістити інші частини додатку про якусь подію, відправник використовує MessagingCenter.Send(). Цей метод може включати додаткові дані, які потрібно передати.

```
10  namespace App6
11  {
        9 references
12      public class SocketClient
13      {
14          //Відправка повідомлень по клієнту
15          public event Action<string> OnMessageReceived;
            1 reference
16          public void HandleServerCommand(string command)
17          {
18              if (command == "EndTurn")
19              {
20                  MessagingCenter.Send(this, "EndTurn");
21              }
22              if (command == "StartTurn")
23              {
24                  MessagingCenter.Send(this, "StartTurn");
25              }
```

Рисунок 6.3.1 – Відправлення повідомлення з корінного простору додатку

Subscribe (рис. 3.3.2) - Об'єкти, які слухають та реагують на певні повідомлення. Частини додатку, які хочуть реагувати на певні повідомлення, реєструються через MessagingCenter.Subscribe(). Вони вказують, на які повідомлення вони хочуть підписатися і який метод викликати при їх отриманні.



```
            1 reference
24          public CombatPage()
25          {
26              InitializeComponent();
27              //Подія від сервера
28              MessagingCenter.Subscribe<SocketClient>(this, "EndTurn", (sender) => {
29                  EndTurn();
30              });
31              MessagingCenter.Subscribe<SocketClient>(this, "StartTurn", (sender) => {
32                  StartTurn();
33              });
```

Рисунок 6.3.2 – Прослуховування та опрацювання команди від серверу

Unsubscribe (рис. 3.3.3) - Щоб уникнути витоків пам'яті або небажаних сповіщень, підписники можуть відписатися від повідомлень за допомогою MessagingCenter.Unsubscribe(), коли це необхідно.

```
53          //Зупинка слідкування за клієнтом
            0 references
54          protected override void OnDisappearing()
55          {
56              base.OnDisappearing();
57              MessagingCenter.Unsubscribe<SocketClient>(this, "EndTurn");
58              MessagingCenter.Unsubscribe<SocketClient>(this, "StartTurn");
59          }
```

Рисунок 6.3.3 – Зупинка прослуховування повідомлень

Недостатком вжитого механізму є необхідність правильно керувати підписками та відписками, щоб уникнути витоків пам'яті та інших проблем, а також надмірне використання MessagingCenter може ускладнити розуміння потоку даних у додатку з пошуком кінцевого адресата.

6.4 Опис переходів між сторінками

Для уникнення відключення серверу після згортання сторінки з його створенням оголосив клас серверу та клієнту в корінному файлі App під час запуску додатку (рис 3.3.1). Також сторінки оголошені як NavigationPage що є контейнером, який забезпечує можливість навігації між сторінками, включаючи стек навігації та відображення кнопки "Назад".

```
            32 references
8           public partial class App : Application
9           {
                15 references
10              public static ObservableCollection<Character> GlobalCharacterSquad { get; set; }
11              public static SocketServer socketServer;
12              public static SocketClient socketClient;
13              public string ipadress;
14
                1 reference
15              public App()
16              {
17                  InitializeComponent();
18                  GlobalCharacterSquad = new ObservableCollection<Character>();
19                  socketServer = new SocketServer();
20                  socketClient = new SocketClient();
21                  MainPage = new NavigationPage(new MainPage());
22
```

Рисунок 6.4.1 – Оголошення глобальних переміних

6.5 Опис CombatPage

На сторінці CombatPage (рис. 6.5.1), яка є частиною додатку розробленого для відтворення сценаріїв бою в рольовій грі (наприклад, Dungeons & Dragons), відбуваються наступні ключові дії та інтерактивні процеси:

```
<!-- Characters List -->
<Label Text="Characters" FontAttributes="Bold"/>
<ListView x:Name="CharactersListView"
          ItemsSource="{Binding Characters}" ItemSelected="CharactersListView_ItemSelected">
    <ListView.ItemTemplate>
        <DataTemplate>
            <TextCell Text="{Binding Name}" Detail="{Binding Health}" />
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>

<!-- Fighters List -->
<Label Text="Fighters" FontAttributes="Bold"/>
<ListView x:Name="FightersListView" ItemsSource="{Binding Fighters}" ItemSelected="FightersListView_ItemSelected">
    <ListView.ItemTemplate>
        <DataTemplate>
            <ViewCell>
                <StackLayout Orientation="Horizontal">
                    <Label Text="{Binding Name}" HorizontalOptions="StartAndExpand"/>
                    <Label Text="{Binding Health}" HorizontalOptions="Center"/>
                    <Label Text="{Binding Initiative}" HorizontalOptions="End"/>
                </StackLayout>
            </ViewCell>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

Рисунок 6.5.2 – Xaml розмітка сторінки

Сторінка показує список Fighters і Characters які беруть участь у бою. Кожен боєць представлений своїм ім'ям, рівнем здоров'я та рівнем ініціативи.

```
2 references
public ObservableCollection<Character> Characters { get; set; }
2 references
public ObservableCollection<Fighter> Fighters { get; set; }
private Fighter selectedFighter;
private Character selectedCharacter;

2 references
public CombatPage()
{
    InitializeComponent();
    //Подія від сервера
    MessagingCenter.Subscribe<SocketClient>(this, "EndTurn", (sender) => {
        EndTurn();
    });
    MessagingCenter.Subscribe<SocketClient>(this, "StartTurn", (sender) => {
        StartTurn();
    });
    MessagingCenter.Subscribe<SocketClient>(this, "LoadFighters", (sender) => {
        LoadFighters();
    });

    Fighters = new ObservableCollection<Fighter>();
    Characters = new ObservableCollection<Character>();
    Characters = App.GlobalCharacterSquad;
    BindingContext = this;
    App.socketClient = new SocketClient();
}
```

Рисунок 6.5.3 – Ініціалізація класу CombatPage

Під час ініціалізації сторінки відбувається завантаження наявних гравців з класу GlobalCharacterSquad на сторінках клієнтів тим часом поки сервер вибирає Fighters для бою та відправляє їх з командою LoadFighters (рис. 6.5.4).

Гравці можуть вибрати дії для своїх персонажів, такі як атака (рис. 6.5.5), оборона, використання спеціальних здібностей або предметів. Взаємодія здійснюється через кнопки дії на сторінці. Залежно від обраної дії, може відбутися розрахунок результату (наприклад, успіх або невдача атаки), який визначається правилами гри.

