Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНІ РОБОТА

на тему:

«Розробка методів і програмних засобів для підвищення ефективності ігрових механізмів на основі Telegram-ботів»

Виконав: студент 2-го курсу групи 2ПІ-20м
спеціальності 121 «Інженерія програмного забезпечення»
Цукрук В. І.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ
Романюк О.В.
(прізвище та ініціали)
«___» _____ 2021 р.

Опонент: д.т.н., проф. каф. КН
Іванчук Я.В.
(прізвище та ініціали)
«___» _____ 2021 р.

Допущено до захисту
завідувач кафедри ПЗ
д.т.н. проф. Романюк О.Н.
(призвіще та ініціали)
« » _____2021 р.

Вінниця ВНТУ – 2021

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 Інформаційні системи
Спеціальність 121 – «Інженерія програмного забезпечення»
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

Романюк О.Н.

“13” вересня 2021 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Цукруку Валентину Івановичу

1. Тема роботи: «Розробка методів і програмних засобів для підвищення ефективності ігрових механізнів на основі Telegram-ботів»

Керівник роботи: Романюк Оксана Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від “24” вересня 2021 року № 277.

2. Строк подання студентом роботи: “01” грудня 2021 року.

3. Вихідні дані до роботи: модель розробки – водоспадна; метод передачі повідомлень між серверами – LongPolling; метод балансування характеристик – транзитивний; метод створення ігрових об'єктів – від характеристик, метод генерації квестів – на основі орієнтованого графу; вхідні дані – інформація про неконтрольованих гравцями персонажів, локації та шаблони предметів у форматі «.xlsx»; ідентифікаційний номер Telegram-бота у відповідному форматі; ідентифікатор користувача; вихідні дані – результати виконання алгоритмів, як відповідь на команди гравця у текстовому фоматі.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану питання та постановка задач дослідження, розробка методів та алгоритмів для підвищення ефективності ігрових механізмів розробка програмних засобів, тестування системи, економічна частина, висновки, список використаних джерел, додатки;

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):
мета, об’єкт та предмет дослідження; завдання дослідження; аналіз стану питання; порівняння з аналогами; використані технології при розробці системи; тестування системи; економічне обгрунтування; наукова новизна одержаних результатів; практична цінність одержаних результатів, апробація та публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Романюк О.В. к.т.н, доцент кафедри ПЗ		
5	Бурєннікова Н.В., д.е.н., професор ЕПВМ		

7. Дата видачі завдання: “14” вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання	15.09.20 – 25.09.20	Вик.
2	Аналіз існуючих аналогів та постановка задач дослідження	26.09.20 – 28.09.20	Вик.
3	Розробка методів та алгоритмів для підвищення ефективності ігрових механізмів	29.09.20 – 12.10.20	Вик.
4	Написання програмних модулів для реалізації розроблених алгоритмів	13.10.20 – 28.10.20	Вик.
5	Розробка серверної частини Telegram-бота	29.11.20 – 20.11.20	Вик.
6	Тестування розробленого програмного продукту	21.11.20 – 25.11.20	Вик.
7	Економічна частина	26.11.20 – 28.11.20	Вик.
8	Оформлення матеріалів до захисту МКР	29.11.20 – 30.11.20	Вик.

Студент _____ **Цукрук В.І.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Романюк О.В.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.422.8

Цукрук В. І. Розробка методів і програмних засобів для підвищення ефективності ігрових механізнів на основі Telegram-ботів. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2021.

На укр. мові. Бібліогр.: 36 назв; рис.: 46; табл.: 17.

У магістерській кваліфікаційній роботі розроблено методи та алгоритми для підвищення ефективності ігрових механізмів, робота яких демонстрована за допомогою розробленого Telegram-бота. Подальшого розвитку отримав метод обчислення характеристик ігрових предметів за допомогою кривої «рівня-потужності». Подальшого розвитку отримав метод генерації квестів. Уперше запропоновано формулу балансування характеристик персонажів. Як результат вдалось прискорити балансування характеристик предметів різних типів на різних рівнях, покращити варіативність генерованих квестів, оптимально збалансувати допустимі межі характеристик на кожний рівень.

Для розробки використано мову програмування Python, середовище розробки PyCharm, СКБД MySQL та інструментарій SQLAlchemy.

Ключові слова: Telegram, бот, ігрові механіки, MMORPG, генератор квестів, балансування характеристик, Python, MySQL.

ABSTRACT

UDC 004.422.8

Tsukruk V. I. Development of methods and software to increase the efficiency of game mechanisms based on Telegram-bots. Master's thesis in specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2021.

In Ukrainian language. Bibliogr .: 36 titles; fig .: 46; tab .: 17.

In the master's qualification work were developed methods and algorithms to increase the efficiency of game mechanisms, the work of which is demonstrated using the developed Telegram-bot. The method of calculating the characteristics of game objects using the "power-level" curve was further developed. The method of generating quests was further developed. For the first time, a formula for balancing the characteristics of characters has been proposed. As a result, it was possible to accelerate the balancing of the characteristics of objects of different types at different levels, to improve the variability of generated quests, to optimally balance the allowable limits of characteristics for each level.

Python programming language, PyCharm development environment, MySQL database and SQLAlchemy tools were used for development.

Keywords: Telegram, bot, game mechanics, MMORPG, quest generator, performance balancing, Python, MySQL.

ЗМІСТ

ВСТУП 8

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ 12

1.1 Аналіз стану питання 12

1.2 Порівняльний аналіз аналогів 14

1.3 Аналіз методів обчислення характеристик 18

1.4 Аналіз методів балансування характеристик 21

1.5 Аналіз методів автоматичної генерації квестів 24

1.6 Постановка задачі розробки 26

1.7 Висновки 26

2 РОЗРОБКА МЕТОДІВ ТА АЛГОРИТМІВ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ІГРОВИХ МЕХАНІЗМІВ 27

2.1 Розробка методу автоматизації обчислення характеристик ігрових предметів 27

2.2 Розробка методу генерації варіативних квестів на основі орієнтованого графу 28

2.3 Розробка методу балансування характеристик персонажів 30

2.4 Розробка інтерфейсу за допомогою Telegram API 32

2.5 Розробка структури інтерфейсу для ігрового Telegram боту 33

2.6 Розробка блок-схем методів та алгоритмів для роботи ігрового Telegram боту 36

2.7 Висновки 42

3 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТ ДЛЯ ІГРОВОГО TELEGRAM-БОТУ 43

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу 43

3.2 Вибір середовища розробки та СКБД 46

3.3 Програмна реалізація ігрового боту 51

3.4 Висновки 60
4 ТЕСТУВАННЯ ПРОГРАМИ 61
4.1 Аналіз методів тестування програмного забезпечення 61
4.2 Тестування розробленого програмного продукту 63
4.3 Розробка інструкції користувача 73
4.4 Вимоги до персонального комп'ютера 77
4.5 Висновки 78
5 ЕКОНОМІЧНА ЧАСТИНА 79
5.1 Оцінювання комерційного потенціалу розробки 79
5.2 Оцінювання рівня новизни розробки 84
5.3 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи. 90
5.3 Прогнозування комерційних ефектів від реалізації результатів розробки ... 96
5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності 98
ВИСНОВКИ 101
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 102
ДОДАТКИ 106
ДОДАТОК А. Технічне завдання 107
ДОДАТОК Б. Протокол перевірки роботи 111
ДОДАТОК В. Лістинг коду 112
ДОДАТОК Г. Іллюстративна частина 180

ВСТУП

Обґрунтування вибору теми дослідження. MMORPG (Massively Multiplayer Online Role Playing Game, масова, розрахована на багато користувачів, ролева онлайн-гра) – це жанр комп'ютерної гри, в якій рольова гра (RPG) поєднується з масовою онлайн-грою (MMO). Головна особливість MMORPG гри – це безліч гравців в одному віртуальному світі [1].

Станом на 2021 рік найпопулярнішими жанрами ігор є: MMORPG, battle royale, симулятори, MOBA [2].

Якщо ще декілька років тому ігри жанру MMORPG були в основному лише на персональні комп’ютери, то з 2020 року ця тенденція цілком змінилась, вони почали з’являтись на всіх платформах, особливої популярності ці ігри набули на платформах Android та IOS.

Окрім класичних ігор цього жанру з'явились аналоги на доволі неочікуваних платформах, як приклад – чат-боти в різних месенджерах. Ця ідея також набрала велику популярність, проте зіштовхнулась з деякими великими проблемами.

Основна складність в реалізаціях ігор на подібних платформах це обмеження у можливостях графічного інтерфейсу, обчислювальної потужності, тощо. Ці ігри обмежені до 2 можливих дій – повідомлення від сервера, відповідь від користувача.

Головною ідеєю MMORPG є забезпечення можливості граців зручно взаємодіяти між собою. Для цього було обрано вже існуючий месенджер Telegram. Першою перевагою є те, що у мессенджері Telegram є особливі акаунти, операторами яких можуть бути не люди, а спеціальним чином написані програми, розташовані на сторонніх ресурсах (не на серверах Telegram). Ці програми-оператори називаються ботами. Боти можуть отримувати адресовані їм повідомлення, а також генерувати і відправляти відповідні повідомлення. Все це вони роблять через свій аккаунт, використовуючи спеціальний API. По-друге, він вже має свою велику

аудиторію, що забезпечить легку рекламу і зручність для гравців: у них не буде потреби завантажувати окремий додаток, коли набагато простіше грати там, де й відбувається спілкування. По-третє, Telegram дозволяє створювати спільні чати між людьми та канали для викладання деякої інформації, що забезпечить зручну взаємодію гравців.

Жанр MMORPG характеризується великою кількістю різноманітних ігрових механізмів для забезпечення різноманітного ігрового процесу для кожної ролі за яку можуть грати користувачі. Подібні механіки потребують широкого спектру можливостей платформи, на якій вони реалізуються, що являється доволі великою перешкодою для ігор на базі чат-ботів. На разі більшість з відомих ігрових механік не знайшли свою адаптацію під текстовий режим гри.

Саме тому модифікація ігрових механізмів до подібних обмежень є доволі актуальною задачею. Отримані модифіковані методи не прив’язані безпосередньо до Telegram месенджеру та можуть бути використані при розробці ігор подібного жанру на будь які платформи.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалась відповідно до плану науково-дослідних робіт кафедри програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення ефективності ігрових механізмів та їх балансування.

Для цього необхідно виконати такі завдання:

- провести аналіз існуючих методів підвищення ефективності ігрових механізмів;

- розробити алгоритм приведення характеристик до універсальної одиниці виміру;

- розробити алгоритм побудування кривої «рівня-потужності»;

- розробити алгоритм автоматизації методу обчислення характеристик ігрових предметів;

- розробити алгоритм генерації варіативних квестів за допомогою орієнтованого графу;

- розробити алгоритми PVP (гравець проти гравця) та PVE (гравець проти навколишнього середовища);

- вивести формулу для балансування характеристик персонажів;

- розробити програмні засоби для реалізації алгоритмів;

- розробити програмні засоби багатокористувацької онлайн рольової гри;

- розробити серверну частину Telegram-бота;

- провести тестування системи.

В результаті виконання перелічених завдань буде розроблено Telegram-бот, який в повній мірі зможе демонструвати всі розроблені методи.

Об’єкт дослідження – процес розробки ігрових механізмів.

Предмет дослідження – методи та засоби підвищення ефективності ігрових механізмів та їх балансування в багатокористувацьких іграх.

Методи дослідження. У процесі дослідження використовувались: теорія чисел, лінійна алгебра та математична статистика для балансування характеристик персонажів та розробки методу обчислення характеристик ігрових предметів; теорія ймовірності та дискретна математика для розробки алгоритму генерації квестів; комп’ютерне моделювання для перевірки та аналізу теоретичних положень.

Наукова новизна одержаних результатів.

1. Подальшого розвитку отримав метод обчислення характеристик ігрових предметів за допомогою кривої «рівня-потужності», який, на відміну від класичного методу, використовує формулу приведення характеристик до універсальної одиниці виміру та таблицю коефіцієнтів розподілення характеристик в залежності від типу та класу предмету, що дозволило прискорити балансування характеристик предметів різних типів на різних рівнях.

2. Подальшого розвитку отримав метод генерації квестів, який, на відміну від класичних методів, використовує орієнтовний граф для будування структури квестів, що дозволили покращити їх варіативність.

3. Уперше запропоновано формулу балансування характеристик персонажів, у якій використано розрахунок середнього пошкодження для персонажу під час поєдинку, що дозволило оптимально збалансувати допустимі межі характеристик на кожний рівень.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та програмні засоби для автоматизації обчислення характеристик ігрових предметів та генерації варіативних квестів на основі орієнтованого графу.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У роботах, опублікованих у співавторстві, здобувачу належить розробка методу обчислення характеристик ігрових предметів в рольових багатокористувацьких іграх [3]; розробка алгоритму генерації варіативних квестів на основі орієнтованого графу [4] та розробка алгоритму для розрахунку бойових характеристик персонажів ігрового Telegram-боту [5].

Апробація матеріалів магістерської кваліфікаційної роботи. Результати роботи доповідалися на XIV науково-практичній конференції «Інформаційні технології і автоматизація» – Одеса 2021, міжнародній науково-практичній інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ» – Вінниця 2021, XI міжнародній науково-технічній конференції «Інформаційно-комп’ютерні технології – 2020 (ІКТ-2020)» - Житомир 2020.

Публікації. Основні результати досліджень опубліковано в 3 наукових працях, у матеріалах конференцій.

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану питання

Ігровий баланс є частиною дизайну гри, його можна описати як математично-алгоритмічну модель ігрових чисел, ігрових механік і відносин між ними. Таким чином, балансування гри полягає у коригуванні механік для створення бажаного досвіду гравців [6].

Ігрові механіки – це процедури та правила гри. Вони визначають мету і методи її досягнення.

Ігровий елемент – термін для всіх видів сутностей у грі. Може стосуватись солдатів у стратегічній грі в реальному часі, персонажа рольової гри, а також до предметів та заклинань останнього.

Незважаючи на те, що ігровий баланс є актуальним для всіх типів ігор, багато публікацій, починаючи з початку тисячоліття, зосереджені на відеоіграх. Автори літератури переважно займаються ігровим дизайном відеоігор загалом і присвячують кілька розділів ігровому балансу. Останнім часом з'явилися також онлайн-блоги та відео, присвячені виключно ігровому балансу.

Раніше автори в основному писали про одиночні ігри та ігри PvE і пов'язаними з ними концепціями, такі як складність. Інтернет-контент, який був в основному створений у 2010-х роках, більше фокусується на PvP та онлайн-іграх, але часто посилається на ідеї, які були описані в попередніх роботах.

PvP – гравець проти гравця та описує ігри, в яких відбувається пряме змагання між гравцями. Однак це не виключає елементів PvE. У деяких випадках використовується термін PvPvE.

PvE – гравець проти навколишнього середовища. У цих іграх гравець знаходиться в межах особистої версії ігрового світу і протистоїть лише «гравцям», керованим комп'ютером.

Лише зовсім недавно балансування гри почало більше враховувати гравців, особливо створену гравцями "ме́ту".

"Метагейм" – найвищий рівень стратегії в багатьох складних іграх, мета-гра відноситься до будь-якого аспекту стратегії, який передбачає роздуми про те, що думає ваш опонент. Метагра вступає в дію в будь-якій грі, де жодна стратегія не є домінантною, а протиборчі сторони знають про безліч стратегій, які можуть досягти успіху залежно від дій супротивників [7].

"Мéта" – в свою чергу, це набір стратегій котрі є найпопулярнішими в даний момент часу. Саме "мéта" може підказати в яку сторону потрібно підкоригувати баланс, на що потрібно звернути особливу увагу, чому певна стратегія входить в "мéту", а інша зовсім не зустрічається серед гравців.

Набір стратегій котрі можуть використувати гравці напряму залежать від реалізовних ігрових механік та того, як вони збалансовані.

В кожногу жанру ігор є свій набір тривіальних ігрових механік якими він характеризується. Тобто, жанр ігри визначається не лише типовим для того чи іншого жанру світом, сюжетом а і ігровими механіками якими гравці керуються взаємодіючи з вище перечисленими ігровими елементами.

MMORPG та RPG ігри в свою чергу володіють найбільшою кількістю та різноманітністю можливих ігрових механік, адже їм важливо перекрити широкий сегмент користувачів та наділити кожну можливу роль гравця своїми унікальними механіками та елементами. Ось декілька основних ігрових механік цих жанрів [8]:

1. Механіка досягнень – є, власне, найважливішою частиною гри. Адже саме розуміння того, що ти чогось досяг, змушує гравців знову і знову повертатися. Як досягнення можна розглядати багато речей: медалі, ордени, будь-які предмети. Головне, щоб це показувало успіх гравця у тій чи іншій галузі гри;

2. Механіка призначеної зустрічі – суть цієї механіки в тому, щоб повернутися в гру у певний час. Найчастіше це робиться щоб одержати якесь досягнення або винагороду. Тісно пов'язана з механікою уникнення покарання;

3. Механіка уникнення покарання – суть цієї механіки в тому щоб покарати гравця якщо він не виконає якусь дію вчасно. Механіка жорстока, але

у поєднанні з механікою призначеної зустрічі працює відміно. Наприклад, у грі відбудеться якийсь захід, а людям, які не відвідали його, буде накладено штраф: не можна брати участь у заходах 7 днів.

Отже, саме через популярність онлайн ігор та MMORPG зокрема, великої різноманітності можливих механік для впровадження, а також наростаючої популярності ботів у месенджерах та збільшення потоку нових користувачів подібних платформ, було вирішено написати свою гру на базі Telegram-бота. А також адаптувати та покращити деякі звичні для ігор цього жанру механіки.

1.2 Порівняльний аналіз аналогів

Наразі існує велика кількість ігор різноманітних жанрів, написаних на базі мессенджеру. Проте основним жанром залишається саме MMORPG.

В ході дослідження було обрано 3 ігрових бота жанру MMORPG для аналізу, а саме CW3 (Chat Wars 3), RF Telegram (Rising Force Telegram), і Hyperion.

CW3 продовження гри CWC (Chat Wars Classic) – гра на тематику середньовіччя [9]. Приклад інтерфейсу гри наведено на рисунку 1.1.

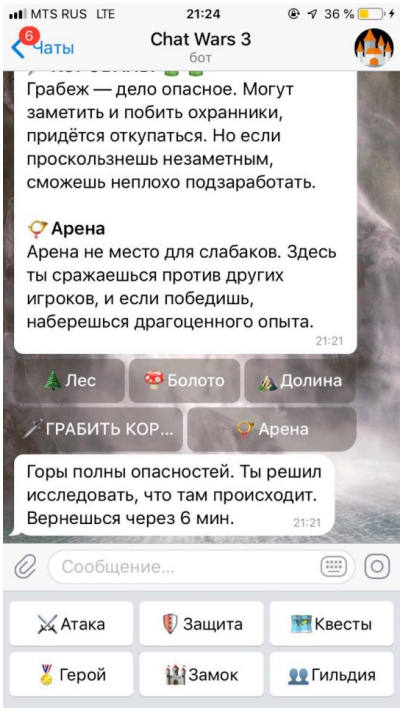


Рисунок 1.1– Приклад інтерфейсу гри «CW3»

Гра характеризується автоматичною системою проведення масових боїв, відсутністю унікальних речей, а також системою PVP (Player vs Player) по типу гри в «камінь-ножиці-папір». Всіх гравців розподілено на 7 замків, всередині яких йде розподіл на гільдії.

RF Telegram – гра на тематику фентезі, написана за мотивами комп'ютерної гри Rising Force Online 2004 року [10]. Гравці розподілені на три раси, всередині яких також йде розподіл на гільдії. Характеризується напів-автоматичним режимом масових боїв та автоматичним режимом PVP, наявний генератор унікальних речей лише на високих рівнях. Приклад інтерфейсу гри наведено на рисунку 1.2.

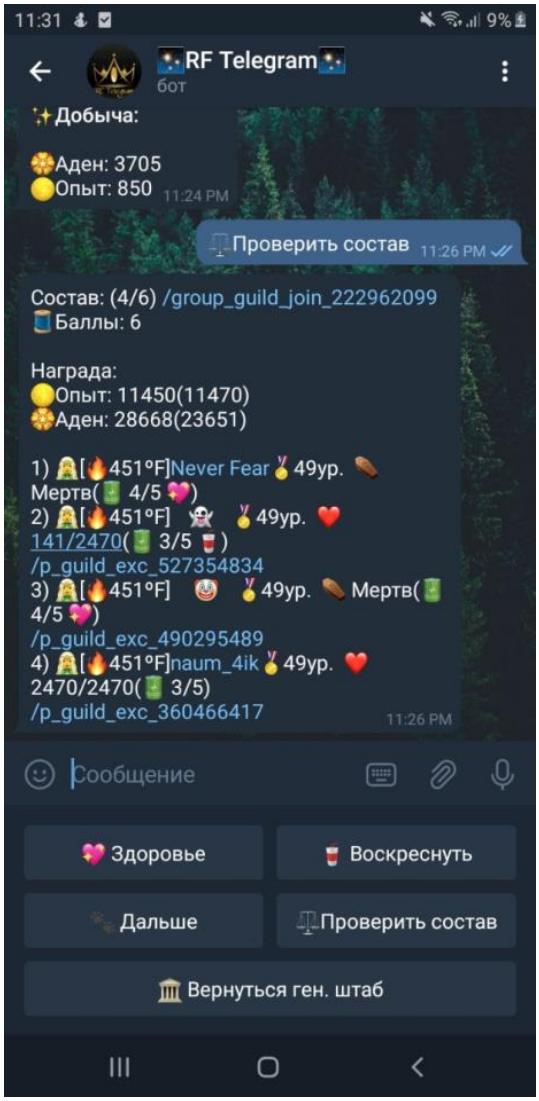


Рисунок 1.2– Приклад інтерфейсу гри «RF Telegram»

Hyperion – гра за мотивами однойменного науково-фантастичного роману Дена Сімсона, написаного в 1989 році [11]. Відрізняється від решти наведених відкритим світом зі свободою пересування, в масових боях гравець повністю відігрує свої дії, коли як PVP проходить повністю автоматично. Кожна річ в грі має декілька ступенів рідкості. Гравці розподілені на 4 міста, в яких йде розподіл на загони. Приклад інтерфейсу гри наведено на рисунку 1.3.