```
60          private void FightersListView_ItemSelected(object sender, SelectedItemChangedEventArgs e)
61          {
62              selectedFighter = e.SelectedItem as Fighter;
63              ((ListView)sender).SelectedItem = null;
64          }
            0 references
65          private void CharactersListView_ItemSelected(object sender, SelectedItemChangedEventArgs e)
66          {
67              selectedCharacter = e.SelectedItem as Character;
68              ((ListView)sender).SelectedItem = null;
69          }
            0 references
70          private async void OnAttackClicked(object sender, EventArgs e)
71          {
72              if (selectedFighter != null && selectedFighter.Health > 0)
73              {
74                  int dmg = 0;
75                  await Navigation.PushAsync(new DiceRollPage("Strength", ref dmg));
76                  selectedFighter.Health -= dmg;
77                  statusLabel.Text = $"Боєць {selectedFighter.Name} атакований. Залишилось здоров'я: {selectedFighter.Health}";
78
79                  if (selectedFighter.Health <= 0)
80                  {
81                      statusLabel.Text += $"\nБоєць {selectedFighter.Name} переможений!";
82                  }
83                  FightersListView.ItemsSource = null;
84                  FightersListView.ItemsSource = Fighters;
85                  var socketServer = new SocketServer();
86              }
```

Рисунок 6.5.6 – Метод нанесення шкоди персонажу

Після кожного ходу стан гри оновлюється. Це може включати зміни в рівнях здоров'я бійців, статусах ініціативи чи інших важливих факторах. Оновлена інформація відображається на екрані завдяки Binding (рис 6.5.7), щоб гравці могли бачити поточний стан бою та планувати свої наступні кроки.

6.6 CreateCharacter

CreateCharacter – це клас що використовується для створення нового персонажа у додатку. Ця сторінка дає можливість користувачеві ввести початкові атрибути та характеристики для свого персонажа.

При створенні сторінки виконується ініціалізація компонентів інтерфейсу користувача, заданих у XAML (рис. 6.6.1).

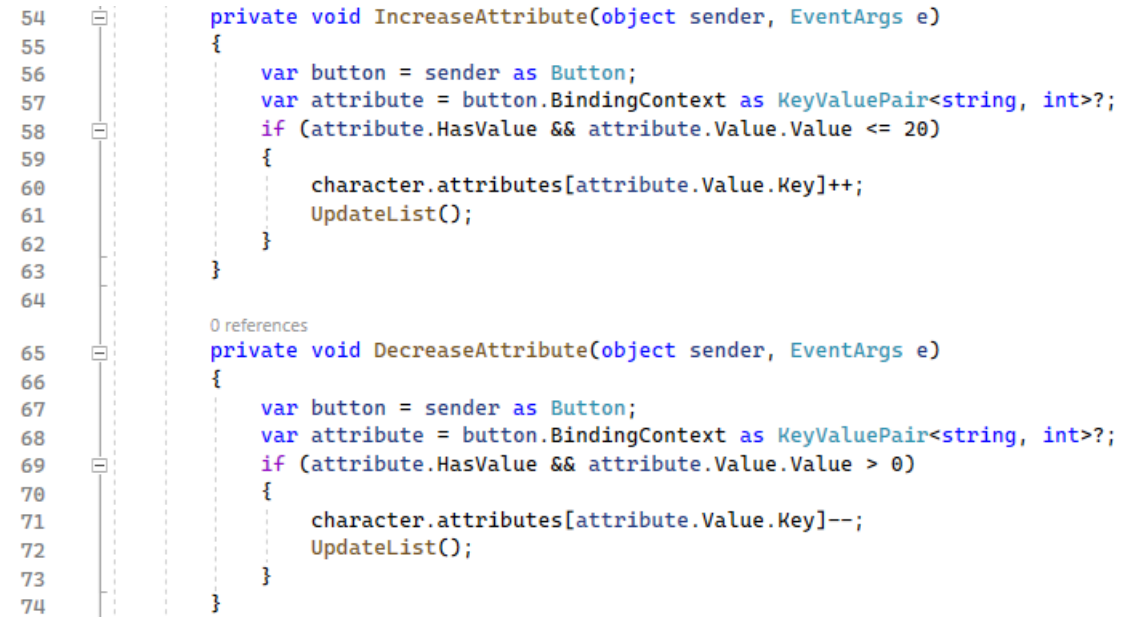
```
<ScrollView>
    <StackLayout Padding="20">
        <Entry Placeholder="Name" x:Name="NameEntry" />
        <Picker Title="Race" x:Name="RacePicker">
        </Picker>
        <Picker Title="Class" x:Name="ClassPicker">
        </Picker>
        <Entry Placeholder="Total Points" Keyboard="Numeric" x:Name="TotalPointsEntry" TextChanged="OnTotalPointsChanged"/>
        <ListView x:Name="AttributesListView">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <Grid Padding="10">
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="Auto"/>
                                <ColumnDefinition Width="*"/>
                                <ColumnDefinition Width="Auto"/>
                            </Grid.ColumnDefinitions>