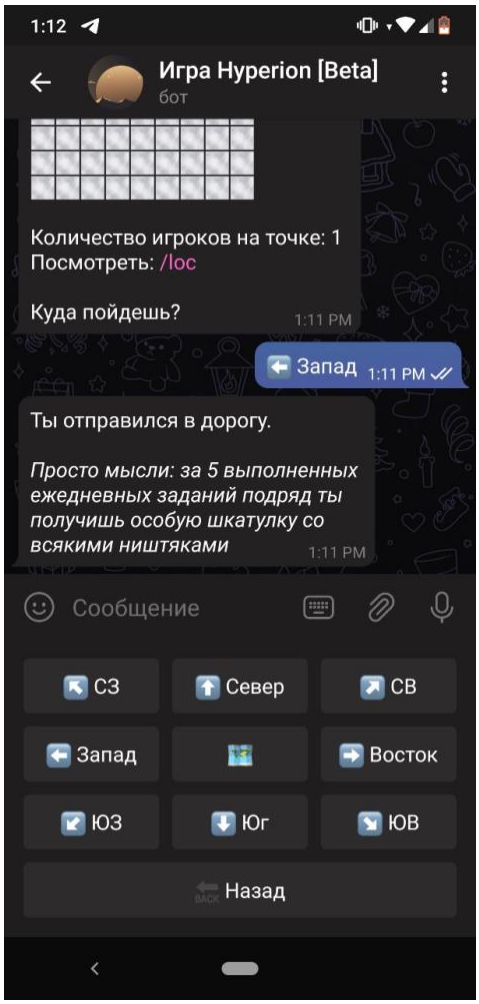


Рисунок 1.3 – Приклад інтерфейсу гри «Hyperion»

У всіх ботах виявлено 2 основних недоліки, які є доволі критичними як для ігор такого жанру. По-перше, це вибраний спосіб отримання повідомлень від Telegram, а другою критичною проблемою є розроблені бойові системи.

Розглянемо першу проблему детальніше. У всіх трьох іграх використовується технологія під назвою WebHooks. Для більшості ботів це

оптимальний вибір, адже її суть полягає в тому, що як тільки Telegram отримує якісь запити, він спочатку робить запит до серверу, повідомляючи про те, що в нього є повідомлення, і лише потім, впевнившись, що сервер готовий його прийняти, відправляє йому. Проте для MMORPG це може стати причиною зниження продуктивності, оскільки час від часу кількість запитів зростає в рази залежно від того, який етап гри буде проходити в той чи інший момент, наприклад на 12- ту годину дня заплановано захват замку гільдії А гільдією Б, а тому в цей час збирається велика кількість людей, які будуть надсилати велику кількість запитів одночасно. Для того, щоб в такі моменти сервер не виконував зайві дії і всі повідомлення оброблялись вчасно зі збереженням коректної послідовності, було вирішено використовувати технологію LonPolling. Суть технології полягає в тому, щоб з певним інтервалом часу брати і забирати всі можливі запити, які надійшли до бота. Мінімізація цього інтервалу дозволить обробляти запити користувачів набагато швидше і ефективніше, хоча і буде потребувати більше потужності сервера.

Друга проблема напряму впливає на рівень зацікавленості гравця. Так, бот, який оперативно відповідає на запити є безсумнівною перевагою, проте багато людей скаржились на доволі скудну, нереалістичну бойову систему, яка може швидко набриднути. У випадку CW3 бойова система нагадує гру у «камінь, ножиці, папір», де кожен з супротивників обирає частину тіла, яку він хоче атакувати та захищати, а у двох інших прикладах бойова система повністю автоматична – хтось наносить пошкодження, хтось може ухилитися, або ж нанести критичне пошкодження.

У жанрі MMORPG можливість битися з іншими гравцями є найцікавішою частиною гри. Нажаль, можливості мессенджеру доволі обмежені, тому створити повноцінну бойову систему практично неможливо [12].

Проаналізувавши усі аналоги, визначено їхні можливості та недоліки, які враховувались при створенні власного програмного забезпечення з назвою «FireBox» (табл. 1.1).

Таблиця 1.1 – Порівняльні характеристики програмних продуктів

Критерій	CW3	Hyperion	RF	FireBox
Зручний графічний інтерфейс	+	+	+	+
Зручний метод передачі повідомлень	-	-	-	+
Відкритий світ	-	+	-	-
Різноманітна бойова система	-	+	-	+
Добре продумана економічна складова	+	-	+	+
Баланс характеристик	-	+	-	+
Генератор унікальних речей	-	-	-	+

Таблиця порівняльних характеристик показала, що розробка програмного продукту є доцільною. В результаті отримаємо продукт, що покриває недоліки існуючих рішень і зможе легко конкурувати на ринку.

1.3 Аналіз методів обчислення характеристик

В ігровому балансі є два основних методи створення ігрових елементів.

Перший метод – від образа. Художник створює малюнок, який описує певний ігровий елемент. На основі цього малюнку робляться висновки про початкові характеристики ігрового елементу.

Приклад створення персонажів цим методом зображено на рисунку 1.4.

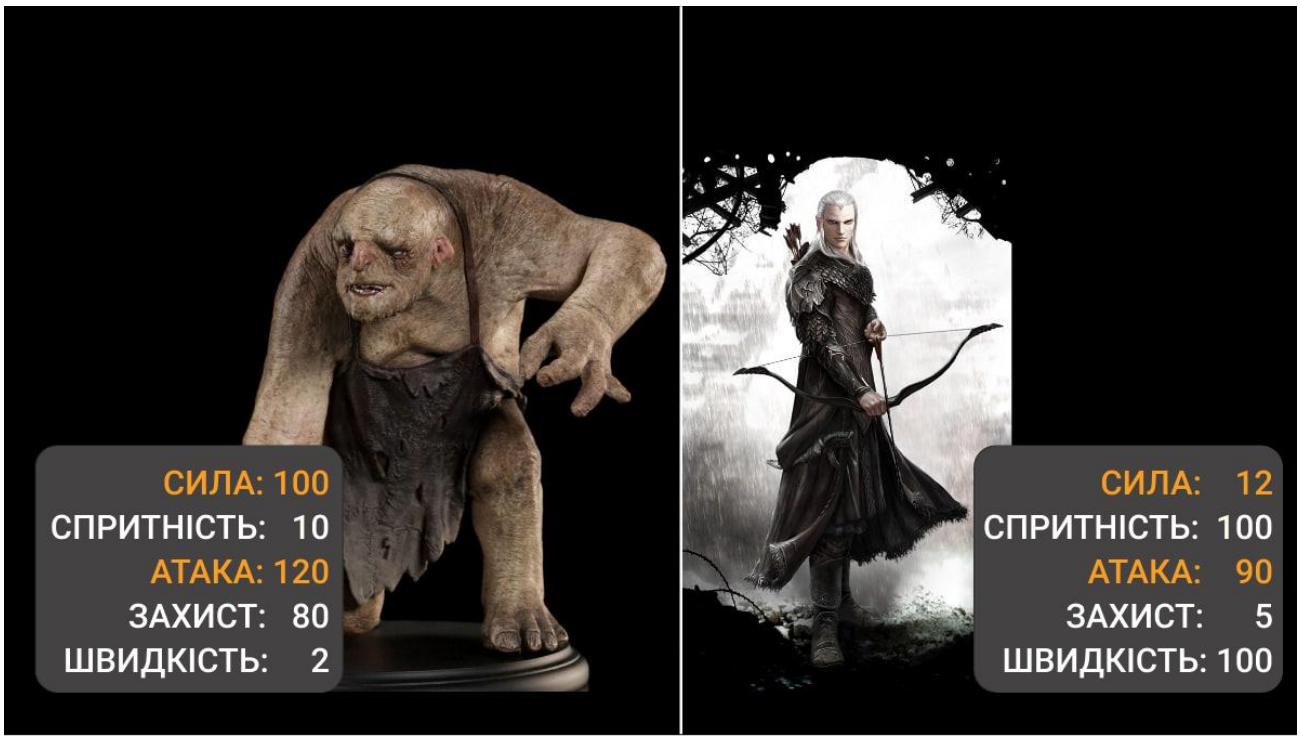


Рисунок 1.4 – Створення ігрових елементів методом "від образу"

Зліва – малюнок великого товстого троля, одразу зрозуміло що він має бути сильним та з великою атакою, а через будову свого тіла незграбним і повільним.

Справа – малюнок ельфа, він має бути швидким та спритним. Проте з набагато меншою силою ніж у троля.

Аналізуючи отримані образи можна зробити висновки у їх співвідношені в різних аспектах та побудувати таблицю початкових характеристик для різних ігрових елементів.

Другий метод – від характеристик. Спочатку описуються характеристики певного предмету, а потім на їх основі створюється образ предмету [13].

Приклад створення бойової техніки за допомогою описаного методу зображено на рисунку 1.5.

Рисунок 1.5 – Створення ігрових елементів методом "від характеристик".

Художник отримує список характеристик певного ігрового елементу, наприклад:

– переміщується зі швидкістю 60 – 80 км/ч;
– легко долає бездоріжжя;
– керується екіпажем з 3-х чоловік;
– завдає великої шкоди живій силі супротивника та легкоброньованій техніці;
– захищений бронею 50 мм.

На основі цих характеристик художник створює образ фантастичного всюдиходу.

Обирати метод потрібно в залежності від наявності ідеї та тих чи інших ключових елементів. Наприклад, якщо важливо щоб в ігрі були присутні конкретні персонажі або предмети, то потрібно обирати метод від образу. Так, як в Telegram-ботах майже відсутня графічна складова – художні образи майже не мають значення. Тому для підрахунку початкових значень було вирішено обрати другий метод – від характеристик.

1.4 Аналіз методів балансування характеристик

Існують 3 основні методи балансування ігрових механік в цілому, а саме транзитивний, інтранзитивний та некомпаративний.

Транзитивний метод – це метод прямого порівняння характеристик. Транзитивний метод використовується у випадках, коли порівнювальні об’єкти мають однаковий, або схожий за сенсом, набір характеристик. Для балансування цим методом будується таблиця, в яку заносяться дані про об’єкти.

Інтранзитивний метод. Балансування інтранзитивним методом передбачає занесення в матрицю інформації про об’єкти, котрі взаємодіють, та результат їх взаємодії. Для цього будується алгоритм котрий повинен симулювати дії гравців.

Некомпаративний метод – це метод балансу при якому використовуються неможливі для порівняння характеристики. Тобто замість балансування як такого, йде заміна характеристик, котрі потребують балансування, на ті, які неможливо порівняти, а відповідно і збалансувати. Такий метод є доволі поганим і його використання можливе лише у крайніх ситуаціях.

Для балансування ігрового процесу у розроблюваному ігровому боті підходить лише транзитивний метод, адже просимулювати дії гравців у онлайнових багатокористувацьких іграх, з великою варіативністю можливих подій, просто неможливо.

Ігрові взаємодії – це сукупність параметрів, які змінюють свої значення під дією будь-яких явищ (сутностей). Якщо у одного об'єкта є параметр «Пошкодження», а в іншого об'єкта наявний параметр «Здоров'я», то можливо створити ігровий процес «Об'єкт здійснює атаку по іншому об'єкту» і в результаті параметр «Здоров'я» змінює своє значення на величину параметра «Пошкодження» [14].

Таким чином, виходить, що вплив (в даному прикладі вплив пошкодження на здоров'я) є причиною зміни параметрів об'єкта. Для більш

глибокого розуміння, спробуємо класифікувати типи впливу. Вони бувають миттєвими або тимчасовими:

- миттєвий вплив змінює значення параметра відразу і безповоротно;
- тимчасовий вплив лише на певний період часу змінює значення параметра, а по закінченню своєї дії повертає нормальне значення параметра.

Ефекти – це ядеякі довготривалі сутності, що змінюють параметри персонажів протягом свого часу дії.

За допомогою ефектів створюються взаємодії між ігровими параметрами. Із взаємодій формується ігрова механіка. А механіка контролює ігровий процес

Ефект – це «контейнер» для впливів. Усередині цього «контейнера» можуть міститися як миттєві, так і тимчасові впливи [15]:

- миттєвий вплив викликається періодично по таймеру (приклад: завдавати пошкодження кожні n секунд);
- тимчасовий вплив змінює параметр протягом дії ефекту.

Важливою характеристикою впливу є інтенсивність – величина, на яку змінюється параметр. Ефект так само володіє тривалістю, яка визначає час життя ефекту. У сукупності ці дві характеристики визначають, наскільки ефект значущий для гравця (збалансований).

Підвищуючи час тривалості або величину інтенсивності, підсилюється ефект. Звідси випливає, що значимість ефекту може залишатися однаковою, якщо одночасно змінювати тривалість та інтенсивність в різних напрямках (наприклад: підвищувати інтенсивність і знижувати тривалість). Ця аксіома – «Правило балансу №1» [13].

Звідси випиває перша і головна проблема балансу – знайти для кожного ефекту такі значення тривалості та інтенсивності, щоб живі гравці сприймали їх як збалансовані.

Існує окремий вид ефектів – стан. Якщо ефект змінює сукупність характеристик об'єкта і наділяє його новими властивостями (приклад: невидимість, нерухомість, оглушення), то такий ефект називається станом. Інтенсивність, як характеристика, відсутня в ефектах стану.

У спробах знайти співвідношення між двома характеристиками ефекту (інтенсивність і тривалість) таке, щоб ефект вважався «збалансованим», можливо прийти до нескінченного числа варіантів, що задовольняють правило балансу №1. Для цього можна підвищувати один параметр (наприклад: інтенсивність) і зменшувати інший (наприклад: тривалість):

- уповільнити ціль на 50% протягом 5 секунд;
- уповільнити ціль на 60% протягом 4 секунд;
- уповільнити ціль на 70% протягом 3 секунд;
- уповільнити ціль на 80% протягом 2 секунд.

Можна проблему нескінченних варіантів вирішити, якщо розділити всі ефекти на 2 типи:

- зміни тільки по тривалості (інтенсивність відсутня, чи постійна);
- зміни тільки по інтенсивності (тривалість завжди постійна);

Тепер дозволено змінювати тільки одну з характеристик ефекту, і, отже, умовно можна знайти шаблонні значення змінюваного параметра для всіх ефектів.

Потрібно скласти таблицю, де всі можливі впливи отримують одне єдине значення: змінювані по тривалості отримують значення шаблонної тривалості, а змінювані по інтенсивності мають завжди задану тривалість і отримують значення шаблонної інтенсивності (таблиця 1.2).

Таблиця 1.2 – Приклад побудови таблиці для балансування характеристик

Миттєвий вплив		
Пошкодження	600	інтенсивність
Стан		
Оглушення	4 сек.	тривалість
Невидимість	10 сек.	тривалість

З таблиці можна зробити деякі висновки. Якщо необхідно створити вміння для гравця, і це вміння повинно застосовувати «збалансований» ефект,

то теоретично не має значення, який з впливів обрати, так як вони рівноцінні. Якщо вони такими не являються, потрібно виправляти їх шаблонні значення.

Для теоретичного глобального балансу відсутня різниця, чи буде вміння наносити шкоди в 600 одиниць, оглушувати на 4 секунди або ж застосовувати ефект, що змінює атаку на 30% протягом n секунд, так як створено таблицю шаблонних значень і зроблено припущення про рівнозначність цих значень. В майбутньому гравці підкажуть, який вплив оцінено невірно з точки зору балансу.

Тут і з'являється правило балансу №2 – редагування балансу повинні бути глобальними, а не точковими. Точкові і локальні зміни створюють виключення, що призводять до безладу в системі, і якщо система досить велика, то безлад призводить до глобальних неконтрольованих проблем.

Отже, проаналізувавши методи балансування характеристик, та ознайомившись з методологію транзитивного балансування, було вирішено розробити формулу середнього пошкодження для персонажу, задля зручного і ефективного балансування шляхом порівняння занесених у таблицю результатів.

1.5 Аналіз методів автоматичної генерації квестів

Процедурна генерація – це метод алгоритмічного створення даних за допомогою комбінацій алгоритмів, поєднаних із випадковістю [16]. У комп’ютерній графіці він зазвичай використовується для створення текстур та 3D-моделей. У відеоіграх він використовується для автоматичного створення великої кількості контенту. Залежно від реалізації, переваги процедурної генерації можуть включати менший розмір файлу, більший обсяг контенту та випадковість для певного ігрового процесу.

На відміну від графічно орієнтованих відеоігор, ігри, жанр яких безпосередньо натхнений настільною рольовою грою “Dungeons & Dragons”, активно використовували процедурну генерацію так само, як це робили настільні системи. До таких ранніх ігор належать Beneath Apple Manor (1978) та

Rogue (1980). Ігри, що базуються на схожих концепціях, дозволяють розвивати складний ігровий процес, не витрачаючи зайвого часу на створення ігрового світу.

Процедурна генерація часто використовується в системах квестів ігор, таких як рольові екшн-ігри та масові багатокористувацькі рольові ігри. Незважаючи на те, що квести можуть мати фіксовані винагороди, інші предмети, наприклад зброя та броня, можуть бути створені для гравця на основі рівня персонажу, рівня квесту та інших випадкових факторів. Це часто призводить до того, що предмети мають різну рідкість, яка застосовується для відображення того, коли система генерації створила предмет з характеристиками, які перевищують середні. Наприклад, серія Borderlands базується на процедурній генерації, що дозволяє створити понад мільйон унікального озброєння.

На сьогоднішній день загальнодоступних генераторів квестів, якщо не зважати на зовсім примітивні варіанти, майже немає. Навіть не зважаючи на те, що питання автоматичної генерації завдань у іграх досить актуальне, особливо це стосується рольових ігор.

Квест у рольових відеоіграх — включно з багатокористувацькими онлайновими рольовими іграми (MMORPG) та їхніми попередниками, це завдання, яке керований гравцем персонаж, чи група персонажів може виконати за винагороду. Винагорода може включати підвищення досвіду персонажа задля вивчення нових навичок і вмінь, здобич або скарб, ігрова валюта, як-от золоті монети, доступ до нових місць або регіонів, чи ж будь-яка комбінація переліченого. У різних іграх квести несуть різний сенс, у деяких іграх квести є лише побічним способом видобутку ресурсів, в інших квести підігрівають інтерес гравців до зацікавленості в сюжетних лініях. Але загальним залишається одне – коли сюжетні, прописані автором квести закінчуються – інтерес гравця зникає, а щоденні завдання, що автоматично згенеровані, нагадують звичайні рутинні справи.

Отже, було вирішено побудувати генератор квестів із великою різноманітністю можливих варіантів.

1.6 Постановка задачі розробки

Після аналізу поточного стану питання та порівняння існуючих рішень за визначеними критеріями визначено завдання, які необхідно виконати, для розробки системи:

1. Розробити метод і алгоритм приведення характеристик до універсальної одиниці виміру;
2. Розробити метод і алгоритм побудування кривої «рівня-потужності»;
3. Розробити метод і алгоритм автоматизації методу обчислення характеристик ігрових предметів;
4. Розробити метод і алгоритм генерації варіативних квестів за допомогою орієнтованого графу;
5. Розробити програмні засоби для реалізації алгоритмів;
6. Розробити програмні засоби багатокористувацької рольової гри;
7. Розробити серверну частину Telegram-бота.

Технічне завдання наведено у Додатку А.

1.7 Висновки

У першому розділі було розглянуто актуальний стан питання програмних реалізацій Telegram-ботів. Досліджено предметну область, а також розглянуто існуючі аналоги та їх недоліки. Аналіз поточного стану питання показав, що розробка власної системи для багатокористувацької онлайн рольової гри є доцільною, оскільки вона буде містити весь необхідний функціонал, а також не матиме недоліків існуючих реалізацій.

Також було розглянуто актуальний стан питання програмних реалізацій Telegram-ботів. Досліджено предметну область, а також розглянуто існуючі аналоги та їх недоліки.

Проведено постановку задач розробки.

2 РОЗРОБКА МЕТОДІВ ТА АЛГОРИТМІВ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ІГРОВИХ МЕХАНІЗМІВ

2.1 Розробка методу автоматизації обчислення характеристик ігрових предметів

В ході дослідження було проаналізовані різні методи для балансування ігрових предметів. Як результат, було вирішено поєднати алгоритм перетворення характеристик предметів в єдину універсальну одиницю виміру та побудувати криву «рівень-потужність». Універсальна одиниця виміру в нашому випадку – це така одиниця, до якої приводяться всі характеристики предметів шляхом перемноження на коефіцієнт корисності, тобто

$$P = a_1 * k_1 + .. + a_{n-1} * k_{n-1} + a_n * k_n \quad (2.1)$$

де a – значення певної характеристика, k – відповідний коефіцієнт корисності. Коефіцієнти корисності встановлюються дослідним шляхом та в подальшому можуть бути скориговані.