                            <Button Text="Minus" Clicked="DecreaseAttribute" Grid.Column="0"/>
                            <StackLayout Orientation="Horizontal" HorizontalOptions="CenterAndExpand" Grid.Column="1">
                                <Label Text="{Binding Key}" HorizontalOptions="Start"/>
                                <Label Text="{Binding Value}" HorizontalOptions="End"/>
                            </StackLayout>
                            <Button Text="Plus" Clicked="IncreaseAttribute" Grid.Column="2"/>
                        </Grid>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
```

Рисунок 6.6.2 – XAML розмітка інтерфейсу

Користувач може вводити інформацію про персонажа, таку як ім'я, клас, расу, а також розподіляти параметри характеристик, наприклад сила, спритність, інтелект. Ці дані вводяться через текстові поля і вибираються з випадаючих списків. З допомогою Binding дані синхронізуються між інтерфейсом користувача та моделлю даних, автоматично оновлюючи в випадку зміни вказаного параментру.

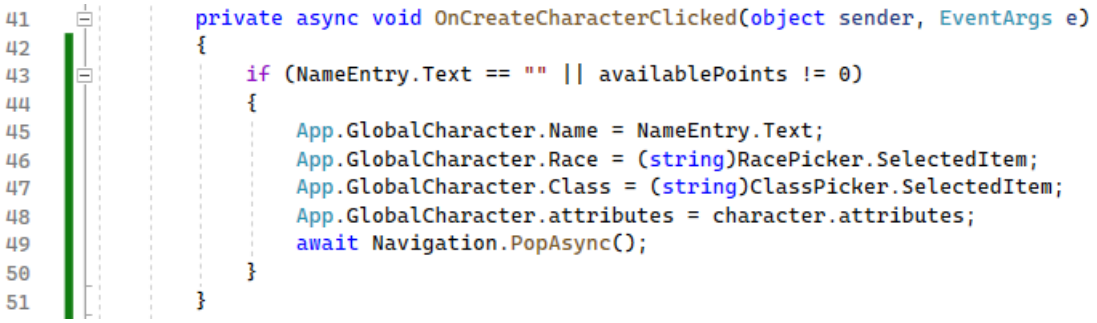
При зміні параметрів персонажа методи DecreaseAttribute і IncreaseAttribute (рис. 6.6.3) відслідковують суму характеристик персонажа та не позволяє їй вийти за межі області $0<x<21$ та перевищити максимально вказану кількість параментрів за допомоги.



```
54      private void IncreaseAttribute(object sender, EventArgs e)
55      {
56          var button = sender as Button;
57          var attribute = button.BindingContext as KeyValuePair<string, int>?;
58          if (attribute.HasValue && attribute.Value.Value <= 20)
59          {
60              character.attributes[attribute.Value.Key]++;
61              UpdateList();
62          }
63      }
64
        0 references
65      private void DecreaseAttribute(object sender, EventArgs e)
66      {
67          var button = sender as Button;
68          var attribute = button.BindingContext as KeyValuePair<string, int>?;
69          if (attribute.HasValue && attribute.Value.Value > 0)
70          {
71              character.attributes[attribute.Value.Key]--;
72              UpdateList();
73          }
74      }
```

Рисунок 6.6.4 – Методи зміни характеристик персонажа

Перед збереженням або використанням даних персонажа, програма перевіряє введену інформацію на коректність та повноту. Програма переконується, що всі обов'язкові поля заповнені та що розподілені очки характеристик не перевищують встановлених лімітів.



```
41      private async void OnCreateCharacterClicked(object sender, EventArgs e)
42      {
43          if (NameEntry.Text == "" || availablePoints != 0)
44          {
45              App.GlobalCharacter.Name = NameEntry.Text;
46              App.GlobalCharacter.Race = (string)RacePicker.SelectedItem;
47              App.GlobalCharacter.Class = (string)ClassPicker.SelectedItem;
48              App.GlobalCharacter.attributes = character.attributes;
49              await Navigation.PopAsync();
50          }
51      }
```

Рисунок 6.6.5 – Валідація введених даних

Після введення та валідації інформації, дані про нового персонажа введені дані зберігаються локально у додатку і відправляються на сервер після наступного підключення до нової сесії гри. Після створення персонажа, користувач буде перенаправлений на попередню сторінку.

6.7 DiceRollPage

DiceRollPage - використовується для того щоб розраховувати та відображати анімований кидок грального кістки. Анімація додає візуального враження до ігрового процесу. Вона робить дію кидка кістки більш динамічною та цікавою, особливо у цифрових версіях гри, де фізичного взаємодії з кістками немає. Анімація може допомогти гравцям відчути більшу присутність та залученість у гру.

```
15      private Random random = new Random();
        2 references
16      public DiceRollPage(string Attribute, ref int result)
17      {
18          InitializeComponent();
19          StartDiceAnimation();
20          result = random.Next(1, 21);
21          DiceResult.Text = "" + result;
22          ABonus.Text = $"Бонус {Attribute}: " + CalcStatBonus(App.GlobalCharacter.attributes[Attribute]);
23      }
```

Рисунок 6.7.1 - Реалізація класу DiceRollPage

Під час переходу на сторінку відбувається ініціалізація кидка кістки та відображення результату (рис. 6.7.2). Результат визначається для D20 кістки та зберігається за посиланням перемінної котра опрацьовується на сторінці виклику.

```
24      private async void StartDiceAnimation()
25          {
26              DiceImage.Opacity = 1;
27              var rotateTask = DiceImage.RotateTo(360, 1000);
28              var jumpUpTask = DiceImage.TranslateTo(0, -100, 500, Easing.SinOut);
29              await Task.WhenAll(rotateTask, jumpUpTask);
30              await DiceImage.TranslateTo(0, 0, 500, Easing.SinIn);
31              DiceImage.Rotation = 0;
32              DiceResult.IsVisible = true;
33          }
```

Рисунок 6.7.3 – Представлення простої анімації методами C#

Анімація виконується переміщенням та обертанням зображення кубика після приземлення котрого поверх відображається результат кидка (рис. 6.7.4).

Подібний варіант представлення анімованої події є легким в модифікації в випадку якщо потрібно добавити нові кубики та змінити їх формат.

```
39          private decimal CalcStatBonus(int Atr)
40              {
41              decimal a = 2;
42              return Math.Floor((Atr - 10) / a);
43              }
```

Рисунок 6.7.5 – Метод розрахунку додаткових очок до кидка гральних кісток

Розрахунок бонусу від характеристик відбувається згідно з правилами оригінальної гри, де від обраного атрибуту віднімають десять та результат ділиться на два. Згідно даного рівняння гравець має можливість отримання бонусу в діапазоні -4<x<10, де x – є бонусом від характеристики персонажа (рис. 6.7.6).

6.8 TradingPage

TradingPage - це сторінка, призначена для взаємодії між персонажами або гравцями в цілях обміну внутрішньо ігровими предметами. Клас ініціліазується його викликом та передачі об’єктів Character (рис. 6.8.1).