В результаті дослідження та побудованої бойової системи крива «рівень-потужність» набула вигляду згладженої параболи. Основні точки кривої можна знайти за формулою:

$$x_y = kp * x_{y-1} \quad (2.2)$$

для випадку при $y > 1$, де y – рівень персонажа, x – нижня границя потужності для певного рівня, а kp – коефіцієнт приросту. Початкове значення x задається також дослідним шляхом, у нашому випадку $x = 80$. Таким чином, отримуємо нижню та верхню границі потужності для кожного рівня, тобто значення, за які не бажано виходити персонажу по його результуючим характеристикам.

Завдяки формулі приведення до універсальної одиниці та кривій «рівень-потужність» можна підрахувати коефіцієнти розподілу характеристик для кожного предмету кожного ігрового класу. Коефіцієнти корисності потрібно

перевести до протилежних значень, тобто, коефіцієнту розподілу, який відповідає найбільшому коефіцієнту корисності, присвоюється значення найменшого коефіцієнта корисності, і навпаки. Фактично, кожен коефіцієнт розподілу відповідає можливим універсальним одиницям виміру, які можуть бути витрачені на ту чи іншу характеристику предмета. Маючи кількість предметів, які може вдягнути персонаж, верхню та нижню границю потужності персонажа на певному рівні, можна побудувати таблиці, стовбці яких будуть відповідати типам предметів, рядки – характеристикам, а в значені комірки буде записано загальний коефіцієнт розподілу характеристики.

Формула підрахунку максимальної границі певної характеристики для предмета має вигляд:

$$X_{max} = \frac{M_{max} * kr}{n} \tag{2.3}$$

де X_{max} – верхня границя характеристик, M_{max} – верхня границя потужності, kr – коефіцієнт розподілу характеристики для певного класу предмета, n – кількість можливих предметів. Для нижньої границі формула аналогічна.

Отже, в результаті застосування запропонованих формул можна отримати декілька таблиць з коефіцієнтами розподілу за допомогою яких, підставивши будь-який предмет, котрому присвоєний клас, до якого він відноситься, та початковий рівень, з якого він може бути застосований, одержують границі його характеристик. Границі характеристик для кожного предмета підраховуються випадковим чином, з відхиленням від середнього на певний коефіцієнт, який може бути змінений для досягнення потрібної кінцевої мети [3].

2.2 **Розробка методу генерації варіативних квестів на основі орієнтованого графу**

Насамперед потрібно вирішити з типами квестів які підтримуватиме генератор. Хоч в різних іграх можуть бути різні типи квестів, всі вони зачасту

зводяться до декількох основних, які описують або квест в цілому, або одну з його частин [17]:

- доставка – завдання має за основну ціль доставити якийсь предмет або ж ресурс до певного персонажа;
- супроводження – цілю завдання є забезпечити безпеку когось під час переміщення з локації «a» до локації «b»;
- вбивство – знищення певної кількості істот, або ж якогось персонажа;
- пошук – сенс завдання полягає у пошуку певного предмета чи персонажа;
- виконання – потрібно виконати певну дію якусь кількість раз;
- переміщення – кінцевою точкою є знаходження гравця в якійсь локації.

Прикладом квесту котрий включає всі вище наведені типи завдань може бути:

- гравцю потрібно дістатись до локації «Локація 1»;
- знайти в «Локації 1» «Персонажа 1»;
- по проханню «Персонажа 1» гравець має вбити 10 «Істота 1»;
- після вбивства потрібної кількості «Істота 1» гравець має знайти «Предмет 1» та віднести його до «Персонажа 2»;
- гравець має супроводити «Персонажа 2» до «Локація 2»;
- в кінці гравець має здійснити «Дію 1» 2 рази щоб отримати винагороду.

Для кожного з типів завдань потрібні свої окремі конструктори, які приймають на вхід певні параметри та в залежності від них генерують відповідне завдання. Розглянемо будування конструктору на прикладі першого типу завдань «доставка». Основними параметрами конструктора є список відкритих гравцем локацій, рівень гравця та список відносин з персонажами.

За допомогою рівня гравця можна визначити список предметів які гравець зможе доставити, список локацій впливає на місця з яких та в які гравець буде доставляти предмет, а список відносин з персонажами впливає на

те до кого цей предмет буде доставлений, або в окремому випадку, у кого цей гравець візьме предмет для доставки.

Для побудови цільного квесту з одного або декількох завдань, було вирішено зберігати дані про квест у вигляді орієнтованого графу, де вузлами являються конкретні завдання, а ребрами – перехід з одного завдання до іншого, тобто послідовність виконання завдань. При створенні графа потрібно притримуватись певних умов:

1. З кожної вершини графа, крім кінцевої, повинна бути можливість дістатись хоча б у одну іншу.

2. У графа обов’язково має бути початкова вершина, яка може містити умову початку квесту, та хоча б 1 кінцева.

3. Кожний квест має бути здійсненний, тобто такий, який персонаж може виконати.

Вершини графа діляться на декілька типів:

- початок - єдина точка входу в квест;
- кінець - маркер завершення квесту;
- точка вибору - вершина, в якій герой повинен зробити вибір подальшого шляху розвитку подій;
- точка умовного переходу – вершина, у якому подальший шлях визначається яким-небудь динамічним параметром (наприклад, кількістю грошей у гравця);
- звичайна вершина – вершина, яка не має додаткових властивостей.

Отже, в даній роботі було розглянуто алгоритм генерації варіативних квестів на основі орієнтованого графу. Даний генератор дозволить розширити різноманіття квестів, внесе у гру більше розмаїття та зможе унікалізувати історію головного персонажа кожного гравця [4].

2.3 Розробка методу балансування характеристик персонажів

У всіх проаналізованих аналогах різниця в значеннях характеристик між рівнями персонажів занадто велика. Особливо це відчутно у грі RF між 49-м та

50-м рівнем. Так, гравці 49-го рівня не мають майже жодної можливості перемогти гравця 50-го рівня. Для більш збалансованої прогресії характеристик персонажів було вирішено вивести формулу середнього пошкодження персонажем другого персонажа.

Враховуючи розроблену бойову систему, формула для балансування характеристик отримала такий вигляд:

$$DamM = DamA - AD + CritA - EvaD + ComboDamA/n, \quad (2.4)$$

де DamA – середнє фізичне або магічне базове пошкодження персонажа в залежності від типу використаної зброї, яке розраховується за формулою:

$$DamA = (DamAMin + DamAMax)/2; \quad (2.5)$$

де AD – різниця між захистом та проникненням захисника та атакуючого:

$$AD = DefD - PenA; \quad (2.6)$$

де $CritA$ – середня надбавка від критичного пошкодження, що розподілена на всю довжину бою:

$$CritA = \big((DamA * cofC) - DamA\big) * chanceCritA; \quad (2.7)$$

де $cofC$ – в скільки разів критична атака більша за звичайну, а $chanceCritA$ – шанс, з яким персонаж може нанести критичну атаку; $EvaD$ – кількість пошкоднення, від якого може ухилитись захисник:

$$EvaD = (DamA - AD + CritA) * (chanceEvaD - AccuracyA); \quad (2.8)$$

де $AccuracyA$ – точність персонажа, яка залежить від навика володіння зброєю, а $chanceEvaD$ – шанс захисника ухилитись; *ComboDamA* – середня надбавка від посиленого пошкодження, яка збільшує базову атаку та може відбутись, якщо атакуючий персонаж проведе успішно *n* атак підряд:

$$ComboDamA = (DamA * ComboCoef - DamA) * ((100\% - chanceEvaD + AccuracyA)^n + \left((chanceEvaD - AccuracyA) * TempAttackA\right)^n) * (1 + cofC * chanceCritA); \quad (2.9)$$

де $ComboCoef$ – коефіцієнт збільшення базової атаки, $TempAttackA$ – шанс зберегти ланцюжок успішних атак при неуспішно проведеній.

За допомогою цих формул можливо з легкістю контролювати баланс характеристик персонажів на різних рівнях, та надати гравцям можливість з доволі непоганим шансом перемогти когось, хто на 1-4 рівня більший за твій [5].

2.4 Розробка інтерфейсу за допомогою Telegram API

Telegram API є HTTP-інтерфейсом для роботи з ботами в Telegram. Всі запити до Telegram API повинні здійснюватися через протоколи HTTPS. Допускаються GET і POST запити. Для передачі параметрів в Bot API доступні 4 способи:

- запит в URL;
- application / x-www-form-urlencoded;
- application / json (не підходить для завантаження файлів);
- multipart / form-data (для завантаження файлів).

Відповідь приходить у вигляді JSON-об'єкта, в якому завжди буде булеве поле «ok» і опційне рядкове поле «description», що містить зручний для читання опис результату. Якщо поле «ok» істинне, то запит пройшов успішно і результат його виконання можна побачити в полі «result». У випадку помилки поле «ok» дорівнюватиме «False», а причини помилки будуть описані в полі «description». Крім того, у відповіді буде цілочисельне поле «error_code» [18].

Основні елементи інтерфейсу ботів універсальні і властиві кожному з додатків для обміну повідомленнями, проте у кожного з них є свої особливості, для Telegram вони такі:

- текстове повідомлення – 4096 UTF символів;
- картинка – до 1GB, підпис до 200 символів;
- відео – до 1.5GB, будь якого формату;
- аудіо – надано можливість прослуховувати в месенджері;
- зникаючі кнопки (або Швидкі відповіді);
- прикріплені файли;
- постійні кнопки;
- постійне меню – представлено списком команд.

Telegram API дозволяє реалізувати інтерфейс за допомогою вбудованого методу «send_message()», який приймає як параметри унікальний ідентифікатор чату користувача, текст повідомлення та всі інші перераховані в списку вище елементи.

Хоча можливостей доволі мало, проте цього більш ніж достатньо для того, щоб зробити зручний інтерфейс для текстової гри.

2.5 Розробка структури інтерфейсу для ігрового Telegram боту

Загальна структура інтерфейсу головного вікна чат-ботів відовідає структурі будь-яких чатів у месенджері. Вона складається із панелі статусу, робочої області та панелі інструментів, яка складається з кнопок для відправлення повідомлень з різними вкладеннями.

Панель статус містить піктограму, кнопку виходу, кнопку меню налаштувань та коротку інфморацію про бота.

При натиснені на панель статусу відкривається сторінка з повною доступною інформацією про бота. Окрім піктограми та назви в ній є кнопка виключення сповіщень, ім’я користувача та повний опис боту, наданий розробником.

Меню налаштувань дає можливість мінімального керування ботом, а саме пошуку по повідомленнях, відправки телефону, очищення історії, відключення сповіщень та видалення чату.

При розробці інтерфейсу програмного продукту було використано такі елементи керування: текстові повідомлення, іноді з прикріпленими картинками, декілька варіантів клавіатури з постійними кнопками, які змінюються в залежності від ситуації, та постійне меню, складене з основних потрібних користувачу команд. Інтерфейс програмного засобу містить в собі лише необхідну, в тій чи іншій ситуації для користувача, інформацію та кнопки керування.

На рисунках 2.1 та 2.2 зображені розроблені структурні схеми інтерфейсів вікон програмного додатку.

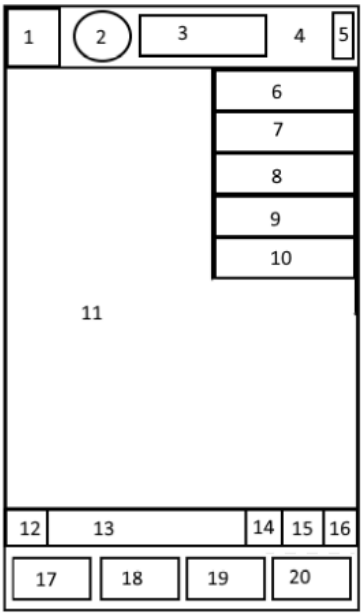


Рисунок 2.1 – Графічна схема головного вікна боту

Основні елементи інтерфейсу головного вікна боту «FireBox»:

1. Кнопка виходу.
2. Піктограма боту.
3. Назва боту.
4. Панель статусу.
5. Кнопка «Меню».
6. Пункт «Пошук».
7. Пункт «Відправити телефон».

8. Пункт «Очистити історію».
9. Пункт «Вимкнути сповіщення».
10. Пункт «Видалити чат».
11. Робоча область.
12. Кнопка для відправки емоцій та “.gif” анімацій.
13. Поле набору текстових повідомлень.
14. Кнопка переключення між клавіатурами.
15. Кнопка прикріплення файлів.
16. Кнопка запису аудіо та відео повідомлень.
17. Клавіша «Профіль»
18. Клавіша «Місто».
19. Клавіша «Пошук монстрів».
20. Клавіша «Атакувати».

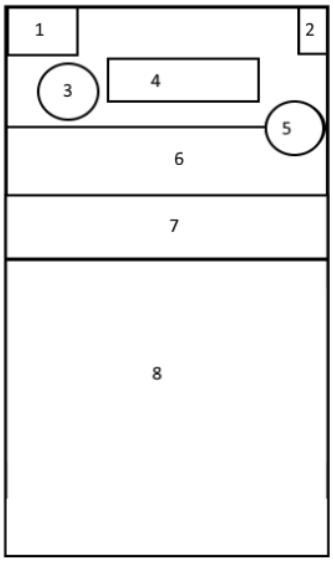


Рисунок 2.2 – Графічна схема вікна інформації про бота

Основні елементи інтерфейсу головного вікна боту «FireBox»:

1. Кнопка виходу.
2. Кнопка «Меню».
3. Піктограма боту.
4. Назва боту.

5. Кнопка «Головне вікно».
6. Ім’я користувача.
7. Кнопка «Виключити сповіщення».
8. Панель інформації про бота.

Розробка інтерфейсу є важливим етапом у процесі розробки чат-ботів, оскільки ефективність та зручність роботи користувача з ботом є вирішальним чинником, що визначає успішність розробленої системи.

2.6 Розробка блок-схем методів та алгоритмів для роботи ігрового Telegram боту

Розроблена блок-схема алгоритму для автоматичного підрахунку характеристик предмету. Блок-схема складається з 6 блоків, на ній зображений процес підрахунку характеристик для одного предмету за допомогою кривої «рівня-потужності». Вхідні дані: список коефіцієнтів, можлива кількість предметів, рівень предмету та максимально і мінімально допустима потужність по цьому рівню.

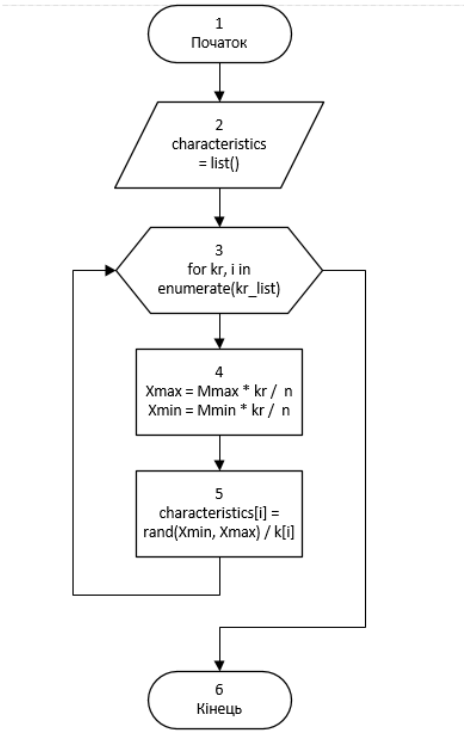


Рисунок 2.3 – Блок-схема алгоритму підрахунку характеристик предмету

Опис алгоритму:

Крок 1-2. Початок, ініціалізація списку характеристик.

Крок 3. Запуск циклу по списку коефіцієнтів з відслідковуванням індексу поточного елементу.

Крок 4. Підрахунок максимальної та мінімальної границі потужності певної характеристики за формулою 2.4.

Крок 5-6. Генерація випадкового значення між вирахованими границями і перевід з універсальної одиниці виміру в конкретне значення для певної характеристики та запис у ініціалізовний раніше список, кінець.

Блок-схема алгоритму автоматичної генерації квестів зображена на рисунку 2.4.

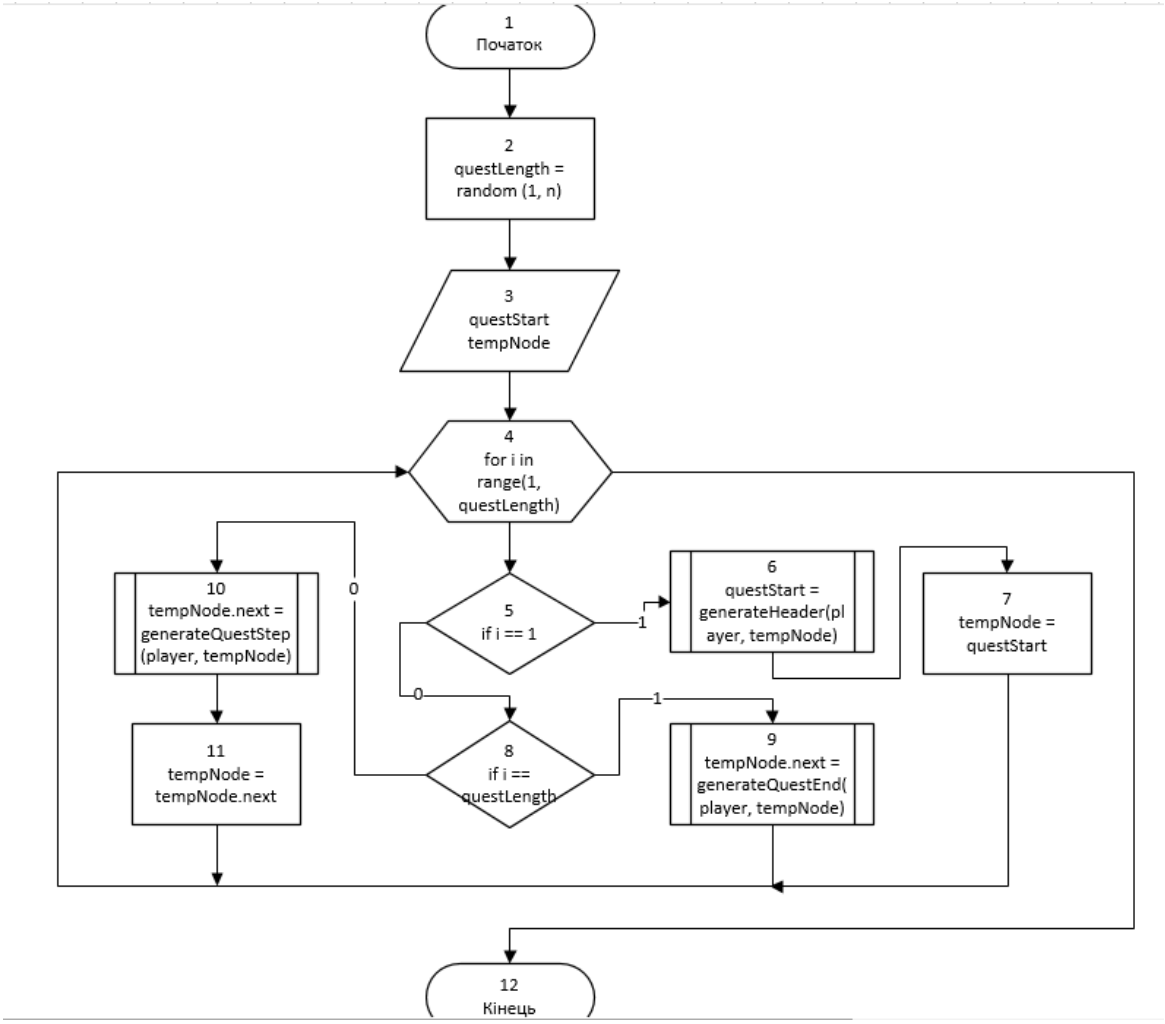


Рисунок 2.4 – Блок-схема алгоритму генерації квестів

Розроблена блок-схема складається з 12 блоків, серед яких присутні 3 процедурні блоки для виклику допоміжних функцій, котрі генерують різні типи завдань квесту в залежності від позиції завдання в орієнтованому графі. Граф побудовано за допомогою зв’язного списку.

Опис алгоритму:

Крок 1-2. Початок, визначається довжина квесту, від 1 до змінної «n».

Крок 3. Ініціалізація вказівника на початок списку та проміжного.

Крок 4. Запуск циклу від 1 до сгенерованої довжини квесту.

Крок 5. Якщо ітератор дорівнює 1 (тобто чи це перше завдання в квесті) то переходимо до кроку 6, інкаше переходимо до кроку 8.

Крок 6. Викликаємо функцію яка відповідає за генерацію завдання для початку квесту, як параметри передаємо об’єкт гравця та проміжний вказівник. Результат у вигляді адреси на об’єкт першого завдання присвоюємо до вказівника на початок квесту.

Крок 7. В проміжний вказівник присвоюємо вказівник на почтаку квесту.

Крок 8. Перевіряємо чи ітератор дорівнює загальній довжині квесту, якщо так, то переходимо до кроку 9, інакше до кроку 10.

Крок 9. Викликаємо функцію яка відповідає за генерацію завдання для кінцівки квесту, як параметри передаємо об’єкт гравця та проміжний вказівник. Результат у вигляді адреси на об’єкт останього завдання присвоюємо до атрибута проміжного вказівника, який посилається на наступний вказівни.

Крок 10. Викликаємо функцію яка відповідає за генерацію звичайного завдання, як параметри передаємо об’єкт гравця та проміжний вказівник. Результат у вигляді адреси на об’єкт завдання присвоюємо до атрибута проміжного вказівника, який посилається на наступний вказівник.

Крок 11-12. Проміжному вказівнику присвоюємо адресу яка зберігається в його атрибуті, кінець.

Результатом виконання алгоритму є зв’язний список в якому зберігаються посилання на всі завдання згенерованого квесту у суворо визначеному порядку.