```
16          public ObservableCollection<Item> Items { get; set; }
            4 references
17          public Character CurrentCharacter { get; set; }
            2 references
18          public Character OtherCharacter { get; set; }
19
            1 reference
20          public TradingPage(Character currentCharacter, Character otherCharacter)
21          {
22              InitializeComponent();
23              CurrentCharacter = currentCharacter;
24              OtherCharacter = otherCharacter;
25              Items = new ObservableCollection<Item>(CurrentCharacter.Inventory);
26              ItemsListView.ItemsSource = Items;
27          }
```

Рисунок 6.8.2 Ініціалізація класу TradingPage

На сторінці TradingPage відображається список предметів поточного персонажа котрі завантажуються з інвентаря персонажа. Гравці можуть вибирати предмети зі свого інвентаря, які вони хочуть передати до інвентаря іншого персонажа з допомогою OnTranssferClicked (рис. 6.8.3). Обраний перелік предметів

Поряд з кожним предметом може бути надана інформація, наприклад, назва, опис, характеристики або значення.

```
29          private void OnTransferClicked(object sender, EventArgs e)
30          {
31              foreach (var item in Items)
32              {
33                  if (item.IsSelected)
34                  {
35                      CurrentCharacter.Inventory.Remove(item);
36                      OtherCharacter.Inventory.Add(item);
37                  }
38              }
39              Items = new ObservableCollection<Item>(CurrentCharacter.Inventory);
40              ItemsListView.ItemsSource = Items;
41          }
42      }
```

Рисунок 6.8.4 Метод передачі обраних предметів

Після підтвердження обміну система передає обрані предмети об’єкту OtherCharacter і видаляє їх з переліку CurrentCharacter. Інвентарі обох сторін оновлюються для відображення нових предметів.

6.9 Тестування

Фінальним етапом розробки програмного забезпечення є тестування, процес, який перевіряє функціональність, надійність, безпеку та взаємодію програми з користувачем. Тестування включає виявлення помилок, відхилень від специфікацій та непередбачених проблем використання. Це допомагає гарантувати, що програмне забезпечення відповідає очікуванням користувачів та стандартам якості перед його випуском у продуктивне середовище.

Сторінка створення персонажа (рис. 6.9.1) містить поля для вводу імені персонажа і сумарної кількості характеристик. Поля для вибору раси і класу і перемикачі для збільшення та зменшення характеристик, після заповнення всіх полей і розподілення характеристик доступна кнопка створення персонажа

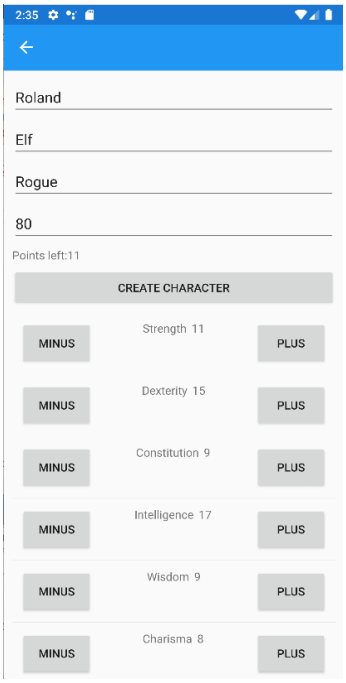


Рисунок 6.9.2 Вікно створення персонажа

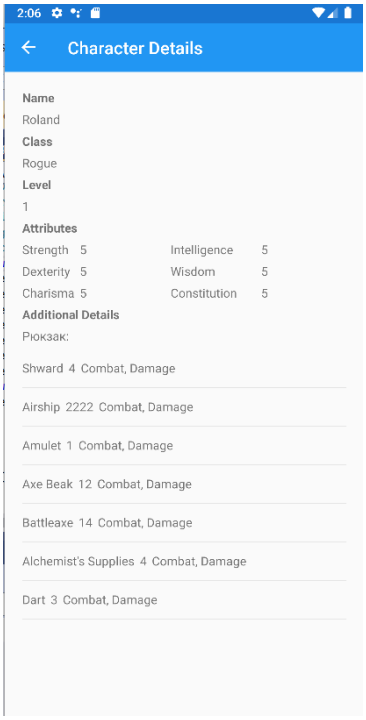


Рисунок 6.9.3 – Вікно параментрів персонажа

Сторінка перегляду характеристик (рис. 6.9.4) містить відображення всіх характеристик персонажа без можливості редагування. Вихід з сторінки відбувається за допомогою стрілки в верхньому лівому куті.

Сторінка передачі предметів (рис. 6.9.5) містить перелік речей в рюкзаку персонажа. Відображається назва предмета, його маса, і тип. Перемикачі біля предметів призначені для помітки предметів для передачі. Вибравши предмети і натиснувши кнопку передати, речі обрані переносяться іншому обраному персонажу.

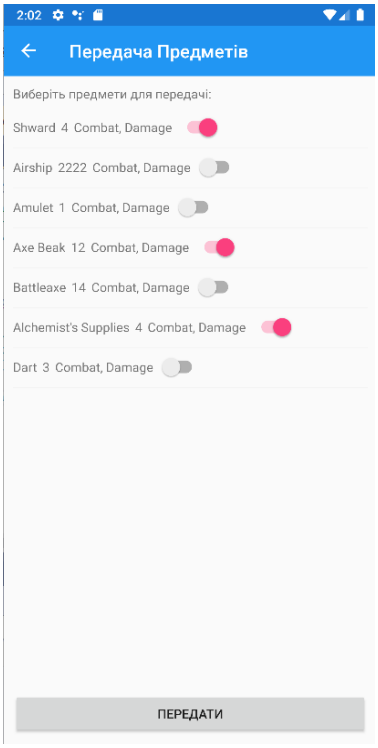


Рисунок 6.9.6 – Вікно передачі предметів

Сторінка CombatPage (рис. 6.9.7) містить списки персонажів і опонентів котрі відображають їхні характеристики. Сторінка має 4 кнопки, при нажатті котрих виконується певна дія.

1. Кнопка «ATTACK» і «USE SKILL» спрацьовують коли персонаж виділений, при спрацюванні відкривається сторінка Dice і відображається анімація кидка кубика. Після виходу з сторінки бачимо кількість шкоди

нанесеної цілі атаки, відмініть тільки в типі шкоди, а саме магічна чи фізична.

2. Кнопка «DEFEND» спрацьовує в випадку якщо інші кнопки не були використані, персонаж отримує половину шкоди до наступного ходу.
3. Кнопка «END TURN» завершує хід передаючи право ходу наступному персонажу, якщо ж походив останній персонаж в даному ході, хід передається опонентам під керуванням сервера.

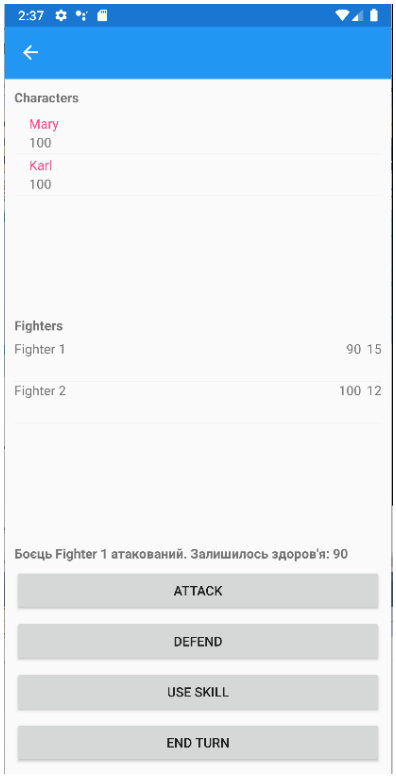


Рисунок 6.9.8 – Вікно бою

Сторінка Dice (рис. 6.9.9) при спрацюванні виконує анімацію кидка кубика D20 і висвічує його результат.

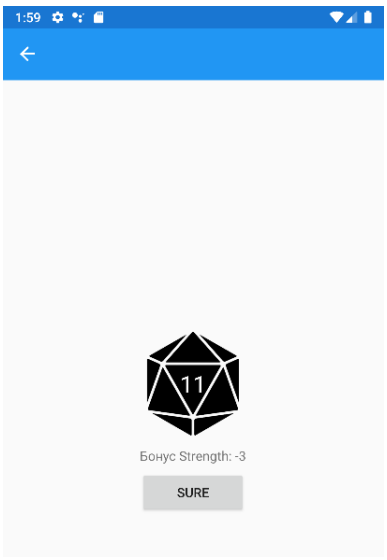


Рисунок 6.9.10 – Вікно кидка кісток

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТАЦІЯХ

7.1 Охорона праці

Зазвичай створення мобільного додатку є складним та вимагає багато часу, оскільки це включає роботу з різноманітними технологіями та обробку значного обсягу даних, що підвищує ризик зробити помилку. Всім відомо, що для підтримки ефективності роботи на комп’ютері важливо забезпечити належні умови праці. Недостатнє освітлення, обмежений простір чи низька якість дисплею можуть спричинити фізичний та з часом психологічний дискомфорт, який неодмінно позначиться на якості роботи. Щоб уникнути таких проблем, необхідно слідувати певним правилам та стандартам.