Бойова система PVP та PVE складається з двух етапів, а саме з почергового нападу кожної зі сторін. Цей цикл триває доки один з супротивників не помре. Результат поєдинку і сам процес підраховується автоматично, а по закінченю гравцям приходить повідомлення з повним описом минулого поєдинку. Побудована блок-схема зображена на рисунках 2.5 та 2.6.

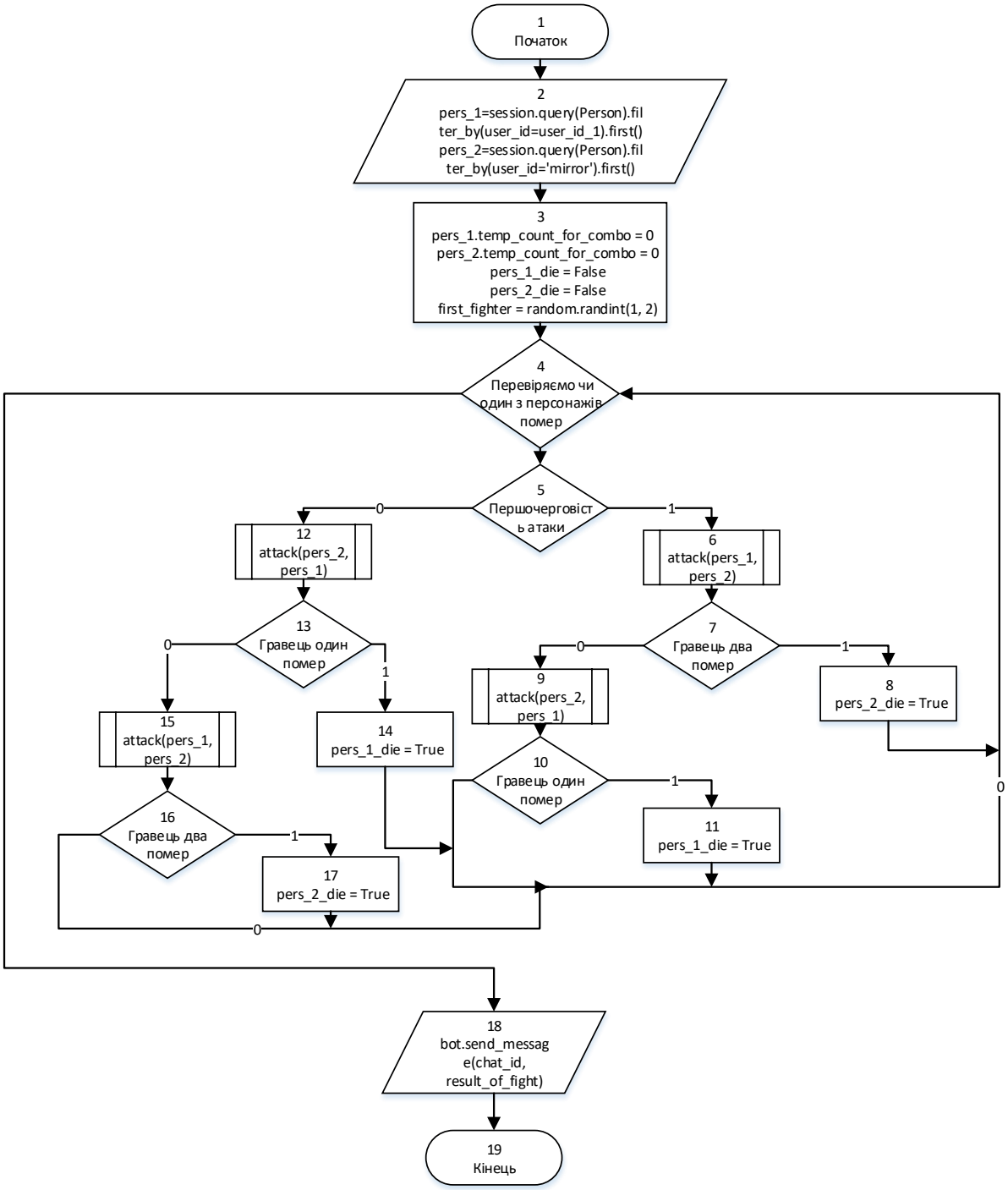


Рисунок 2.5 – Алгоритм PVP та PVE ч.1.

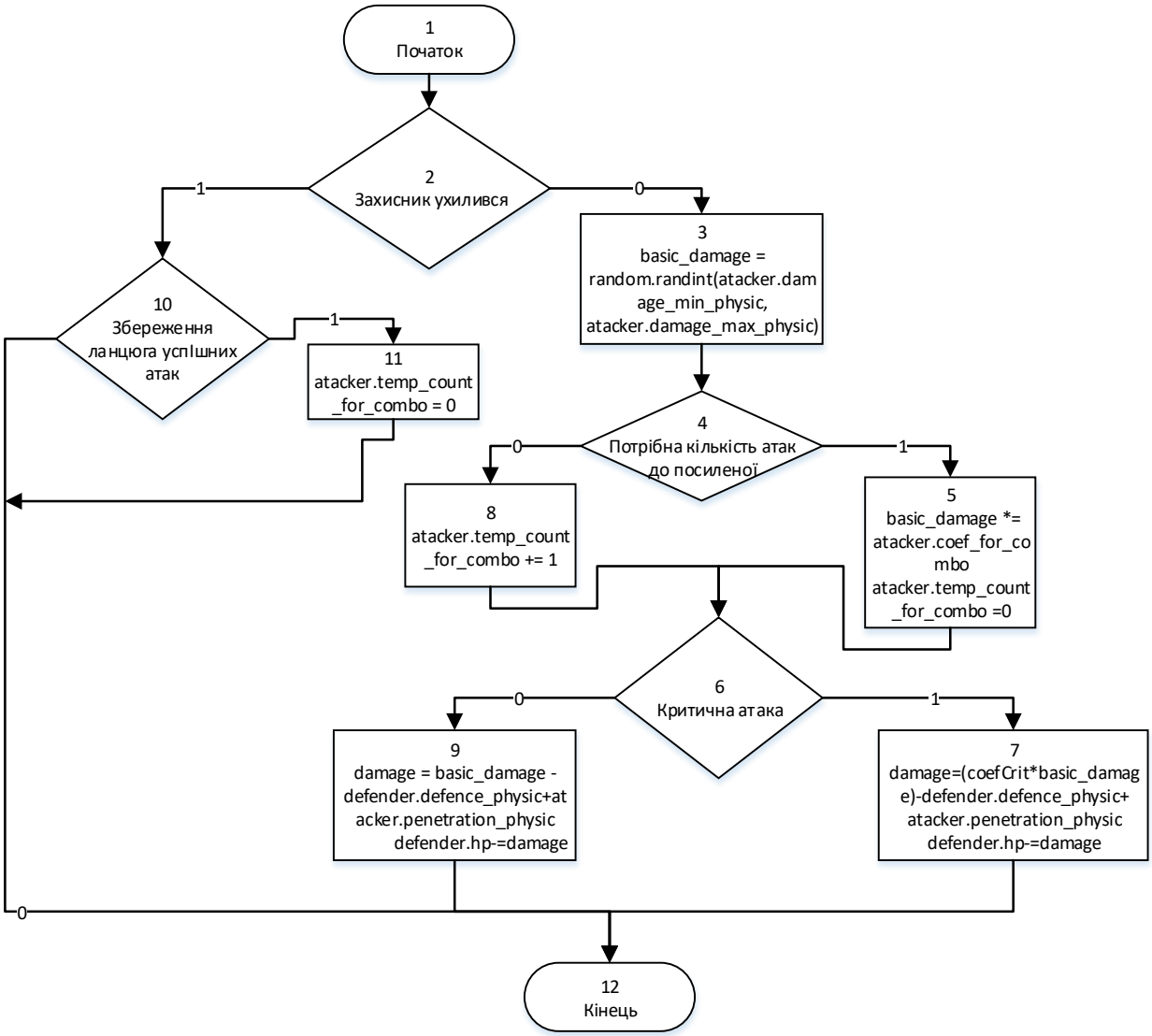


Рисунок 2.6 – Алгоритм PVP та PVE ч.2.

Алгоритми PVP та PVE складаються з 19 блоків, блоки 6, 9, 12 та 15 відповідають розробленому алгоритму функції нападу на рисунку 2.6. Опис алгоритму:

Крок 1-2. Початок, дістаємо з бази даних учасників поєдинку.

Крок 3. Обнулюємо лічильники ланцюгів успішних влучень, створюємо змінні визначення смерті противників і визначаємо першочерговість ходу.

Крок 4. Запуск циклу, тривалістю доки один із супротивників не загине.

Крок 5. Визначаємо першочерговість ходу.

Крок 6. Виклик функції нападу.

Крок 7. Перевірка на те чи залишився живим захисник.

Крок 8. Якщо було вбито захисника, значення змінної, що відповідає за його смерть встановлюється, як істинно.

Крок 9. Якщо захисник живий, виконуємо функцію нападу, змінивши місцями супротивників.

Крок 10. Перевірка на те чи залишився живим перший супротивник.

Крок 11. Якщо перший супротивник загинув, значення змінної, що відповідає за його смерть встановлюється, як істинно.

Кроки 12-17. Аналогічні дії до кроків 7-11 зі змінною черги нападу.

Крок 18-19. Повідомлення про результат бою гравцям, кінець

Алгоритм функції нападу складається з 12 блоків. Опис алгоритму:

Крок 1-2. Початок, перевірка чи захисник ухилився.

Крок 3. Якщо захисник не ухилився, вираховується базове пошкодження шляхом отримання випадкового значення між мінімальним та максимальним.

Крок 4. Перевірка на потрібну кількість влучень по супротивнику.

Крок 5. Якщо кількість влучень рівна, то базове пошкодження множиться на коефіцієнт посиленого пошкодження та зкидується лічильник, інакше переходимо до кроку 8.

Крок 6-7. Перевірка на критичне влучення. Якщо істино, то результуюче пошкодження множиться на коефіцієнт критичного пошкодження.

Крок 8. Збільшуємо лічильник влучень на 1 та виконуємо кроки 6-7.

Крок 9. Від результуючого пошкодження віднімається значення опору захисника та додається значення проникнення нападаючого.

Крок 10-12. Якщо захисник ухилився, перевірка на те, чи зберігся ланцюг успішних атак, якщо ні – обнуляємо лічильник, кінець.

Отже, розроблено 4 блок-схеми які описують принцип роботи основних алгоритмів розроблених методів, а саме блок-схема алгоритму генерації квестів, блок-схема алгоритму підрахунку характеристик згенерованих предметів, та блок-схеми алгоритмів PvP та PvE, одна з яких описує основний алгоритм, а друга процедурний блок котрий викликається у основному

алгоритмі та відповідає за реалізацію функції почергового нападу кожного з персонажів.

2.7 Висновки

У другому розділі було розроблено основні методи та їх алгоритми для роботи програмного продукту та відповідні формули, а саме:

- метод обчислення характеристик ігрових предметів та алгоритм для автоматизації розробленого методу;
- метод та алгоритм генерації варіативних квестів на основі орієнтованого графу;
- алгоритми PvP та PvE (гравець проти гравця та гравець проти навколишнього середовища);
- формула для балансування характеристик персонажів на різних рівнях.

Для алгоритмів було побудовано блок-схеми, котрі описують покрокову реалізацію кожного з них.

Розроблені методи та алгоритми характеризуються великою швидкодією, оптимізацією роботи чат-боту та підвищують ефективність ігрових механізмів для багатокористувацьких рольових ігор на базі месенджеру «Telegram» та ігор подібного жанру в цілому.

Також було розроблено структури інтерфейсу для ігрового «Telegram» боту, котра відповідає стандартам «Telegram API», яке використовується для розробки інтерфейсів чат-ботів. Розроблений інтерфейс характеризується мінімалізмом та згрупованістю керуючих елементів.

3 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТ ДЛЯ ІГРОВОГО TELEGRAM-БОТУ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу

Для розробки будь-якого програмного продукту вибір мови програмування є дуже важливим, оскільки від цього залежить простота реалізації алгоритму роботи, а також ефективність і швидкість роботи програми. Для реалізації даного програмного засобу необхідно обрати таку мову програмування, яка підійде найкраще, і буде мати всі потрібні бібліотеки для працювання з базою даних, багатопоточністю, математичними формулами та TelegramAPI. Порівнюватися будуть розповсюджені 4 мови (C++, C# та Java та Python), серед яких обиратиметься одна, ефективність якої для даної розробки є найвищою.

C++ — мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної [19].

C++ дуже швидка мова програмування, її синтаксис заснований на синтаксисі мови C.

Нововведеннями C++ порівняно з C є:

- підтримка об'єктно-орієнтованого програмування через класи;
- простори імен;
- вбудовані функції;
- підтримка узагальненого програмування через шаблони;
- доповнення до стандартної бібліотеки;
- додаткові типи даних;
- обробка винятків;
- перевантаження операторів;
- перевантаження імен функцій;
- посилання і оператори управління вільно розподіленою пам'яттю [20].

Java – об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи [21].

Особливості мови Java:

- синтаксис мови простий, об'єктно-орієнтовний та звичний;
- реалізація безвідмовна та безпечна;
- незалежна від архітектури та переносна;
- висока продуктивність виконання;
- інтерпретована, із динамічним зв'язуванням модулів.

C# — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML [22]. Перейнявши багато що від своїх попередників — мов C++, Object Pascal, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів [23].

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Із переваг цієї мови можна виділити [24]:

1. Інтерпретатор Python реалізований практично на всіх платформах та операційних системах.

2. Розширюванність мови (є можливість удосконалювати мову усіма зацікавленими програмістами).

3. Інтерпретатор написаний на C і вихідний код доступний для будь-яких маніпуляцій. У випадку необхідності можна вставити його в свою програму та використовувати як вбудовану оболонку. Або ж, написавши на C свої доповнення, отримати “розширений” інтерпретатор з новими

можливостями.

4. Наявність великої кількості модулів, що підключаються до програми, які забезпечують різноманітні додаткові можливості.

5. Простота та гнучкість мови. Завдяки їм, Python може використовуватись користувачами (математиками, фізиками, економістами та ін.), що не є програмістами, але використовують обчислювальну техніку в своїй роботі [25].

В таблиці 3.1 наведено результати порівняння розглянутих мов програмування за обраними критеріями.

Таблиця 3.1 – Порівняння мов програмування

Назва критерію	C#	C++	Java	Python
Глобальні змінні	1	1	0	1
Збирач сміття	1	0	1	1
Наявність бібліотек для Telegram API	1	0	1	1
Метапрограмування	1	1	1	1
Простота та зручність розробки	1	0	1	1
Розширюванність	0	0	0	1
Сумарний коефіцієнт	5	2	4	6

Згідно таблиці 3.1, мова програмування python має перевагу перед іншими мовами, тому для реалізації програмного додатку було обрано саме її, адже вона задовольняє всі необхідні вимоги для створення Telegram-боту і має низку бібліотек, що значно полегшує розробку програмних продуктів подібного типу.

Враховуючи всі вищенаведені переваги, можна сказати, що мова програмування Python є найбільш доцільною для розробки ігрового Telegram-боту.

3.2 Вибір середовища розробки та СКБД

Через те, що бот передбачає збереження великої кількості даних, потрібно використовувати базу даних. Для проекта було проаналізовано дві найпоширеніших СКБД.

MongoDB – документо-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів [26].

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів

Переваги використання MongoDB:

- можливість створювати документи, не задаючи їх структуру заздалегідь;
- кожен документ може мати власну структуру;
- у кожної бази даних може бути власний синтаксис;
- можливість додавати поля прямо під час роботи з даними.

MySQL – вільна система керування реляційними базами даних, вона була розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування [27]. Із переваг можна виділити:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно

працюють із БД;

- кількість рядків у таблицях може досягати 50 млн.;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

Враховуючи те, що важливість збереження та структуризації даних користувачів являється більш пріорітетним, ніж збільшення швидкості в деяких випадках, та наявність загальної структури для всіх класів, що виключає потребу в гнучкій базі даних, було вирішено використовувати СКБД MySQL [28].

Окрім СКБД постала дилема вибору інструментарію SQL. Вибір пав на SQLAlchemy.

SQLAlchemy – інструментарій SQL та об'єктно-реляційне відображення для мови програмування Python, випущене під ліцензією MIT. SQLAlchemy надає повний набір шаблонів корпоративного рівня стабільності, сконструйованих для високопродуктивного доступу до бази даних, написаних простою мовою Python. Філософія SQLAlchemy стверджує, що бази даних SQL поводяться тим менш подібно на колекції об'єктів, чим більше починають важити розмір та продуктивність, і навпаки, колекції об'єктів починають поводитись тим менш подібно на таблиці і записи, чим більш починає важити рівень абстракції. Тому, було впроваджено шаблон Data mapper (подібний на Hibernate для Java) замість шаблону active Record, який використовується в багатьох інших об'єктно-реляційних відображеннях. Проте, додаткові плагіни, такі як Elixir та declarative дозволяють користувачам розробку з декларативним синтаксисом [29].

Ще одним не менш важливим кроком для ефективної розробки є вибір інтегрованого середовища розробки. Інтегроване середовище розробки (IDE) – це програма, призначена для розробки програмного забезпечення.

IDE об'єднує інструменти спеціально призначені для розробки програмних продуктів. Ці інструменти зазвичай включають редактор для

роботи з кодом, інструменти збірки, виконання та налагодження, систему управління версіями.

PyCharm – одна з кращих повнофункціональних IDE, призначених саме для Python. Загалом це середовище розробки підтримує різні типи проектів, має модуль для завантаження потрібних бібліотек та керування версіями, а також має зручний «debug» режим. Існує як безкоштовний open-source (Community), так і платний (Professional) варіанти IDE. PyCharm доступний на Windows, Mac OS X і Linux [30]. Приклад інтерфейсу зображено на рисунку 3.1.

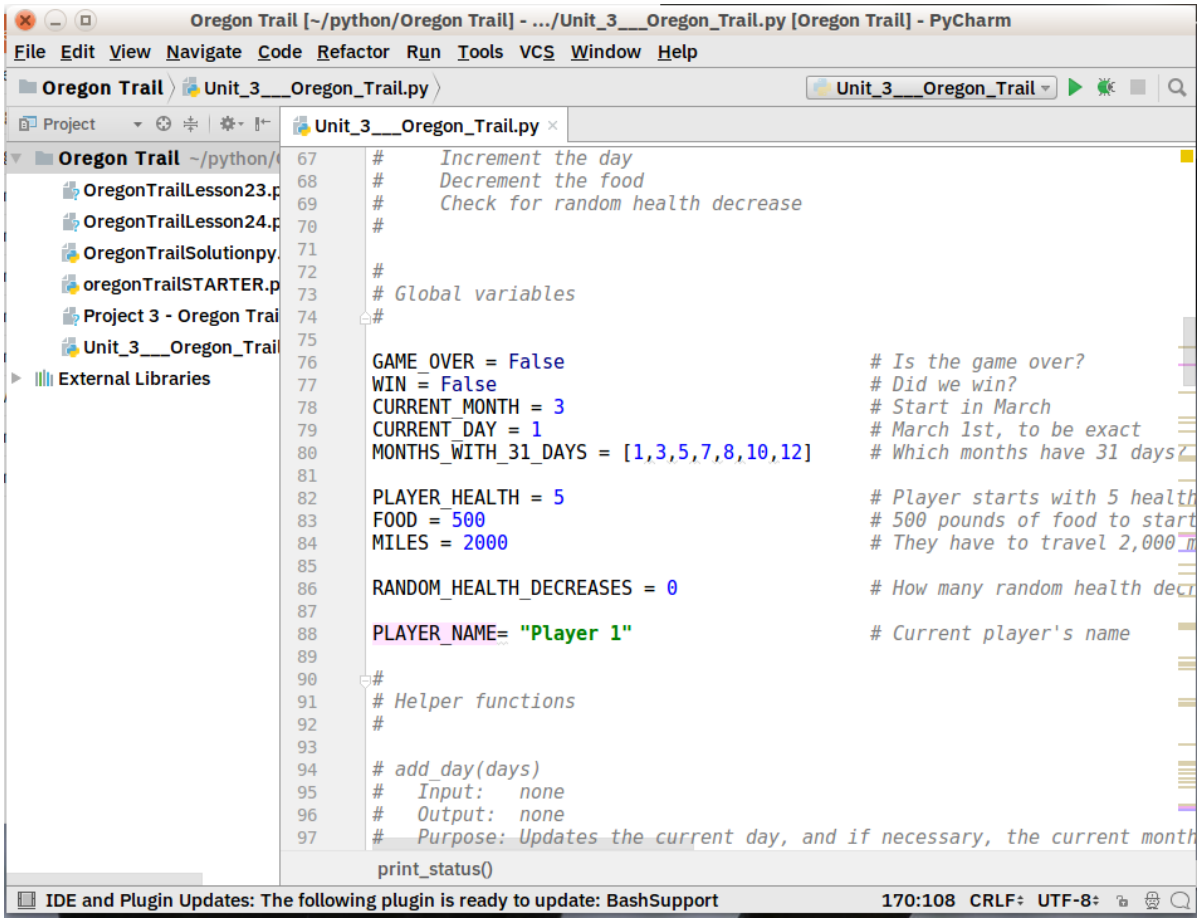


Рисунок 3.1 – Приклад інтерфейсу PyCharm

Spyder – open-source IDE для Python, оптимізована для машиного навчання. Spyder йде в комплекті з менеджером пакетів Anaconda. Spyder добре взаємодіє з такими бібліотеками як SciPy, NumPy і Matplotlib.