В наказі № 207 від 14.02.2018 НПАОП 0.00-7.15-18 [12], якраз описується частина вимог, яких потрібно дотримуватись при роботі з екранними пристроями.

Відповідно до третьої частини наказу [12], робоче місце працівника має відповідати наступним вимогам:

1. Робоче місце має мати достатні розміри, щоб працівник мав простір для зміни робочого положення та рухів.
2. Усе випромінювання від екранних пристроїв має бути знижене до гранично допустимого рівня з погляду безпеки та охорони здоров’я працівників.
3. Усі елементи робочого місця та їх розташування мають відповідати ергономічним, антропологічним, психофізіологічним вимогам, а також характеру виконуваних робіт.
4. Освітлення робочого місця має створювати відповідний контраст між екраном і навколишнім середовищем та відповідати вимогам ДСанПІН 3.3.2.007- 98.
5. Мікроклімат виробничих приміщень має підтримуватись на постійному рівні та відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [13].

6. Робочий стіл чи поверхня повинні бути достатнього розміру та мати поверхню з низькою відбивною здатністю, бути гнучкою під час розміщення екрана, клавіатури, документів чи устаткування.
7. Робоче крісло має бути стійким і дозволяти працівнику легко рухатися та займати зручне положення. Сидіння має регулюватися по висоті, спинка сидіння – по висоті, та з можливістю нахилу. Для зручності слід передбачати підніжку для тих, кому це необхідно.

Також потрібно пам’ятати про безпеку, відповідно до четвертої частини наказу [12], потрібно дотримуватись наступних мінімальних вимог безпеки:

1. Перед початком роботи необхідно очищати екранні пристрої від пилу та інших забруднень.
2. Після закінчення роботи пристрої слід відключати від живлення.
3. При виникнення аварійної ситуації необхідно в той же час відключити пристрій від електричної мережі.
4. Не допускається:
   – Ремонтувати чи виконувати технічне обслуговування, і налагодження екранних пристроїв на робочому місці працівника під час роботи з екранними пристроями;
   – Вимикати захисні пристрої чи проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження;
   – Працювати з несправними екранними пристроями, у яких під час роботи виникають нехарактерні сигнали, мигання та інші несправності.
5. Під час виконання робіт з комп’ютером, пов’язаних з нервово-емоційним напруженням мають дотримуватися оптимальні умови мікроклімату відповідно до вимог ДСН 3.3.6.042-99 [13].

Для забезпечення безпеки відповідно до п’ятої частини наказу «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [12], усі екранні пристрої повинні відповідати наступним мінімальним вимогам безпеки:

Відповідно до третьої частини наказу [12], робоче місце працівника розробника мобільного-додатку, який працює над контентним та функціональним наповненням додатку має відповідати наступним вимогам:

1. Екранні пристрої не мають бути джерелом ризику для працівників.
2. Усе випромінювання має бути зведене до мінімального рівня з погляду безпеки і охорони здоров'я працівників.
3. Символи на дисплеї мають бути чіткими, відповідного розміру. Між символами і рядками символів повинна бути правильна відстань.
4. Зображення на дисплеї має бути стабільним, без миготінь або інших видів несправності.
5. Яскравість та контрастність символів має легко регулюватися, а також швидко адаптуватися до навколишніх умов.
6. Під час вибору монітора, слід надавати перевагу тим пристроям, які мають можливості повороту та нахилу екрану.
7. При потребі монітор може бути закріпленим на окремому столі чи підставці.
8. При виборі монітора надавайте перевагу дисплеям з матовим покриттям, щоб мінімізувати відблискування або відбивання світла.
9. При виборі клавіатури, слід надавати перевагу тій, яка відкидається і є автономною, щоб працівник міг вибрати зручну робочу позу й уникнути втоми рук.
   - Поверхня клавіатури має бути матовою, щоб уникнути віддзеркалювання.
   - Устаткування, яке входить до робочої станції, не повинно виділяти надлишкового тепла.
   - Під час розробки, вибору, замовлення та модифікації програмного забезпечення, а також під час розробки завдань, що передбачають використання устаткування з екранними пристроями, роботодавець має керуватися таким програмним забезпеченням, яке відповідає

розв’язуваним завданням і є простим у використанні, а де необхідно - адаптованим до рівня знань і досвіду працівника.

Отже, для безпечної та ефективної роботи розробника програмного забезпечення забезпечено належні умови праці, починаючи від робочого місця та його оснащення, та закінчуючи мікрокліматом робочого середовища, відповідно до вимог чинного законодавства.