Spyder має функціональність стандартної IDE, на зразок редактора коду з

підсвічуванням синтаксису, автодоповнення коду і навіть вбудованого оглядача документації.

Відмінною особливістю Spyder є наявність провідника змінних. Він дозволяє переглянути значення змінних у формі таблиці прямо всередині IDE. Також добре працює інтеграція з IPython / Jupyter [31].

Про Spyder можна сказати, що він більш «приземлений», ніж інші IDE. Його можна розглядати як інструмент для певної мети, а не як основне середовище розробки. Загалом це безкоштовне, open-source і доступне на Windows, macOS і Linux середовище розробки. Приклад інтерфейсу зображено на рисунку 3.2.

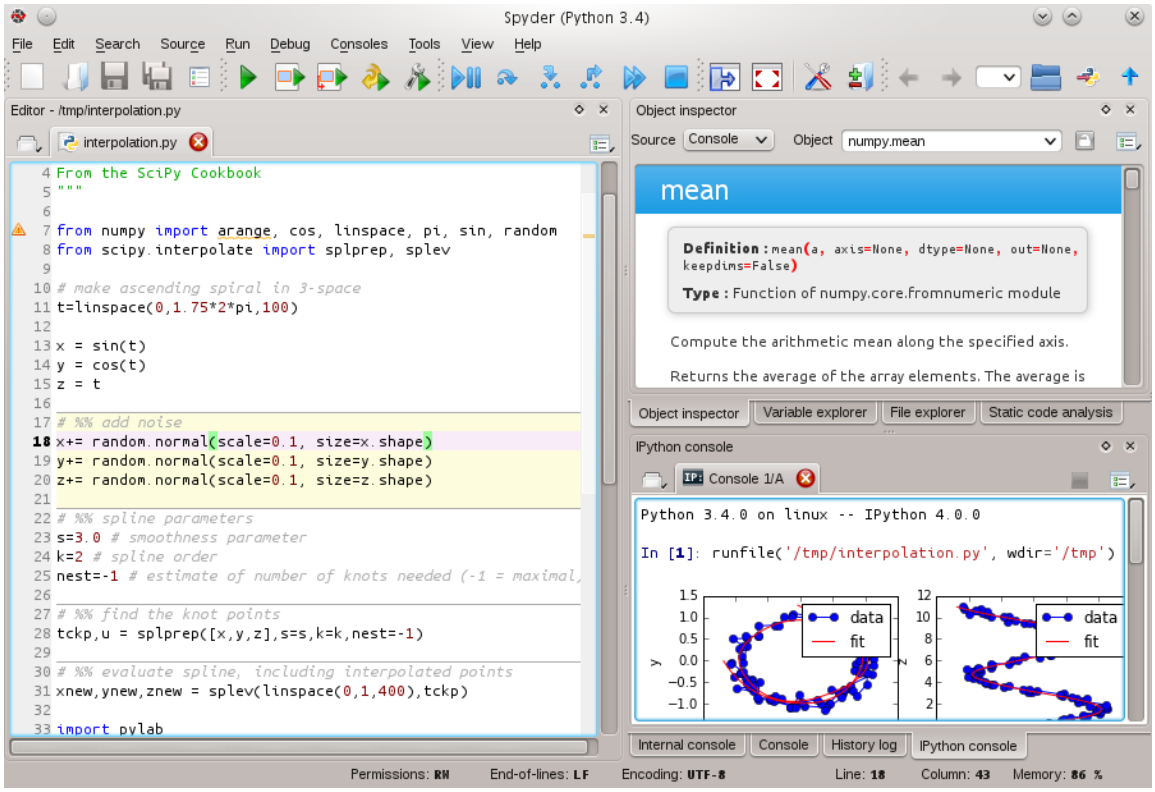


Рисунок 3.2 – Приклад інтерфейсу Spyder

Visual Studio – повнофункціональна IDE від Microsoft, яка багато в чому схожа з Eclipse. Доступна на Windows і Mac OS, Visual Studio представлена як в безкоштовному (Community), так і в платному (Professional і Enterprise) варіантах. Visual Studio дозволяє розробляти програми для різних платформ і надає свій власний набір розширень [32].

Python Tools for Visual Studio (PTVS) дозволяє писати на Python в Visual Studio і включає в себе Intellisense для Python, режим налагодження та інші інструменти. Приклад інтерфейсу зображено на рисунку 3.3.

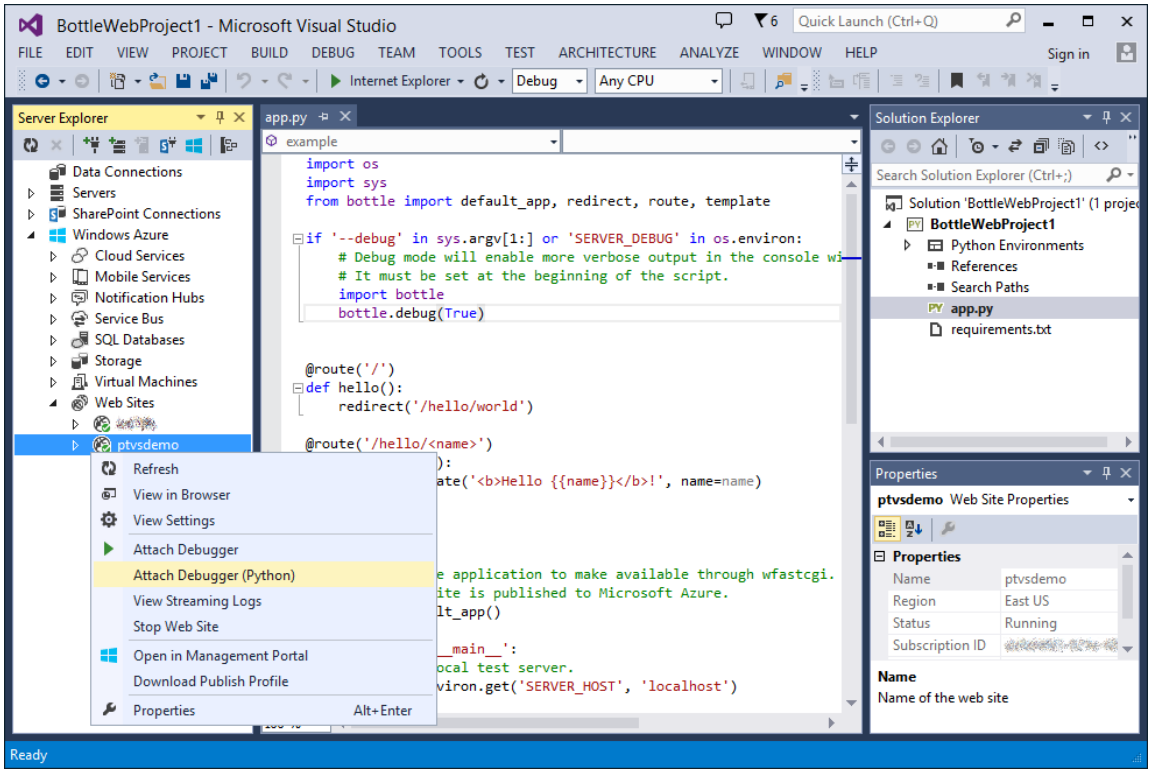


Рисунок 3.3 – Приклад інтерфейсу Visual Studio

В таблиці 3.2 наведено результати порівняння IDE для мови Python.

Таблиця 3.2 – Порівняння засобів розробки

Назва критерію	Microsoft Visual Studio	PyCharm	Spyder
Крос-платформеність	-	+	+
Простота у налагоджені	-	+	+
Автодоповнення	+	+	+
Система контролю версій	+	+	-
Переглядач змінних	-	-	+
Функціональність	+	+	-
Сумарний коефіцієнт	3	5	4

В результаті проведення аналізу засобів розробки програмного додатку сумарний коефіцієнт PyCharm – 5, що переважає коефіцієнти інших аналогів, також оптимальним є використання СКБД MySQL та SQLAlchemy, тому дані інструменти є оптимальними для даної розробки та вирішення поставленої задачі.

3.3 Програмна реалізація ігрового боту

Під час програмної реалізація проекту було розроблено велику кількість алгоритмів, серед яких:

– алгоритм передачі повідомлень між сервером та серверами Telegram;
– алгоритм переміщення персонажа між локаціями;
– алгоритм автоматичної генерації квестів;
– алгоритм генерації характеристик предметів;
– алгоритми PVP та PVE;
– генератор унікальних персоналізованих речей;
– алгоритм для балансування характеристик персонажів.

Спочатку у файлі «config.py» ініціалізуються всі допоміжні змінні для успішного встановлення з’єднання з БД та серверами Telegram. У файлі «server.py» створюється головна функція «__main__», в якій ініціалізуються всі створенні класи, та запускається опитування серверів Telegram (рисунок 3.4).

```
if __name__ == '__main__':
    Base.metadata.create_all(engine)
    while True:
        try:
            bot.polling(none_stop=True)
        except Exception as err:
            logging.error(err)
            time.sleep(5)
            print "Internet error!"
```

Рисунок 3.4 – Лістинг функції «__main__»

Опитування відбувається безперервно. У випадку помилок, ведуться записи деталей, та опитування продовжується.

Для аналізу команд та повідомлень від користувачів, використовуються функції-обробники, котрі викликаються або ж при конкретній команді від бота, або при повідомлені певного типу, як приклад, команда «/start» викликає функцію «send_welcome» зображену на рисунку 3.5.

```
@bot.message_handler(commands=['start'])
def send_welcome(message):
    if get_person(message.from_user.id):
        bot.send_message(message.chat.id, "Ваш персонаж вже існує")
    else:
        bot.send_message(message.chat.id, "Вперше тебе бачу, введи своє ім'я (команда /name (ім'я)")
```

Рисунок 3.5 – Лістинг функції «send_welcome»

Одним з найважливіших алгоритмів в іграх є алгоритм переміщення між локаціями, адже персонаж майже завжди рухається, тому цей алгоритм має бути добре оптимізованим, не завантажувати сервер, працювати швидко та злагодженно. Для дотримання цих умов було вирішено використовувати систему таймерів, адже на відміну від функції sleep(), яку часто використовують для того, щоб затримати виконання певних дій, таймер відкладує виконання функції на певний час, що не завантажує сервер. Окрім цього на цей час потрібно не дати змогу персонажу робити що-небудь ще, тому його потрібно перевести в стан виконання дії, який забороняє виконувати будь-які операції окрім перегляду інформації про персонажа.

Загалом розроблений алгоритм працює наступним чином: при отримані повідомлення про переміщення в певну локацію, дані від персонажу проходять валідацію, і при успішному проходжені викликається функція «move_to», яка приймає як параметри, назву локації, ідентифікатор користувача та чату (рисунок 3.6).

Функція «move_to» викликається в асинхронному режимі, що дозволяє зменшити навантаження на сервер, а гравцю виконувати інші дії поки персонаж буде рухатись між локаціями. Після збору потрібної інформації створюється і запускається таймер «t = Timer(time_to.time, wait, args=[pers, location_name, chat_id])», який приймає як аргументи функцію «wait», час переміщення

персонажа між локаціями та аргументи, які потрібно передати в функцію. Після запуску таймеру він відраховує потрібний час і виконує завершуючу переміщення функцію, яка встановлює потрібні дані персонажу, відкриває йому доступ діям та видає відповідні повідомлення (рисунок 3.7).

```
@app.task
def move_to(location_name, user_id, chat_id):
    pers = session.query(Person).filter_by(user_id=user_id).first()
    previous_loc = pers.location if pers.location else u"Місто"
    time_to = session.query(Time).filter_by(from_=previous_loc, to_=location_name).first()
    if not time_to:
        bot.send_message(chat_id, u"Шлях не знайдено")
    bot.send_message(chat_id, u"Ти відправився в шлях")
    pers.monster_for_fight = None
    pers.is_in_action = True
    session.add(pers)
    session.commit()
    t = Timer(time_to.time, wait, args=[pers, location_name, chat_id])
    t.start()
```

Рисунок 3.6 – Лістинг функції «move_to»

```
def wait(pers, location_name=None, chat_id=None):
    pers.is_in_action = False
    pers.location = location_name
    session.add(pers)
    session.commit()
    keyboard = keyboard_main_town
    if location_name == u'Місто':
        keyboard = keyboard_main_town
    elif location_name == u'Ліс':
        keyboard = keyboard_forest
    elif location_name == u'Арена':
        keyboard = keyboard_arena
    bot.send_message(chat_id, u"Ты прибув до локації '{}'".format(location_name), reply_markup=keyboard)
```

Рисунок 3.7 – Лістинг функції «wait»

Алгоритм генерації квестів складається з загального циклу для з’єднання всіх згенерованих завдань у орієнтований граф, та з функцій для генерації різних типів завдань. На рисунку 3.8 та 3.9 зображені функції для генерації першого завдання у квесті.

```python
def getRandomTarget(lvl):
    targets = session.query(Items).filte_by(lvl)
    return targets[random.randint(0, len(targets))]


def getRandomEnemy(lvl):
    enemies = session.query(NPC).filte_by(lvl)
    return enemies[random.randint(0, len(enemies))]


def getRandomType():
    return ['kill', 'support', 'find', 'delivery'][random.randint(0, 5)]


def getRandomLocation(lvl):
    locations = session.query(Location).filter_by(lvl)
    return locations[random.randint(0, len(locations))]
```

Рисунок 3.8 – Лістинг функцій для генерації завдань ч.1.

```python
def generateQuestByLocation(lvl, location, npc_quest):
    quest = Quest(id=shortuuid.uuid())
    quest.type = getRandomType()
    quest.location = location
    quest.npc = npc_quest
    if quest.type in ['find', 'delivery']:
        quest.target = getRandomTarget(lvl)
    elif quest.type in ['support', 'delivery']:
        quest.location_to = getRandomLocation(lvl)
    elif quest.type == 'kill':
        quest.enemy = getRandomEnemy(lvl)
    session.add(quest)
    session.commit()
    return quest


def generateHeader(player, tempNode):
    location = getRandomLocation(player.lvl)
    npc_list = session.query(NPC).filter_by(location)
    npc_quest = random.randint(0, len(npc_list))
    return generateQuestByLocation(player.lvl, location, npc_quest)
```

Рисунок 3.9 – Лістинг функцій для генерації завдань ч.2.

Функції «getRandomTarget», «getRandomEnemy», «getRandomType» та «getRandomLocation» загальні і використовуються при генерації майже всіх типів завдань. Функція «generateHeader» відповідає за генерацію першого завдання в квесті. Вона збирає дані про гравця, вибирає випадкового персонажа з локацій які доступні для гравця та передає ці дані у функцію «generateQuestByLocation». Далі в функції створеться об’єкт завдання, якому присвоюється випадковий тип, локація та персонаж який має видати це завдання гравцю. Якщо тип завдання – знайти предмет або доставити його, то завданню призначається випадковий предмет з тих який теоретичино може знайти гравець в доступних для нього локаціях. Якщо тип завдання – супроводження або доставка, то випадковим чином визначається персонаж якому гравець має доставити предмет, або супроводити. Якщо тип завдання – вбиство, то випадковим чином визначається противник якого має вбити гравець. Об’єкт завдання записується у базу даних та повертається у зовнішню функцію, яка записує його у зв’язний список.

Алгоритм генерації характеристик для предмету складається з функції «generateItem». Ця функція приймає як параметри назву предмету, його тип клас та рівень, а також список коефіцієнтів розподілення характеристик та список коефіцієнтів для переводу у універсальну одиницю виміру. В функції формується список характеристик. За допомогою згенерованої кривої «рівня-потужності» отримується мінімальна та максимальна границя потужності в залежності від рівня предмету. За формулами описаними раніше у розділі 2 вираховуються максимальне та мінімальне значення кожної з характеристик та підраховується випадкове значення серед цього інтервалу. Далі створюється об’єкт моделі предмету, та вираховані характеристики переносяться зі списку в цей об’єкт. Характеристики які мають при цьому зберігати певний діапазон мінімального та максимального значення для подальшої унікалізації предметів беруть відповідно відхилення у 5 відсотків від вирахуваного значення характеристики. Після заповнення об’єкту йде запис у базу даних.

```
def generateItem(name, type_, clas_s, lvl, koef_rosp_list, koef_divide_list):
    n = 6
    characteristic = list()
    Min, Max = getMinMax(lvl)
    for kr, i in enumerate(koef_rosp_list):
        x_max = Max * kr / n
        x_min = Min * kr / n
        characteristic.append(random.randint(x_min, x_max) / koef_divide_list[i])

    item = DefaultItem(name=name)
    item.type = type_
    item.clas_s = clas_s
    item.damage_physic_min = characteristic[0] * 0.95
    item.damage_physic_max = characteristic[0] * 1.05
    item.damage_magic_min = characteristic[1] * 0.95
    item.damage_magic_max = characteristic[1] * 1.05
    item.defence_physic_min = characteristic[2] * 0.95
    item.defence_physic_max = characteristic[2] * 1.05
    item.defence_magic_min = characteristic[3] * 0.95
    item.defence_magic_max = characteristic[3] * 1.05
    item.dodge_physic = characteristic[4]
    item.dodge_magic = characteristic[5]
    item.crit_physic = characteristic[6]
    item.crit_magic = characteristic[7]
    item.rarity = characteristic[8]
    item.price = characteristic[9]
    item.lvl = lvl

    session.add(item)
    return item
```

Рисунок 3.10 – Алгоритм генерації характеристик предмету.

Алгоритми PVP та PVE майже ідентичні, окрім того, що при написані алгоритму PVE потрібно врахувати те, що з одним і тим же NPC (not played character) можуть битися різні персонажі, тому при його витягнені з БД потрібно робити клона з такими ж характерстиками. Сам алгоритм полягає в розроблені функції випадкового вибору результату атаки в залежності від доволі великої кількості різноматних факторів та правильної черговості нападу персонажів в залежності від того, на кого випав жереб для першого ходу.

Алгоритм PVE складається з двох функцій. Функція «PvE(pers, monster)»

приймає як аргументи ідентифікатор персонажа та ім’я противника. Спочатку встановлюються початкові дані для поєдинку, потім визначається черговість атаки, для чого обирається випадкове число від одного до двох (рисунок 3.11).

```
def PvE(pers, monster):
    monster = session.query(NPC).filter_by(name=monster).first()
    monster_temp_hp = monster.max_hp
    pers.temp_count_for_combo = 0
    pers_die = False
    monster_die = False
    first_fighter = random.randint(1, 2)
    result_of_fight = u"Сражение с {0} \n".format(monster.name)
    count = 0
    chance_to_physic = session.query(PersonalItem).filter_by(uuid=pers.weapon).first()
    if chance_to_physic:
        chance_to_physic = chance_to_physic.chance_to_physic
    else:
        chance_to_physic = 100
```

Рисунок 3.11 – Лістинг функції «PvE» (частина 1)

Далі запускаємо цикл, який буде виконуватись поки хтось з суперників не загине, почергово виконуючи функцію атаки для кожного з супротивників і перевіряючи їх рівень життя (рисунок 3.12).

```
while not pers_die and not monster_die:
    if first_fighter == 1:
        add_result_of_fight, monster_temp_hp = attack_mob(pers, monster, monster_temp_hp,
        chance_to_physic)
        result_of_fight += add_result_of_fight
        if monster_temp_hp <= 0:
            monster_die = True
        else:
            result_of_fight += mob_attack(monster, pers)
            if pers.hp <= 0:
                pers_die = True
    else:
        result_of_fight += mob_attack(monster, pers)
        if pers.hp <= 0:
            pers_die = True
        else:
            add_result_of_fight, monster_temp_hp = attack_mob(pers, monster, monster_temp_hp,
            chance_to_physic)
            result_of_fight += add_result_of_fight
            if monster_temp_hp <= 0:
                monster_die = True
    count += 1
```

Рисунок 3.12 – Лістинг функції «PvE» (частина 2)

Після виходу із циклу (якщо загинув персонаж) відправляється відповідне повідомлення з результатами поєдинку, а також асинхронно запускається функція «pers_death», яка переводить персонажа в стан смерті на деякий час. Якщо ж герой виграв поєдинок, виконується функція «drop_from_mob», котра визначає чи знайшов персонаж якісь речі у загиблого супротивника, після цього бот також видає повідомлення про результати поєдинку та обнуляє потрібні значення (рисунок 3.13).

```
if pers_die:
    result_of_fight += u"Поразка, у монстра залишилось {0}хп, поєдинок"+/
    "тривав {1} ходів".format(monster_temp_hp, count)
    bot.send_message(pers.chat_id, result_of_fight)
    pers_death.delay(pers.user_id)
else:
    pers_item = drop_from_mob(monster)
    if pers_item:
        pers_item.person = pers.user_id
        session.add(pers_item)
        result_of_fight += u"Гравець {0} переміг з {1}, поєдинок тривав" +/
        "{2} ходів, вы отримали {3}".format(pers.name, pers.hp, count, pers_item.__unicode__())
    else:
        result_of_fight += u"Гравець {0} переміг з {1}, поєдинок тривав" +/
        "{2} ходів".format(pers.name, pers.hp, count)
    bot.send_message(pers.chat_id, result_of_fight)

pers.temp_count_for_combo = 0
session.add(pers)
session.commit()
```

Рисунок 3.13 – Лістинг функції «PvE» (частина 3)