7.2 Безпека в надзвичайних ситуаціях

Приділяючи велику увагу зміцненню обороноздатності нашої країни, урядом неодноразово підкреслювалося, що оборонна міць держави складається не тільки з високої готовності й оснащення Збройних сил, а й нерозривно позв’язана з високим рівнем економічного розвитку країни, підготовкою населення й об’єктів народного господарства до захисту від зброї масового ураження.

Велику роль у цьому важливому питанні відіграє цивільний захист країни, що будучи системою загальнодержавних оборонних заходів, покликаний не тільки забезпечити захист населення в надзвичайних ситуаціях, але і здійснювати заходи, спрямовані на забезпечення стабільної роботи підприємств господарювання країни в таких умовах.

Під стійкістю роботи промислового об’єкта розуміють здатність його в умовах надзвичайних ситуацій мирного і воєнного часу випускати продукцію в запланованому обсязі й номенклатурі, а при одержанні слабких і середніх руйнувань, порушенні зв’язків по кооперації і постачанням відновлювати виробництво в мінімальний термін.

Дослідження стійкості роботи об’єкта - це всебічне вивчення обстановки, яка може скластися під час НС та визначення її впливу на виробничу діяльність підприємства. Мета дослідження полягає в тому, щоб виявити слабкі місця в

роботі об`єкта та виробити найбільш ефективні пропозиції, спрямовані на підвищення його стійкості.

Проводиться воно силами інженерно-технічного персоналу із залученням спеціалістів науково-дослідних та проектних організацій. Організатором та керівником досліджень є керівник підприємства.

Оцінка стійкості є другим етапом процесу планування і проведення досліджень. В ході дослідження визначаються умови захисту робітників та службовців від уражаючих факторів, проводиться оцінка уразливості виробничого комплексу від різних уражаючих факторів, оцінюється характер можливих пошкоджень від вторинних уражаючих факторів, вивчається стійкість роботи системи постачання та кооперативних зв`язків з іншими об`єктами, з`ясовуються вразливі місця в системі управління виробництвом.

Послідовність оцінки стійкості роботи об’єкта:

а) визначається критерій (показник), за яким буде проводитись оцінка стійкості щодо конкретного уражаючого фактора, і умови його стійкості;

б) розраховується максимальне значення параметра уражаючого фактора, який може виникати на об’єкті внаслідок аварії, стихійного лиха або застосування сучасної зброї;

в) відповідно до вибраного критерію стійкості визначають межу стійкості об’єкта до даного уражаючого фактора;

г) порівнюють отриману межу стійкості з максимальним значенням уражаючого фактора.

За результатами порівняння визначають, чи об’єкт стійкий щодо даного уражаючого фактора та чи потрібно підвищувати його стійкість.

Розглянемо методику оцінки стійкості об’єкта до дії повітряної ударної хвилі при вибухах:

Основним параметром, що визначає руйнуючу дію повітряної ударної хвилі, є надмірний тиск $\Delta P_{\text{ф}}$.

Критерієм стійкості об'єкта до дії ударної хвилі є граничне значення надмірного тиску ($\Delta P_{\text{ф}}$), при якому елементи об'єкта або не руйнуються, або отримують слабкі та середні зруйнування.

Це значення надмірного тиску називають межею стійкості об'єкта до дії ударної хвилі ($\Delta P_{\text{ф}} lim$). Умови стійкості: якщо $\Delta P_{\text{ф}} lim \geq \Delta P_{\text{ф}.max}$х - об'єкт стійкий до дії ударної хвилі; якщо $\Delta P_{\text{ф}} lim < \Delta P_{\text{ф}.max}$- не стійкий.

($\Delta P_{\text{ф}.max}$ - максимальне значення надмірного тиску ударної хвилі, що очікується на об'єкті при вибуху).

Методика оцінки стійкості об'єкта до дії ударної хвилі включає:

1. Визначення максимального значення надмірного тиску ($\Delta P_{\text{ф}.max}$) ударної хвилі, що очікується у районі об'єкта під час вибуху.

При ядерному вибуху вихідними даними є:

- потужність ядерних боєприпасів (q, кт);
- вид вибуху (наземний чи повітряний);
- відстань від центру міста до об'єкта ($R_{\text{м}}$, км);
- максимальне ймовірне відхилення центра вибуху боєприпасів від точки прицілювання ($r_{\text{відх}}$).

На карті (плані місцевості) позначають імовірну точку прицілювання (нею може бути центр міста). З цієї точки (ТП) з радіусом $r_{\text{відх}}$будується коло, в межах якого найбільш імовірно влучать боєприпаси. Чим ближча точка ЦВ (центр вибуху) до об'єкта, тим більше ушкоджень він отримає (рис. 5.2.1).

Визначають мінімальне можливе віддалення центру (епіцентру) вибуху від об'єкта

$$R_{min} = R_{\text{м}} - r_{\text{відх}} ;$$

і, в залежності від потужності ядерних боєприпасів та виду вибуху, у таблиці додатку 1 знаходять максимальне значення надмірного тиску ударної хвилі, що очікується на об'єкті $\Delta P_{\text{ф}.max}$.

Під час вибуху газоповітряної суміші вихідними даними є:

- маса вуглеводневого продукту (пропану, бутану і т.ін.);

- відстань від центру вибуху до об’єкта.

Шляхом розрахунку або за графіком визначають значення надмірного тиску ударної хвилі, що очікується на об’єкті, та приймають його за максимальне.

2. Визначення границі стійкості об’єкта до дії ударної хвилі ($\Delta P_{\text{ф}} lim$).

Спочатку виділяють основні елементи цеху (об’єкта), від яких залежить виробництво продукції і їх характеристики (з технічної документації). Потім визначається(границя) стійкості кожного з основних елементів об’єкта. Межею стійкості елемента є надмірний тиск, при якому елемент дістане середній ступінь руйнувань. Якщо надмірний тиск, при якому елемент отримує середні руйнування, визначений не одним значенням, а діапазоном (наприклад, 20...30 кПа), то за межу стійкості приймають нижню межу діапазону (у прикладі 20 кПа).

За межу стійкості цеху (об’єкта) в цілому приймають межу стійкості найбільш слабкого елемента об’єкта.

3. Визначення можливої шкоди (відсотків виходу з ладу) елементів об’єкта при очікуваному $\Delta P_{\text{ф}.max}$.

Виявляють, який ступінь руйнування може отримати кожен з елементів об’єкта при надмірному тиску $\Delta P_{\text{ф}.max}$, і визначають можливий збиток залежно від ступеня руйнувань елемента за наведеною нижче таблицею:

Таблиця 4.2.1 – Ступені руйнування елементу

Ступінь руйнувань	Слабкі	Середні	Сильні	Повні
Очікуваний збиток, %	10...30	30...50	50...90	90...100

4. Аналіз результатів оцінки, висновки: - порівнюючи $\Delta P_{\text{ф}} lim$ об’єкта з очікуваною величиною $\Delta P_{\text{ф}.max}$, виявляють чи стійкий об’єкт до дії ударної хвилі. При $\Delta P_{\text{ф}} lim \geq \Delta P_{\text{ф}.max}$ - об’єкт стійкий, а при $\Delta P_{\text{ф}} lim < \Delta P_{\text{ф}.max}$- не стійкий до дії ударної хвилі;

– до якої величини доцільно підвищувати стійкість об’єкта;
– які з елементів найменш стійкі (з малими $\Delta P_{\text{ф}} lim$).

Доцільно підвищувати стійкість об’єкта до очікуваного значення $\Delta P_{ф.max}$, якщо це не зумовить великих економічних витрат. У іншому випадку достатньо буде підвищити стійкість найбільш слабких елементів до рівня стійкості більшості елементів об’єкта.

На основі висновків пропонують заходи щодо підвищення стійкості роботи об’єкта.

Такими заходами можуть бути:

- укріплення несучих конструкцій та перекрить будівель встановленням додаткових колон, ферм, контрфорсів або підкосів;
- розміщення обладнання на нижніх поверхах будівель або в підвалах, надійне закріплення його на фундаменті, встановлення захисних кожухів або ковпаків;
- прокладання кабельних мереж та трубопроводів під землею;
- створення резервних запасів контрольно-вимірювальної апаратури.

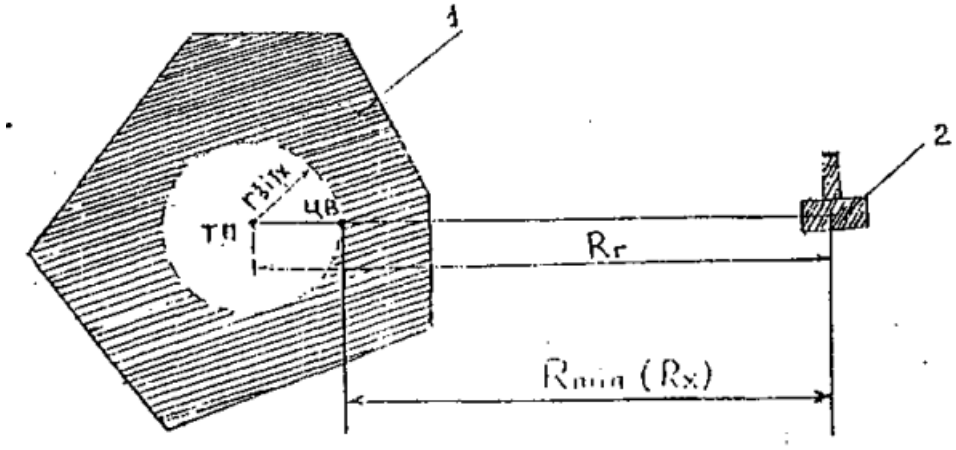


Рисунок 4.2.1 - Визначення мінімальної відстані до імовірного центру вибуху. 1-місто, 2-об'єкт [14]

Таким чином дана система оцінки стійкості роботи промислового підприємства до впливу вторинних вражаючих факторів виконана з вимог техніки безпеки.

ВИСНОВОК

Спроектований та розроблений додаток призначений для підтримки гри в Dungeons & Dragons шляхом надання корисних інструментів та функцій як для гравців, так і для майстрів гри.

Розробка серверного додатку була критично важливою для забезпечення централізованого управління даними та послугами. Серверна частина була розроблена для обробки запитів від клієнтів, таких як збереження інформації про персонажів, їх інвентарі та прогрес у грі. Використання сокетів на мові програмування C# забезпечило ефективну взаємодію між клієнтами та сервером.

Клієнтська сторона додатку була створена для надання інтуїтивно зрозумілого інтерфейсу користувача, що дозволяло гравцям легко здійснювати запити до сервера, вносити дані та отримувати команди та інформацію. Функціональність клієнта включає створення та управління персонажами, ведення ігрових записів та доступ до ігрових ресурсів.

Використання асинхронного програмування дозволило серверу "Блокнота D&D" ефективно обробляти багато одночасних запитів без блокування або затримок. Це було особливо важливо для підтримки великої кількості користувачів та забезпечення швидкої реакції додатку.

Під час розробки додатку всебічною та враховані важливі аспекти сучасної розробки програмного забезпечення, масштабованість, асинхронність та ефективність. Цей додаток стане важливим інструментом для гравців Dungeons & Dragons, забезпечуючи їм зручний та надійний спосіб керування ігровим процесом та персонажами.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. START YOUR D&D JOURNEY! [Електронний ресурс] // Wizards of the Coast. – 2023. – Режим доступу до ресурсу: https://www.dndbeyond.com/how-to-play-dnd.

2. D MELZER J. What Is D&D Beyond - and How Do You Use It? [Електронний ресурс] / JENNY MELZER // cbr.com. – 2023. – Режим доступу до ресурсу: https://www.cbr.com/dnd-beyond-guide/#what-is-dnd-beyond-anyway.