Функція «attack_mob:» приймає як аргументи ідентифікатори персонажа та монстра. Далі йде перевірка на те, яким типом пошкодження атакує персонаж (фізичним чи магічним). Перевіряється, чи зміг монстр ухилитися від атаки, чи набрав атакуючий потрібну кількість успішних атак для посиленної атаки, чи буде посиленна атака критичною; якщо ні, то чи буде його звичайна атака критичною. Якщо ж монстр зміг ухилитись, йде перевірка на те, чи зміг персонаж зберегти ланцюг успішних атак, чи ні. Після потрібних перевірок функція формує повідомлення про стан атаки, віднімає потрібну кількість очків здоров’я та повертає результат в головну функцію «PvE» (рисунок 3.14).

```
def attack_mob(pers, mob, mob_hp, chance_to_physic):
    with_combo = False
    with_crit = False
    if pers.weapon and random.uniform(0, 100) > chance_to_physic:
        if random.uniform(0, 100) > mob.dodge_magic - pers.accuracy_magic:
            basic_damage = random.randint(pers.damage_min_magic, pers.damage_max_magic)
            if pers.temp_count_for_combo == pers.count_for_combo:
                with_combo = True
                basic_damage *= pers.coef_for_combo
                pers.temp_count_for_combo = 0
            else:
                pers.temp_count_for_combo += 1
            if random.uniform(0, 100) <= pers.crit_magic:
                with_crit = True
                damage = (coefCrit * basic_damage) - mob.defence_magic + pers.penetration_magic
            else:
                damage = basic_damage - mob.defence_magic + pers.penetration_magic
            mob_hp -= damage
            result = u"{0} наніс ".format(pers.name)
            if with_crit:
                result += u"💥"
            if with_combo:
                result += u"🥊"
            result += u" 🔮{0}пошкодження, у {1} залишилось {2}HP\n".format(damage, mob.name, mob_hp)
        else:
            if random.uniform(0, 100) > pers.temp_attack:
                pers.temp_count_for_combo = 0
            result = u"{0} ухилився 👈".format(mob.name)
```

Рисунок 3.14 – Лістинг функції «attack_mob»

Алгоритм генерації персоналізованих речей складається з двох функцій. Функція «drop_from_mob» викликається, якщо персонаж знаходить якусь річ з певного супротивника. Вона генерує випадкове число: якщо воно в межах від 0 до шансу випадення речі, то витягуються дані про шанс випадення речей з того чи іншого монстра. Також шляхом генерації випадкового числа, вибирається конкретний шаблон певної речі (рисунок 3.15).

```
def drop_from_mob(monster):
    pers_item = None
    if random.uniform(0, 100) <= 100:
        drops = session.query(Drop).filter_by(monster=monster.name)
        range_for_random = []
        temp_range = 0
        for d in drops:
            temp_range += d.chance_for_drop
            range_for_random.append(temp_range)
        rand = random.uniform(0, 100)
        iterate_drop = 0
        for i, t_r in enumerate(range_for_random):
            if rand <= t_r:
                iterate_drop = i
                break
        for i, drop in enumerate(drops):
            if i == iterate_drop:
                pers_item = generate_personal_item(drop)
    return pers_item
```

Рисунок 3.15 – Лістинг функції «drop_from_mob»

Після вибору шаблона викликається функція «generate_personal_item», яка приймає як аргумент ідентифікатор шаблона. Далі йде розрахунок всіх характеристик шляхом випадкових чисел за правилами і рамками, визначеними в шаблоні, після чого речі присвоюється ідентифікатор персонажа (рисунок 3.16).

```
def generate_personal_item(drop):
    model = session.query(DefaultItem).filter_by(name=drop.model).first()
    pers_item = PersonalItem(name=model.name, price=model.price, type=model.type,
                             damage_physic_min=random.randint(model.damage_physic_min,
                                                              (model.damage_physic_min + model.damage_physic_max) / 2),
                             damage_physic_max=random.randint((model.damage_physic_min + model.damage_physic_max) / 2,
                                                              model.damage_physic_max),
                             damage_magic_min=random.randint(model.damage_magic_min,
                                                             (model.damage_magic_min + model.damage_magic_max) / 2),
                             damage_magic_max=random.randint((model.damage_magic_min + model.damage_magic_max) / 2,
                                                             model.damage_magic_max),
                             defence_physic=random.randint(model.defence_physic_min, model.defence_physic_max),
                             defence_magic=random.randint(model.defence_magic_min, model.defence_magic_max),
                             dodge_magic=model.dodge_magic,
                             dodge_physic=model.dodge_physic,
                             crit_magic=model.crit_magic,
                             crit_physic=model.crit_physic,
                             chance_to_physic=model.chance_to_physic,
                             rarity=model.rarity,
                             uuid=get_default_as_uuid())
    return pers_item
```

Рисунок 3.16 – Лістинг функції «generate_personal_item»

Отже, у підрозділі 3.3 було описано розроблений програмний код основних складових програмного додатку. Повний лістинг наведено у додатку Б.

3.4 Висновки

В даному розділі було проведено аналіз мов програмування C++, C#, Java та Python, середовищ розробки PyCharm, Spyder та Visual Studio, а також СКБД MongoDB та MySQL. В результаті аналізу було прийнято рішення використовувати середовище розробки PyCharm, мову Python, СКБД MySQL та інструментарій SQLAlchemy для розробки програмного засобу "FireBox". Також було описано основні розроблені програмні модулі ігрового боту.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення — процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином [33].

Методи тестування включають процес пошуку помилок або інших дефектів і тестування програмних компонентів для їх оцінки.

Може оцінюватись:

- відповідність вимогам дизайнерів та розробників;
- правильна відповідь на всі можливі вхідні дані;
- виконання функцій у розумні терміни;
- практичність;
- сумісність із програмним забезпеченням та операційними системами;
- відповідність завданням замовника.

Сьогодні тестування ПЗ є одним із найдорожчих етапів життєвого циклу ПЗ [34], на нього припадає від 50% до 65% загальних витрат.

Існує кілька основних методів тестування, які є частиною режиму тестування програмного забезпечення. Ці тести, як правило, вважаються самостійними у пошуку помилок у всій системі.

Тестування методом «чорної скриньки» здійснюється без будь-яких знань внутрішньої роботи системи. Тестер симулюватиме дії користувача програмного середовища, надаючи різні входи і тестуючи згенеровані виходи. Цей тест також відомий як Black-box, closed-box тестування або функціональне тестування [35].

Основні особливості тестування методом «чорної скриньки»:

– Тестування, як функціональне, так і нефункціональне, що не передбачає знання внутрішнього пристрою компонента або системи.

– Тест-дизайн, заснований на методі «чорної скриньки» – процедура написання або вибору тест-кейсів на основі аналізу функціональної або нефункціональної специфікації компонента або системи без знання її внутрішнього пристрою.

Мета цього методу це пошук помилок у таких категоріях:

– неправильно реалізовані або відсутні функції;
– помилки інтерфейсу;
– помилки у структурах даних чи організації доступу до зовнішніх баз даних;
– помилки поведінки або недостатня продуктивність системи.

Переваги:

– тестування проводиться з позиції кінцевого користувача і може допомогти виявити неточності та протиріччя у специфікації;
– тестувальнику не потрібно знати мови програмування та заглиблюватися особливо реалізації програми;
– тестування може проводитись фахівцями, незалежними від відділу розробки, що допомагає уникнути упередженого ставлення;
– можна починати писати тест-кейси, як тільки готова специфікація.

Недоліки:

– тестується лише дуже обмежена кількість шляхів виконання програми;
– без чіткої специфікації досить складно скласти ефективні тест-кейси;
– деякі тести можуть бути надмірними, якщо вони вже були проведені розробником на рівні модульного тестування.

Таким чином, ми не маємо уявлення про структуру та внутрішній устрій системи. Потрібно зосереджуватися на тому, що програма робить, а не на тому, як вона це робить.

Протилежністю техніки «чорної скриньки» є тестування методом «білої скриньки». Тестування методом «білої скриньки» враховує внутрішнє функціонування і логіку роботи коду. Для виконання цього тесту, тестувальник

повинен мати досвід програміста, щоб дізнатися точну частину коду, в якій виникають ті чи інші помилки.

Основні особливості тестування методом «білої скриньки»:

- тестування, засноване на аналізі внутрішньої структури компонента або системи;
- тест-дизайн, що базується на методі «білої скриньки» – процедура написання або вибору тест-кейсів на основі аналізу внутрішнього пристрою системи або компонента.

Переваги:

- тестування може проводитися на ранніх етапах: немає необхідності чекати створення інтерфейсу користувача;
- можна провести ретельніше тестування, з покриттям великої кількості шляхів виконання програми.

Недоліки:

- для виконання тестування білої скриньки потрібна велика кількість спеціальних знань;
- при використанні автоматизації тестування на цьому рівні, підтримка тестових скриптів може бути досить накладною, якщо програма часто змінюється.

Враховуючи особливості кожного з розглянутих методів, для тестування програмного додатку було вирішено використати метод «чорної скриньки», оскільки цей метод дозволяє виявити помилки інтерфейсу, ініціалізації та завершення, некоректностні певних програмних модулів, що неможливо при тестуванні білої скриньки.

4.2 Тестування розробленого програмного продукту

Тестування розробленого програмного продукту проводиться за методикою «чорної скриньки». Вона базується на використанні шаблонів тестування, які називаються тест-кейсами.

Тестовий випадок (англ. Test Case) – це артефакт, що описує сукупність кроків, конкретних умов та параметрів, необхідних для перевірки реалізації функції, що тестується чи її частини.

Під тест кейсом мається на увазі наступна структура: дія, очікуванний результат, отриманий результат.

Тест кейси поділяються за очікуваним результатом на позитивні та негативні.

Позитивний тест кейс використовує лише коректні дані і перевіряє, що додаток правильно виконує функцію що викликається.

Негативний тест кейс оперує як коректними, так і не коректними даними (мінімум 1 некоректний приклад) і ставить за мету перевірку виняткових ситуацій (спрацювання валідаторів), а також перевіряє, що функція, яка викликається додатком, не виконується при спрацюванні валідатору.

Буде створено декілька тест-кейсів для перевірки правильності роботи основних функцій програмного додатку. Як приклад, деякі із них:

Тест-кейс №1 – Реєстрація в боті.

1. Відкрити бота.
2. Натиснути кнопку «Розпочати».
3. Ввести команду реєстрації «/name (name)».
4. Подивитись на відповідь бота.

Очікуваний результат – на екрані відображається повідомлення «Ваш профіль створенно».

Враховуючи різноманітність «нік-неймів» користувачів, залежність від нього у розробленому ігровому буті була вилучена. Ідентифікація ігрового профілю була зосереджена на власному ідентифікаторі профілю користувача месенджеру «Telegram». Саме тому обмеженя на символи в «нік-неймі» не накладаються. Система реєструє користувача, та сповіщає, що його персонаж знаходиться в локації «Місто».

Результат виконання тест-кейсу №1 зображено на рисунку 4.1.

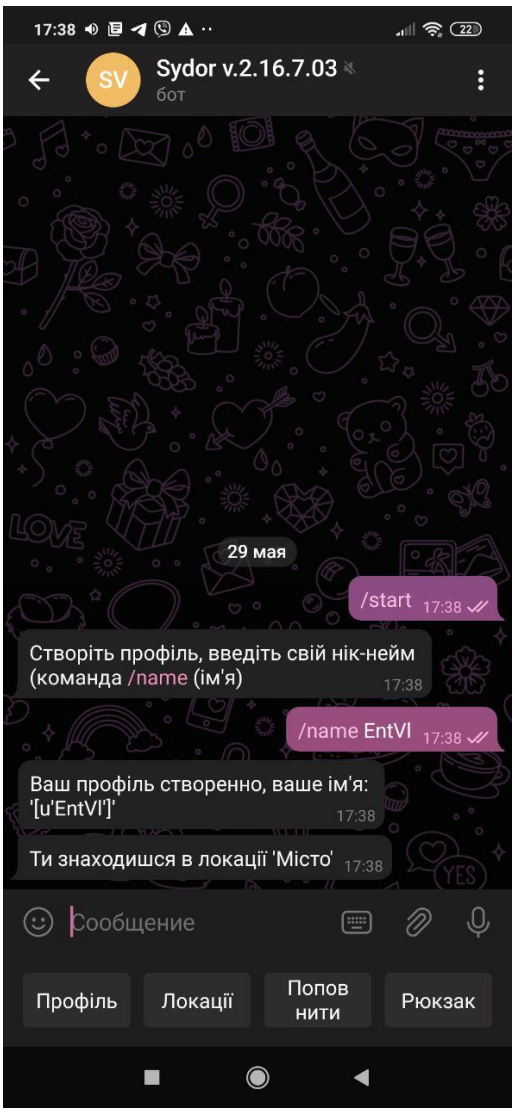


Рисунок 4.1 – Результат виконання тест-кейсу №1

Отриманий результат відповідає очікуваному, отже тест-кейс №1 успішно пройдено.

Тест-кейс №2 – Переміщення між локаціями.

1. Натиснути кнопку «Локації».
2. Натиснути кнопку «Ліс».
3. Дочекатись відповідь бота.

Очікуваний результат – через 15 секунд після натисненням користувачем кнопки про переміщення персонажу до локаціїї «Ліс» прийде повідомлення про прибуття у відповідну локацію.

Результат виконання тест-кейсу №2 показано на рисунку 4.2.

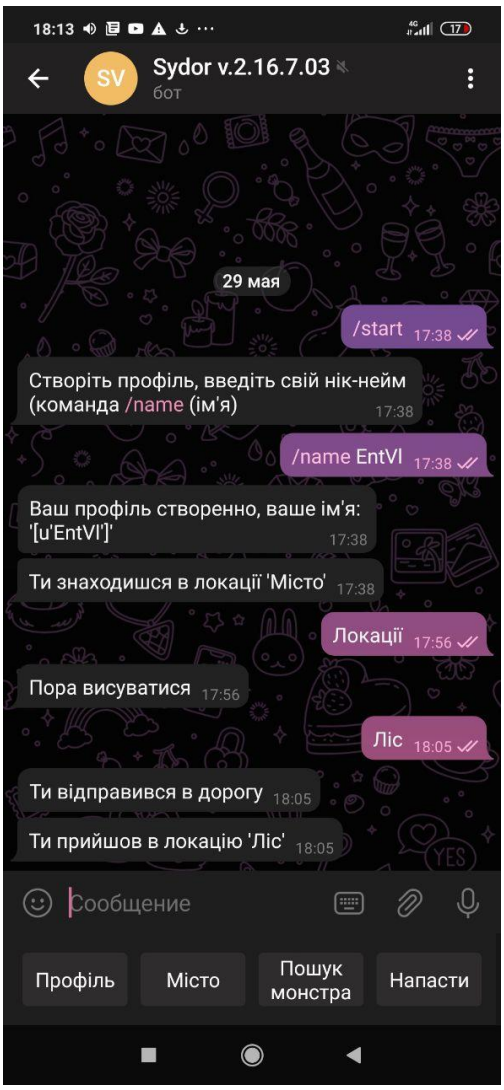


Рисунок 4.2 – Результат виконання тест-кейсу №2

Отриманий результат відповідає очікуваному, отже тест-кейс №2 успішно пройдено.

Тест-кейс №3 – Пошук супротивника та бій з ним.

1. Натиснути кнопку «Пошук монстра».
2. Дочекатись відповіді про знайденого супротивника.
3. Натиснути кнопку «Напасти».
4. Подивитись на відповідь бота.

Очікуваний результат – Відбудеться бій з монстром.

У кожній локації є різний шанс зустріти різних супротивників, проти кожного супротивника потрібна своя тактика, також з кожного з них можуть випасти свої речі. Результат тест-кейсу №3 зображено на рисунку 4.3.

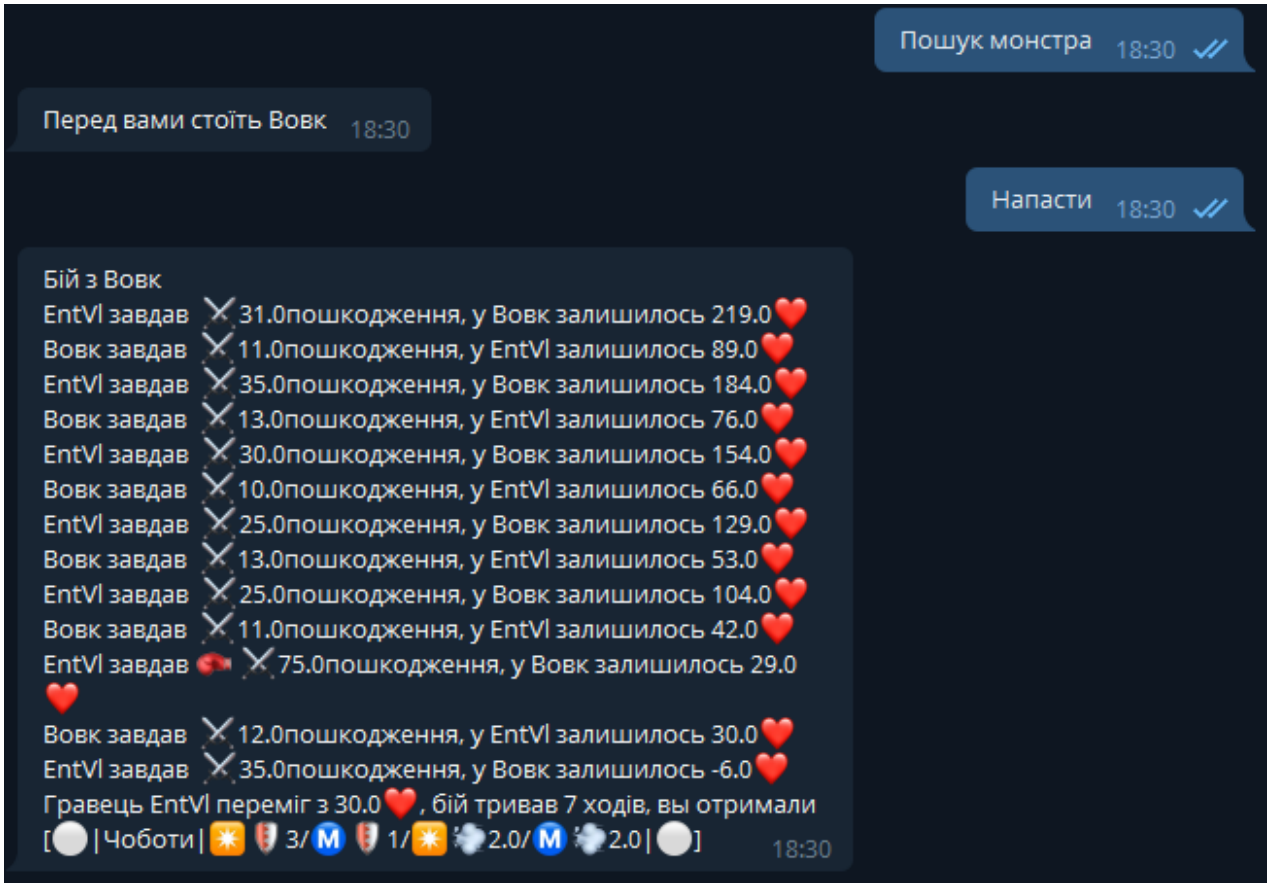


Рисунок 4.3 – Результат виконання тест-кейсу №3

Отриманий результат відповідає очікуваному, отже тест-кейс №3 успішно пройдено.

Тест-кейс №4 – Вдягання речі та випробування її в поєдинку.

1. Ввести команду «/backpack».
2. Натиснуту команду «/wear» біля потрібної речі.
3. Перевірити стан інвентаря.

Очікуваний результат – інвентарь працює справно, предмет з відповідним ідентифікатором вдягнувся на потрібне місте, та був вилучений з рюкзака, відображено всі характеристики вдягеного предмету.

Результат виконання тест-кейсу №4 зображено на рисунку 4.4.

Отриманий результат відповідає очікуваному, отже тест-кейс №4 успішно пройдено.

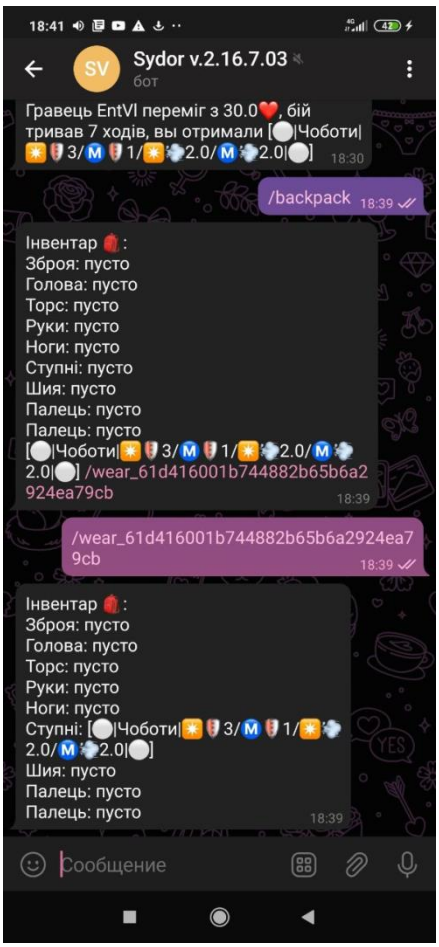


Рисунок 4.4 – Результат виконання тест-кейсу №4

Тест-кейс №5 – Перевірка механіки смерті.

1. Напасти на монстра маючи низький рівень життя.
2. Померти в поєдинку.
3. Дочекатись воскресіння.
4. Поповнити здоров’я.

Вдягнувши річ, видно, що той самий супротивник, почав наносити меньше пошкодження, що відбулось через підвищення характеристики «фізичний опір» через вдягнення на персонажа предмету «Чоботи». Після поразки, персонаж вмирає та через хвилину воскресає в локації «Місто».

Результат виконання тест-кейсу №5 зображено на рисунку 4.5.

Отриманий результат відповідає очікуваному, отже тест-кейс №5 успішно пройдено.