3. Fight Club 5th Edition [Електронний ресурс] // Mac App Store. – 2019. – Режим доступу до ресурсу: https://apps.apple.com/us/app/fight-club-5th-edition/id1484084460?mt=12.

4. EditionGame Master 5th Edition [Електронний ресурс] // amazon. – 2020. – Режим доступу до ресурсу: https://www.amazon.com/Lions-Den-Game-Master-Edition/dp/B08473NM2H.

5. Microsoft to acquire Xamarin and empower more developers to build apps on any device [Електронний ресурс] // Microsoft. – 2016. – Режим доступу до ресурсу: https://blogs.microsoft.com/blog/2016/02/24/microsoft-to-acquire-xamarin-and-empower-more-developers-to-build-apps-on-any-device/.

6. TCP/IP Sockets in C# / David B. Makofske, David Makofske, Michael J. Donahoo, Kenneth L. Calvert., 2004. – 175 с. – (Elsevier Science).

7. What is Use Case Diagram? [Електронний ресурс] // Visual Paradigm. – 2023. – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/.

8. What is Class Diagram? [Електронний ресурс] // Visual Paradigm. – 2023. – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/.

9. What is Sequence Diagram? [Електронний ресурс] // visual paradigm. – 2023. – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/.

10.CLR via C# (Developer Reference), 2012. – 894 с. – (Microsoft Press). – (4-е вид.; кн. 978).

11.Paul F. Johnson. Cross-platform UI Development with Xamarin.Forms / Paul F. Johnson., 2015. – 330 с.

12.Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508-18#Text.

13.Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [Електронний ресурс]. – 1999. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99#Text

14.Навчальний посібник «ТЕХНОЕКОЛОГІЯ ТА ЦИВІЛЬНА БЕЗПЕКА. ЧАСТИНА «ЦИВІЛЬНА БЕЗПЕКА»» / автор-укладач В.С. Стручок– Тернопіль: ФОП Паляниця В. А., – 156 с. Отримано з http://elartu.tntu.edu.ua/handle/lib/39424

ДОДАТКИ

Додаток А

Тези

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 004.41
О. Кузьмич
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

РОЗРОБКА ANDROID-ДОДАТКУ ДЛЯ ГРАВЦІВ НАСТІЛЬНО-РОЛЬОВОЇ ГРИ D&D З ВИКОРИСТАННЯМ XAMARIN ТА SOCKET

O. Kuzmych

DEVELOPMENT OF AN ANDROID APP FOR TABLETOP ROLE-PLAYING GAME D&D USING XAMARIN AND SOCKET

В сучасному світі технологій настільно-рольові ігри, зокрема Dungeons & Dragons (D&D), знову набувають великої популярності. Однак, щоб зробити гру більш доступною та захоплюючою, виникає необхідність в розробці інноваційних мобільних додатків. У цьому контексті, розробка Android-додатку, який використовує технології Xamarin та Socket для поліпшення взаємодії гравців та Майстра Гри, визначається як важливий крок у напрямку модернізації та вдосконалення гри.

Настільно-рольові ігри, такі як D&D, вимагають від гравців та Майстра Гри постійного спілкування та обміну інформацією. Використання мобільного додатку може значно полегшити цей процес, роблячи його більш зручним та динамічним. Першою задачею є розробка інтуїтивного інтерфейсу для гравців, що дозволяє легко створювати та редагувати персонажів, ведення логів пригод, та обмін інформацією з іншими гравцями у реальному часі. Зокрема, додаток буде використовувати технологію Socket для створення зручного механізму обміну даними між гравцями під час гри.

Однією з ключових переваг проекту є використання фреймворку Xamarin, що дозволяє розробникам створювати крос-платформенні додатки, що працюють як на Android, так і на iOS. Це рішення дозволяє широкому колу гравців користуватися додатком на будь-якому мобільному пристрої.

Використання технології Socket відкриває можливості комунікації в реальному часі не зважаючи на дистанцію між ними під час гри. Гравці можуть обмінюватися повідомленнями, стратегіями та даними про своїх персонажів, створюючи більш імерсивне ігрове середовище. Крім того, це дозволяє проводити сесії гри в онлайн-режимі, об'єднуючи гравців з різних місць.

Додаток надає можливість витрачати більше часу на пригоди, не відволікаючи процесом розрахунку результатів перевірок чи кидків кубика за допомоги системи розрахунку за формулами, може автоматизувати складні обчислення під час гри, зокрема при використанні спеціальних кубів або розрахунків модифікаторів. Під час довгих пригод дозволяє уникнути помилок у розрахунках та забезпечити точність механіки гри, зокрема в умовах, коли вам потрібно здійснювати багато кидків чи враховувати складні правила

Література

1. Florence Desiata. Surprising Board Games Statistics to Look Out for in 2023. PlayToday.co. URL: https://playtoday.co/blog/stats/board-games-statistics/ (date of access: 01.12.2023).

2. What is Xamarin?. Microsoft. URL: https://learn.microsoft.com/en-us/xamarin/get-started/what-is-xamarin (date of access: 01.12.2023).

3. Sockets in .NET. Microsoft. URL: https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/sockets/sockets-overview (date of access: 01.12.2020).