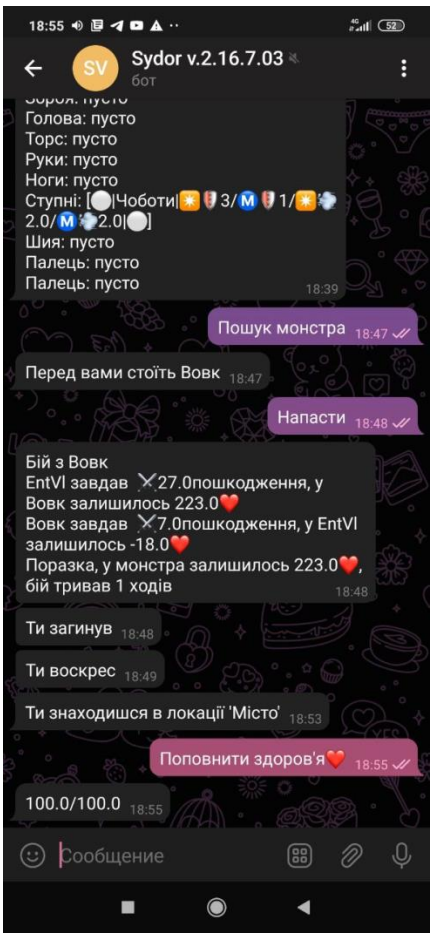


Рисунок 4.5 – Результат виконання тест-кейсу №5

Окрім роботи самого боту потрібно перевірити генерацію характеристик предметів та генерацію квестів.

Для роботи генерації характеристик предметів потрібно зіставити таблицю коефіцієнтів для розподілу характеристик та коефіцієнтів перетворення значення у універсальну одиницю. Як результат було зіставлено 6 різних таблиць для кожного з типів предметів та типів персонажей:

1. Персонаж із ухилом на захист фізичного типу;
2. Персонаж із ухилом на захист магічного типу;
3. Персонаж із ухилом на захист змішаного типу;
4. Персонаж із ухилом на пошкодження фізичного типу;
5. Персонаж із ухилом на пошкодження магічного типу;
6. Персонаж із ухилом на пошкодження змішаного типу.

Приклади згенерованих таблиць розподілу зображено на рисунках 4.6 та 4.7.

	A	B	C	D	E	F	G	H	I
1	ФТ	Ор	Шл	Дос	Шт	Об	Пер	Коеф	Сума
2	АФ	9	0	0	0	0	1	2	10
3	АМ	0	0	0	0	0	0	2	0
4	СФ	0	4	7	7	3	3	3	24
5	СМ	0	4	7	7	3	3	3	24
6	Т	2	0	0	0	0	1	1	3
7	Л	0	0	0	0	0	2	1	2
8	ФПР	4	0	0	0	0	1	2	5
9	МПР	0	0	0	0	0	0	2	0
10	К	3	0	0	0	0	2	1	5
11	ХП	0	5	8	8	3	3	10	27
12		18	13	22	22	9	16		100
13									

Рисунок 4.6 – Таблиця розподілу для персонажа із ухилом на захист фізичного типу

	A	B	C	D	E	F	G	H	I
1	МД	Ор	Шл	Дос	Шт	Об	Пер	Коеф	Сумма
2	АФ	0	0	0	0	0	0	2	0
3	АМ	20	0	0	0	0	10	2	30
4	СФ	0	1	2	2	0	0	3	5
5	СМ	0	1	2	2	0	0	3	5
6	Т	4	1	1	1	1	2	1	10
7	Л	1	1	1	1	5	1	1	10
8	ФПР	0	0	0	0	0	0	2	0
9	МПР	13	0	0	0	0	7	2	20
10	К	5	0	0	0	0	5	1	10
11	ХП	0	2	3	3	2	0	10	10
12		43	6	9	9	8	25		100
13									

Рисунок 4.7 – Таблиця розподілу для персонажа із ухилом на пошкодження магічного типу

Після генерації таблиць за допомогою скрипта переносимо дані до бази даних. В іншу таблицю вносимо початкові дані про предмети для майбутньої генерації та запускаємо скрипт для генерації характеристик предметів.

Отже тест-кейс №6 виглядить наступним чином:

1. Ввести початкові дані про предмети;
2. Запустити генерації решти характеристик;
3. Звірити отримані характеристики з розробленими формулами.

Введенні дані про предмети виглядять наступним чином (рис. 4.8 – 4.9).

	A	B	C	D	E	F	G	H	I
1	Название	Тип	Класс	Мин.физ.урон	Макс.физ.урон	Мин.маг.урон	Макс.маг.урон	Мин ф.б.	Макс ф.б.
2	Деревяный посох	weapon	md						
3	Рваная туника	body	pt						

Рисунок 4.8 – Введенні дані про предмети для генерації характеристик ч.1

J	K	L	M	N	O	P	Q	R	S	T
Мин м.б.	Макс м.б.	Точность	Ловкость	ФПР	МПР	Крит	Хп	Редкость	Цена	Лвл
								Обычный		1
								Обычный		1

Рисунок 4.9 – Введенні дані про предмети для генерації характеристик ч.2

Після запуску генерації таблиця набула наступного вигляду (рис 4.10 – 4.11):

	A	B	C	D	E	F	G	H	I
1	Название	Тип	Класс	Мин.физ.урон	Макс.физ.урон	Мин.маг.урон	Макс.маг.урон	Мин ф.б.	Макс ф.б.
2	Деревяный посох	weapon	md	0	0	4	5	0	0
3	Рваная туника	body	pt	0	0	0	0	1	2

Рисунок 4.10 – Згенеровані характеристики предметів ч.1

J	K	L	M	N	O	P	Q	R	S	T
Мин м.б.	Макс м.б.	Точность	Ловкость	ФПР	МПР	Крит	Хп	Редкость	Цена	Лвл
0	0	1	0	0	4	2	0	Обычный	40	1
1	1	0	0	0	0	0	22	Обычный	20	1

Рисунок 4.11 – Згенеровані характеристики предметів ч.2

Підставивши згенеровані характеристики предметів у розроблені формули можемо відмітити що отриманий результат відповідає очікуваному і алгоритм генерації характеристик працює.

Тест-кейс №7:

1. Відправити команду для тестової генерації квесту;
2. Отримати у відповідь квест який потрібно виконати;
3. Виконати квест;

4. Отримати сповіщення про виконання квесту;
5. Забрати винагороду;
6. Отримати винагороду.

Для початку було створено команду для тестової генерації квесту. Після відправлення було отримано квест з умовами зображеними на рисунку 4.12.

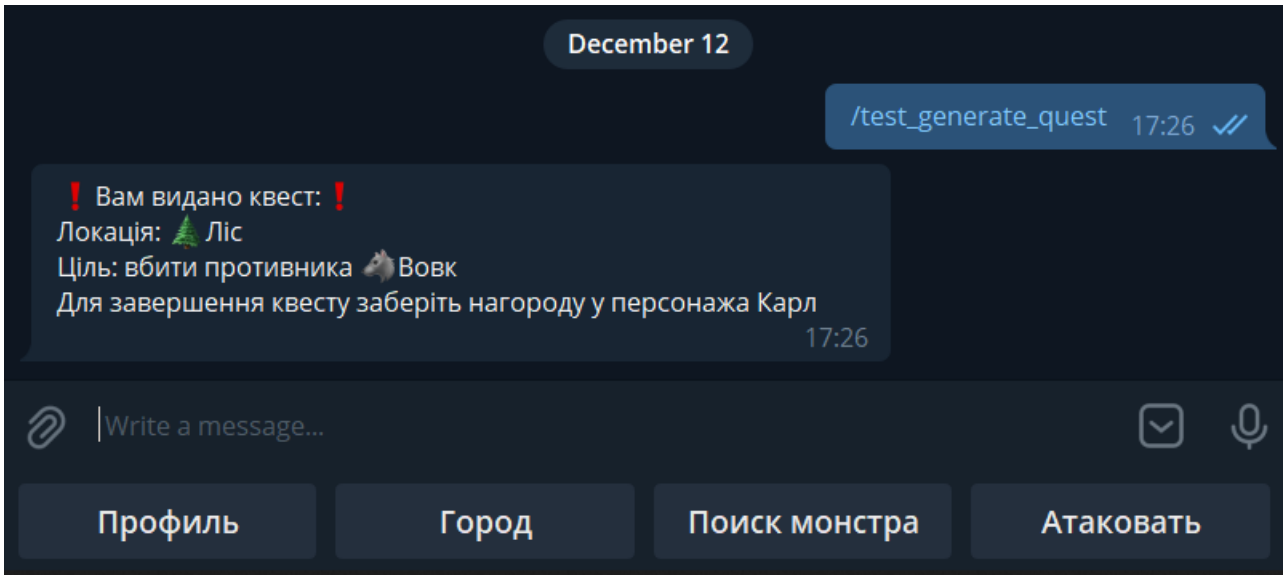


Рисунок 4.12 – Результат генерації квесту

Після отримання квесту були виконані зазначені у опису умови, та як очікувалось повідомлення про те, що квест був виконаний (рис. 4.13).

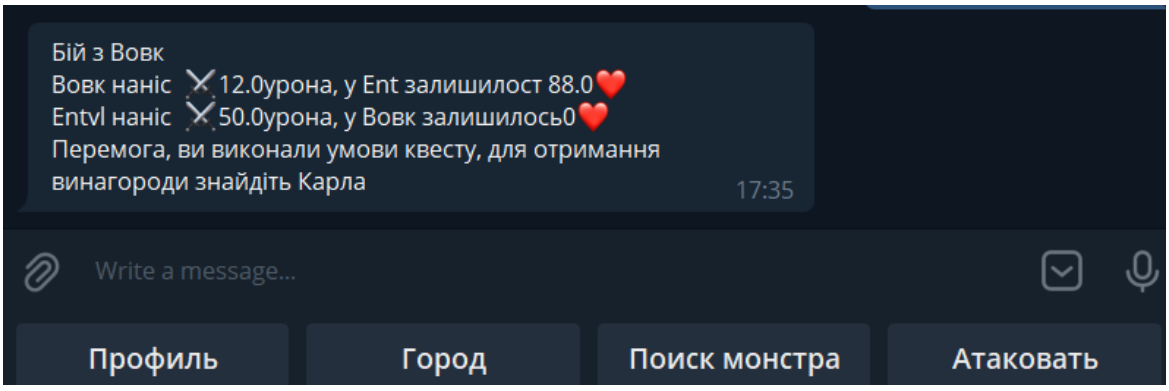


Рисунок 4.13 – повідомлення про виконання умов квесту

Після отримання сповіщення персонаж гравця був направлений до вказоного в умовах квесту персонажа та отримав винагороду за квест (рис. 4.14).

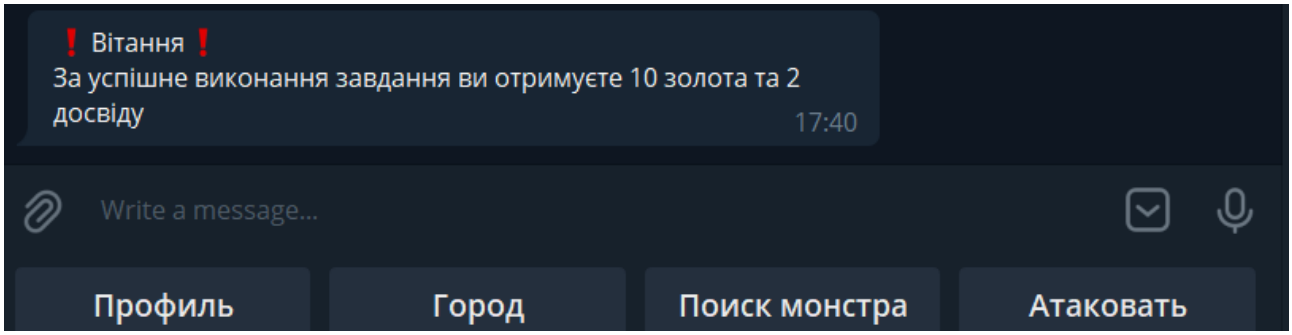


Рисунок 4.14 – Повідомлення про отримання винагороди за квест

Було розроблено та виконано ще ряд різноманітних тест-кейсів, які перевіряють роботу усіх модулів Telegram-бота. Усі вони були успішно пройдені, у зв'язку з чим можна зробити висновок, що бот функціонує нормально і є придатним для використання.

4.3 Розробка інструкції користувача

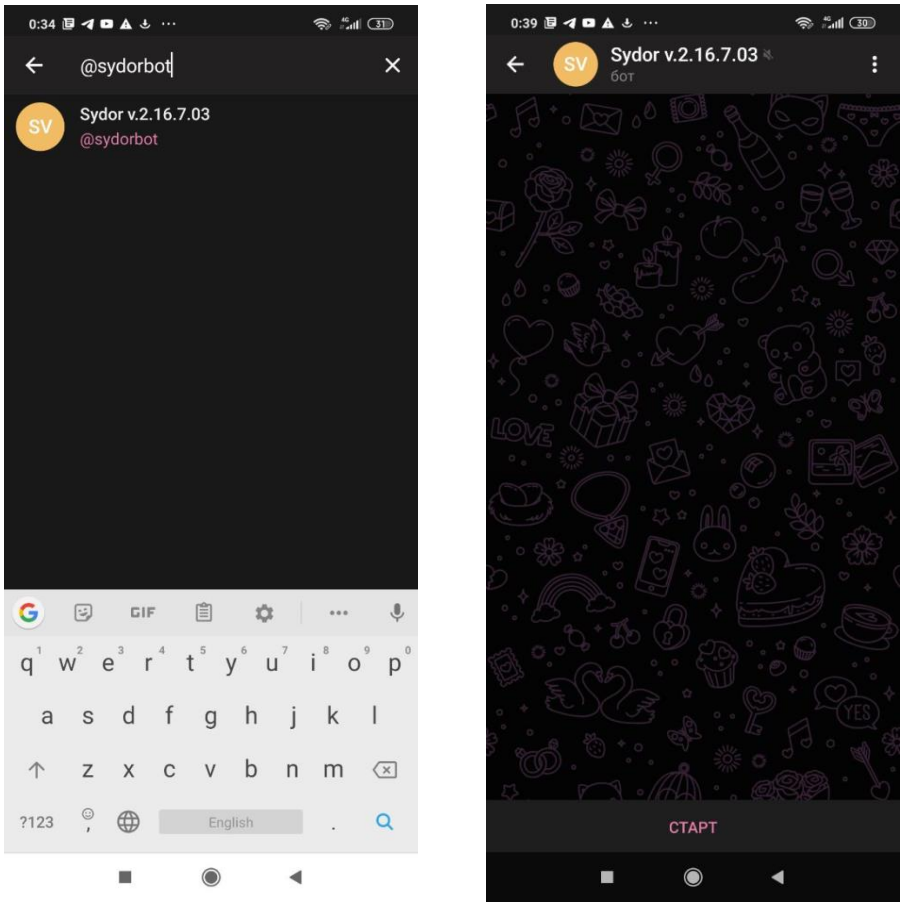
«FireBox» – ігровий бот створенний на базі мессенджеру «Telegram».

Для запуску боту потрібно встановити будь-який клієнт месенджеру та заруєструватись, або авторизуватись. Після входу на свій аккаунт потрібно в пошуку ввести «@sydorbot», що одночасно може виступати посиланням в смс-повідомленні (рисунок 4.15.а).

Клієнтів месенджеру «Telegram» на разі існує дуже багато вони доступні на всі популярні платформи:

1. Windows;
2. Linux;
3. Mac OS;
4. Android;
5. IOS;
6. Web-Telegram.

Далі потрібно зайти на аккаунт бота та натиснути команду старт, після якої бот привітається з користувачем та дасть інструкцію по реєстрації в грі (рис. 4.15.б).



а) б)

Рисунок 4.15 – Робота з ботом

Для реєстрації потрібно ввести команду «/name», після якої через пробіл вказати будь-який нік-нейм. Після регістрації у користувача є свобода дії, обмежена лише наданими ботом командами. Ігровий процес у кожного індивідуальний, а навчати та допомогати новачкам будуть «ветерани» гри.

Окрім доступних команд на клавіатурі бот може розпізнавати команди швидкого доступу, такі як: «/backpack» та «/hero». Перша команда дозволяє відкрити та продивитись інвентарь в будь-який момент, а друга дозволяє побачити свій профіль (рисунок 4.16).

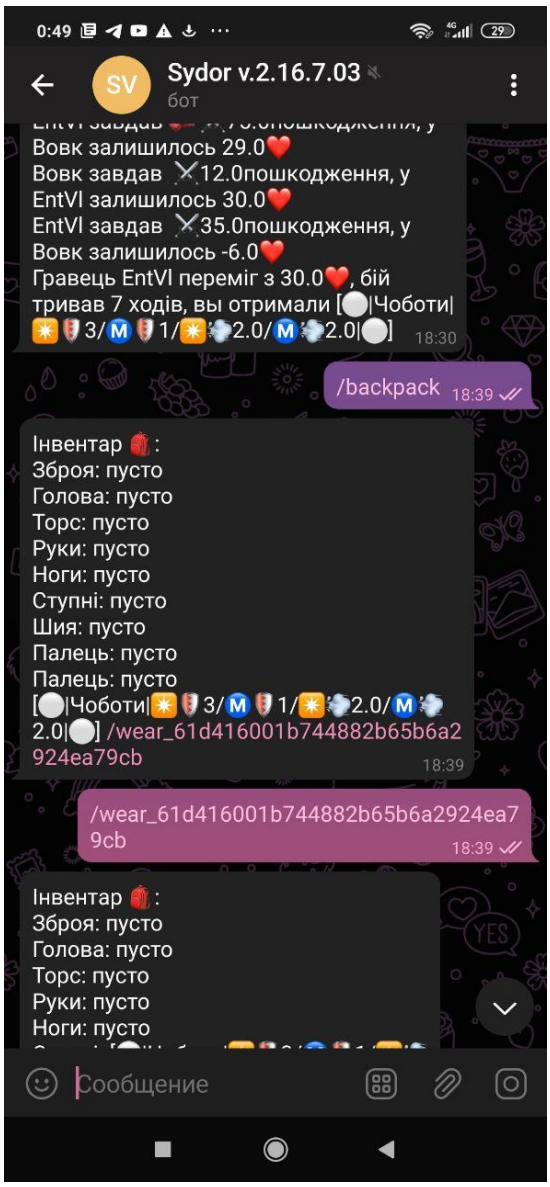


Рисунок 4.16 – Приклад використання швидких команд

У верхній правій частині бота при натиснені кнопки у вигляді 3-х крапок викликається випадне меню для керування ботом, в ньому є наступні функції:

1. Пошук по повідомленням.
2. Відправка телефону.
3. Очищення історії листування.
4. Включення/відключення сповіщень.
5. Видалення чату.

Приклад виклику меню зображено на рисунку 4.17.

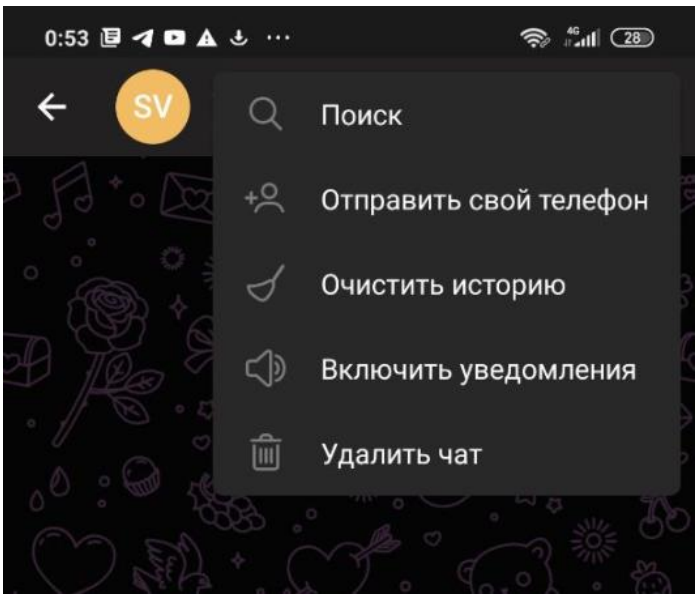


Рисунок 4.17 – Випадне меню для керування ботом

Натисненням на назву боту можливо відкрити вікно статусу, яке містить коротку інформацію про бота, кнопку для копіювання посилання на нього та перемикач для включення та відключення сповіщень (рисунок 4.18).

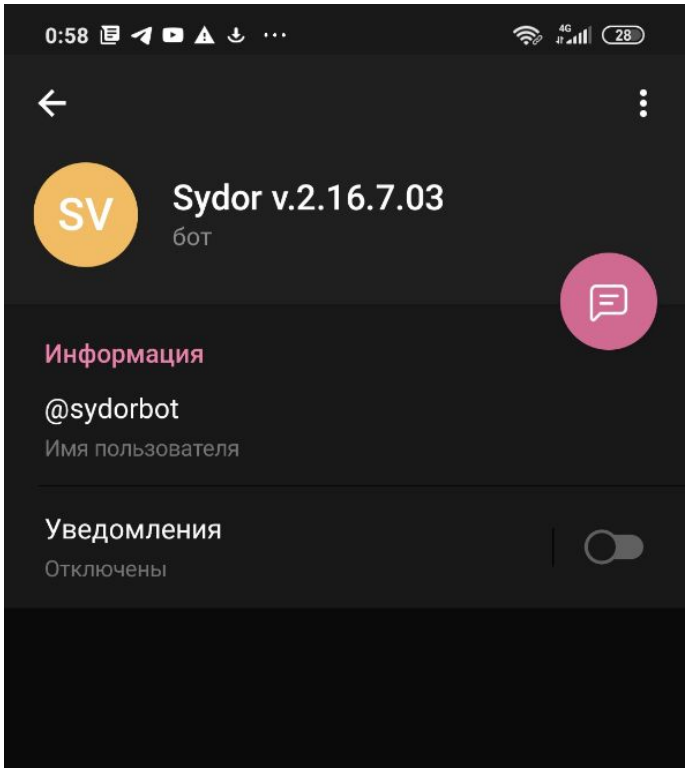


Рисунок 4.18 – Вікно статусу бота

У вікна статусу також є своє меню, яке містить наступні команди:

1. Додати в спільний чат;
2. Поділитись посиланням на бота;
3. Зупинити роботу бота;
4. Створити ярлик на робочому столі.

Виклик меню продемонстрований на рисунку 4.19.

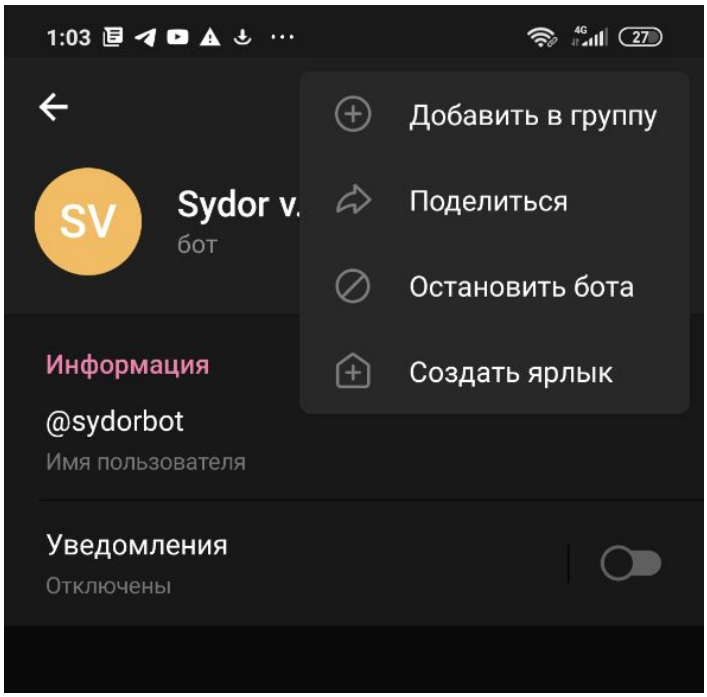


Рисунок 4.19– Меню вікна статусу бота

Отже, було розроблено інструкцію користувача. В ній описано дії для початку роботи з ботом, та можливі дії користувача для досягнення того чи іншого результату.

4.4 Вимоги до персонального комп’ютера

Вимоги до складу і параметрів технічних засобів. Для запуску бота користувачу лише потрібно встановити мессенджер Telegram. Існують версії Telegram як і на телефони з ОС Android та IOS, так і версії на персональний комп’ютер і Web-версія [36]. Мінімальні та рекомендовані конфігурації телефону на базі ОС Android наведені в таблицях 4.1. та 4.2.

Таблиця 4.1 – Мінімальна конфігурація

Характеристика	Значення
Android	4.0 Ice Cream Sandwich
Об’єм оперативної пам’яті	512 МБ
Об’єм вбудованної пам’яті	2 ГБ

Таблиця 4.2 – Рекомендована конфігурація

Характеристика	Значення
Android	6.0 Ice Cream Sandwich
Об’єм оперативної пам’яті	1 ГБ
Об’єм вбудованної пам’яті	4 ГБ

Для того, щоб запустити бота після інсталяції мессенджеру Telegram, потрібно перейти по його посиланю «@sydorbot» та натиснути на запропоновану команду «/start».

4.5 Висновки

Тестування ігрового Telegram-боту проводилось за методикою «чорної скриньки». Результат тестування показав працездатність програмного продукту та відповідність поставленому технічному завданню. Розроблено інструкцію користувача, що допоможе користувачу полегшити користування програмним продуктом. Визначено мінімальну та рекомендовану конфігурації персональних комп’ютерів та телефонів, необхідних для коректної роботи програмного засобу.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Магістерська кваліфікаційна робота з розробки методів і програмних засобів для підвищення ефективності ігрових механізнів на основі Telegram-ботів відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок, тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію проекту і переконати його в економічній доцільності такого кроку.

Для досягення мети мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором

Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть к.е.н., доц. каф. ЕПВМ Нікіфорова Л.О. к.т.н., доц. каф. ПЗ Романюк О.В. та к.т.н. доц. кафедри ПЗ Войтко В. В.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за п’ятибальною шкалою)					
	0	1	2	3	4
1	*2*	*3*	*4*	*5*	*6*
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено работоздатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижча за ціни аналогів	Ціна продукту значно нижча за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці 5.1

1	*2*	*3*	*4*	*5*	*6*
Ринкові перспективи					
6	Ринок малий і не має позитив-ної динаміки	Ринок малий, але має позити-вну динаміку	Середній ринок з позитивн ою ди-намікою	Великий стабі-льний ринок	Великий ринокз позитивною динамікою
7	Активна конку-ренція великих компаній на ринку	Активна конку-ренція	Помірн а конку-ренція	Незначна кон-куренція	Конкурен тівнемає
Практична здійсненність					
8	Відсутні фахів-ці як з техніч-ної, так і з комерційно ї реалізації ідеї	Необхідно най-мати фахівців або витрачати значні кошти тачас на навчання наявних фахів- ців	Необхідн е незна-чне навчання фахівців та збі-льшення їх штату	Необхідне не- значне навчан-ня фахівців	Є фахівці з пи-тань як з техні-чної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресу-рси, які відсут-ні. Джерела фінансуван ня ідеї відсутні	Потрібні незна-чні фінансові ресурси. Дже-рела фінансу-вання відсутні	Потрібні значні фінансов і ресур-си. Джерела фі-нансуван ня є	Потрібні незна-чні фінансові ресурси. Дже- рела фінансу-вання є	Не потребує додатковог о фінансуван ня
10	Необхідна роз-робка нових матеріалів	Потрібні мате- ріали, що вико-ристовують ся увійськово про-мисловому комплексі	Потрібн і дорогі матеріа ли	Потрібні дося-жні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно викорис-товуються у виробництві

Продовження таблиці 5.1

1	*2*	*3*	*4*	*5*	*6*
11	Термін реаліза- ції ідеї більший за 10 років	Термін реаліза-ції ідеї більший за 5 років. Тер-мін окупності інвестицій бі- льше 10-ти ро- ків	Термін реалізації ідеї від 3-х до 5- ти років. Термін окупності інвес- тицій більший 5-ти років	Термін реаліза-ції ідеї менший3-х років. Тер-мін окупності інвестицій від 3-х до 5-ти ро- ків	Термін реаліза-ції ідеї менший3-х років. Тер-мін окупності інвестицій менший 3-х років
12	Необхідна роз- робка регламе-нтних докумен-тів та отриман-ня великої кіль-кості дозвіль-них документів на виробництв ота реалізацію продукту	Необхідно отримання ве- ликої кількості дозвільних до- кументів на виробництво тареалізацію про-дукту, що пот-ребує значних коштів та часу	Процедура отри- мання дозвільних документів для виробництв а та реалізації проду- кту потребує не- значних коштів та часу	Необхідно тіль-ки повідомлен -ня відповідни м органам про виробництв о та реалізацію про-дукту	Відсутні будь- які регламентн іобмеження на виробництв о та реалізацію про-дукту

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.2.

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1. Експерт 1	2. Експерт 2	3. Експерт 3
	Бали, виставлені експертами:		
1	2	3	4

Продовження таблиці 5.2

1	2	3	4
Технічна здійсненність концепції	4	4	3
Ринкові перспективи (наявність аналогів)	1	1	2
Ринкові переваги (ціна продукту)	3	4	3
Ринкові перспективи (технічні можливості)	3	4	4
Ринкові перспективи (експлуатаційні витрати)	3	2	3
Ринкові перспективи (розмір ринку та динамика)	2	2	1
Ринкові перспективи (конкуренції)	3	3	3
Практична здійсненість (наявність фахівців)	3	4	3
Практична здійсненість (кількість ресурсів)	1	2	3
Практична здійсненість (матеріали)	3	3	2
Практична здійсненість (термін реалізації)	4	4	4
Практична здійсненість (регламентація)	4	4	5
Сума балів	$СБ_1 = 34$	$СБ_2 = 37$	$СБ_3 = 38$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = 36{,}3$		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Отже, з отриманих даних таблиць видно, що нова розробка має рівень комерційного потенціалу вище середнього.

5.2 Оцінювання рівня новизни розробки

Виводячи на ринок новинку виробник вважає, що тієї новизни, якою наділена нова розробка є достатньо для того, щоб вона була сприйнята споживачем як нова. Але це не завжди так, в силу того, що споживач і виробник неоднозначно визначать її рівень новизни. Тому доцільним є визначення рівня новизни розробки отриманої в результаті досліджень за темою «Розробка методу та програмних засобів оцінки основних параметрів роботи кол-центрів».

Саме визначення рівня і ступеня інтегральної новизни є найбільш актуальним, оскільки її рівень визначає ступінь однакового позитивного сприйняття новизни розробки як виробником, так і споживачем, а отже і ринком в цілому, а це, у свою чергу, є гарантією того, що новинка знайде своє місце на ринку, користуватиметься попитом у споживачів і забезпечить відшкодування витрат, зазнаних товаровиробником під час розроблення та виробництва технічної розробки **[Кавецький практикум 2016]**.

Рівень новизни нової продукції розраховуємо експертним методом шляхом протиставлення нової продукції та її аналогів, що існують в даний час на ринку, за чинниками що визначають її значення, в системі «краще-гірше». Рівень новизни встановлюємо відносно рівня аналога (або продукту, що досить близький до аналога).

Для визначення *і*-го виду новизни, застосуємо чинники, які впливають на її рівень. Кожен чинник *і*-го виду новизни розраховуємо в балах. Більша кількість набраних балів свідчить про більший рівень новизни. Для оцінювання рівня новизни використаємо думки експертів, які встановлюють визначені бали відповідним чинникам. Бал відповідності проставляється в діапазоні від (-5 – значно гірше аналога до +5 – значно краще аналога). Результати попереднього оцінювання зведемо до відповідного листа оцінювання (таблиця 5.4).

Таблиця 5.4 – Лист оцінювання рівня новизни експертами

Види та чинники		Бали та експерти		
		Експерт 1	Експерт 2	Експерт 3
1		*2*	*3*	*4*
Споживча новизна	Питома вага 0,225	Максимальний бал $Б_{i\ MAX}$		25
1. Зміна поведінкових звичок споживача		2	2	3
2. Ступінь задоволення потреб і запитів		2	1	2
3. Спосіб задоволення потреби		2	3	3
4. Формування нової потреби		1	2	1
5. Формування нового споживача		2	1	2
Середній бал експертів $Б_{i\ отр}$		9.6		
Товарна новизна	Питома вага 0,217	Максимальний бал $Б_{i\ MAX}$		30
1. Параметричні зміни показників продукції				
1.1. Якісні		3	4	4
1.2. Технічні		4	4	3
1.3. Економічні		2	3	3
1.4. Сервісні		3	3	4

Продовження таблиці 5.4

1		*2*	*3*	*4*
2. Якість продукції по відношенню до конкурентів		5	4	4
3. Функціональні зміни		3	4	4
Середній бал експертів $Б_{i\ отр}$		21.3		
Виробнича новизна	Питома вага 0,042	Максимальний бал $Б_{i\ MAX}$		25
1. Рівень унікальності товару для підприємства		3	4	3
2. Рівень унікальності для галузі		3	3	4
3. Рівень унікальності товару для країни		3	3	2
4. Зміна виробничої системи		1	2	2
5. Відносно існуючого асортименту		3	2	3
Середній бал експертів $Б_{i\ отр}$		8.2		
Прогресивна новизна	Питома вага 0,179	Максимальний бал $Б_{i\ MAX}$		25
1. Зміна технології виготовлення		3	4	3
2. Рівень застосування нових компонентів і матеріалів		1	0	1
3. Зміна технологічного принципу дії виробу		3	2	3
4. Зміна конструктивного виконання		3	2	3
5. Рівень застосування інновацій		2	2	3
Середній бал експертів $Б_{i\ отр}$		11.6		
Ринкова новизна	Питома вага 0,12	Максимальний бал $Б_{i\ MAX}$		20
1. Новий виріб на новому ринку		3	4	3
2. Новий виріб на відомому ринку		1	1	1
3. Модернізований виріб		3	3	3

Продовження таблиці 5.4

1		*2*	*3*	*4*
4. Нова модель		3	2	2
Середній бал експертів $Б_{i\ отр}$		6.5		
Екологічна новизна	Питома вага 0,035	Максимальний бал $Б_{i\ MAX}$		20
1. Рівень екологічної чистоти технології виробництва		0	0	0
2. Рівень впровадження мало- та безвідходних технологій		0	0	0
3. Рівень екологічно небезпечних режимів експлуатації продукції		0	0	0
4. Рівень забруднення навколишнього середовища		0	0	0
Середній бал експертів $Б_{i\ отр}$		0		
Соціальна новизна	Питома вага 0,036	Максимальний бал $Б_{i\ MAX}$		20
1. Використання нового товару приводить до покращення стану здоров’я нації		0	0	0
2. Використання нового товару приводить до зростання доходів населення		0	0	0
3. Виробництво нового товару приводить до збільшення (зменшення) кількості робочих місць на підприємстві		1	2	2
4. Виробництво нового товару приводить до підвищення кваліфікації персоналу		4	3	4
Середній бал експертів $Б_{i\ отр}$		4		
Маркетингова новизна	Питома вага 0,146	Максимальний бал $Б_{i\ MAX}$		20
1. Нові методи маркетингових досліджень		0	0	0
2. Вживання нових стратегій сегментації ринку		3	2	3

Продовження таблиці 5.4

1	*2*	*3*	*4*
3. Вибір нової маркетингової стратегії обхвату і розвитку цільового сегмента	2	1	2
4. Побудова нових каналів збуту	0	0	0
Середній бал експертів $Б_{i\ отр}$	3,25		

Значення *i*-го виду новизни розрахуємо за формулою **[Кавецький практикум 2016]**:

$$I_i = \frac{Б_{i\,отр}}{Б_{i\,MAX}}, \tag{5.1}$$

де $Б_{i\ отр}$ – отримана кількість балів за шкалою оцінок чинників, що визначають *i*-й вид новизни;

$Б_{i\ MAX}$ – максимальна кількість балів, що може бути отримана за *i*-м видом новизни.

Загальний рівень інтегральної новизни розраховуємо шляхом перемноження отриманого значення *i*-го виду новизни на її вагомість, причому вагомість *i*-го виду новизни визначаємо експертним методом, за формулою **[Кавецький практикум 2016]**:

$$N_{інт} = \sum_{i}^{n} W_i \cdot I_i, \tag{5.2}$$

де $N_{інт}$ – рівень інтегральної (сукупної) новизни;

W_i – вагомість (питома вага) *i*-го виду новизни;

n – загальна кількість видів новизни.

$$N_{інт} =$$

$$(0{,}225\cdot 9{,}6/25)+(0{,}217\cdot 21{,}3/30)+(0{,}042\cdot 8{,}2/25)+(0{,}179\cdot 11{,}6/25)+(0{,}12\cdot 6{,}5/20)+$$

$$(0{,}035\cdot 0/20)+(0{,}036\cdot 4/20)+(0{,}146\cdot 3{,}25/20) = 0{,}40.$$

Отримане значення інтегрального рівня новизни зіставляємо зі шкалою, що наведена в табл. 4.5 **[Козловський, Лесько, Кавецький]**.

Таблиця 5.5 – Рівні новизни нового товару та їхня характеристика

Рівні новизни товару	Значення інтегральної новизни	Характеристика товару	Вид нового товару
Найвища	1,00	Абсолютно новий товар	Новий товар, що наділений ознаками інноваційності (інноваційний товар)
Висока	0,8…0,99	Товар, який не має аналогів	
Значуща	0,6…0,79	Принципова зміна споживчих властивостей товару	
Достатня	0,4…0,59	Принципова технологічна модифікація товару	
Незначна	0,2…0,39	Кардинальна зміна параметрів	Новий товар
Помилкова	0,00…0,19	Малоістотна модифікація	

Згідно таблиці 5.5 розробка відповідає рівню при значенні інтегральної новизни 0,40 - достатня новизна; за характеристикою: принципова технологічна модифікація товару; вид розробки - новий товар, що наділений ознаками інноваційності (інноваційний товар).

Розроблене програмне забезпечення базується на проведенні аналізу математичної моделі сучасного контакт-центру та оцінки її основних характеристик. Сформульовано визначення та виконано аналіз вхідних параметрів узагальненої моделі, яка описує в загальному вигляді процес надходження і обслуговування заявок в контакт-центрах. У моделі враховуються такі особливості функціонування діючих та перспективних довідково-інформаційних служб: наявність пристроїв IVR, облік кваліфікації

операторів, наявність місць очікування початку обслуговування у оператора або консультанта, наявність обмеження на максимально можливий час перебування на очікуванні початку обслуговування у оператора або консультанта, можливість повторення заблокованої заявки. Інтервали часу між здійсненням подій в моделі мають експоненціальний розподіл і не залежать один від одного як і ймовірності переходів зі стану в стан. За побудовою функціонування моделі описується марковським процесом. Використання даного класу моделей істотно спрощує оцінку характеристик, зберігаючи при цьому, можливість урахування особливостей формування вхідних потоків заявок, а також дає можливість використовувати результати для обґрунтування вибору значень вхідних параметрів моделі контакт-центру при вирішені завдань планування елементів його інфраструктури та проведенні заходів щодо підвищення ефективності роботи.

5.3 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою **[Козловський, Лесько, Кавецький]**:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (5.3)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	12000	545,45	5	2727,25
Інженер-програміст	10000	454,54	25	11363,5
Дизайнер	8000	363,63	10	3636,3
Всього:				17727,05

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$$З_{до д}= 0,1 \cdot 17727,05 = 1772,7 \text{ (грн.)}$$

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$$Зн = (17727,05+1772,7) \cdot \frac{22}{100} = 4289,9 \text{ (грн.)}.$$

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.7)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

Отримані результати зведемо до таблиці:

Таблиця 5.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
PyCharm	1	1080	1080
Photoshop	1	567	567
Всього			1647,00

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} , \qquad (5.8)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.8 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Величина амортизаційних відрахувань,грн
PyCharm	1080	2/12	2	1080
Photoshop	567	1/12	1	567
Всього:				1647

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Розробка методу та програмних засобів оцінки основних параметрів роботи кол-центрів».

Витрати на матеріали (M), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{в\,j}\,, \qquad (5.9)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, (K_j = *1,1 … 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Чорнила для принтера Epson	475,00	0,2	0,000	0,00	104,5
Всього					104,5

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (5.10)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 5.10 – Витрати на комплектуючі, що були використані для розробки ПЗ

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість, грн.
Пачка паперу	уп.	100	1	100
Ручка	шт.	15	4	60
Всього з урахуванням транспортних витрат				160

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.10)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 4,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.5 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер	0,25	280,0	308,50
Робоче місце розробника п. з.	0,13	200,0	114,4
Робоче місце дизайнера	0,13	80,0	45,7
Всього			468,6

Розрахуємо інші витрати $І_в$.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.11)$$

де $H_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 55%.

Отже, розрахуємо інші витрати:

$$Ів = 0{,}55 * (17727{,}05+1772{,}7) = 10724{,}8(грн).$$

Витрати на проведення науково-дослідної роботи на тему «Розробка методу та програмних засобів оцінки основних параметрів роботи кол-центрів» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (5.12)$$

В = 17727,05+1772,7+4289,9+104,5+160+1647+1647+468,6 = 27816,75 (грн.).

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.13)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{27816{,}75}{0{,}95} = 29280 \text{ (грн.)}.$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\pm \Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}) \quad (5.14)$$

де $\pm\Delta Ц_о$ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році.

N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки;

ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році;

$Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, $Ц_о = Ц_б \pm \Delta Ц_о$;

$Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів;

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2021 році ϑ =18%.

В результаті впровадження результатів наукової розробки інвестор отримає додаткове джерело прибутку, а кількість користувачів, які будуть користуватись розробленим додатком буде збільшуватись: протягом першого року – прогнозується 1000 користувачів, протягом другого року – на 2000 користувачів більше, протягом третього року – на 2500 користувачів В середньому прогнозований прибуток від 1 користувача складає 37,50 грн

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_{2022} = (37{,}5 \cdot 0 + 37{,}5 \cdot 1000) \cdot 0{,}34 = 12750 \text{ грн.}$$

Протягом другого року:

$$\Delta П_{2023} = (37{,}5 \cdot 1000 + 37{,}5 \cdot 2000) \cdot 0{,}34 = 38250 \text{ грн.}$$

Протягом третього року:

$$\Delta П_{2024} = (37{,}5 \cdot 3000 + 37{,}5 \cdot 2500) \cdot 0{,}34 = 70125 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність ПП вкладених інвестицій розраховується за формулою:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \quad (5.15)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

т – період часу, протягом якого виявляються результати впровадженя та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}05 \ldots 0{,}15$;

t – період часу (в роках) від моменту початку впровадження науковотехнічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{12750}{(1+0{,}05)} + \frac{38250}{(1+0{,}05)^2} + \frac{70125}{(1+0{,}05)^3} + \frac{102000}{(1+0{,}05)^4} + \frac{127500}{(1+0{,}05)^5} + \frac{152250}{(1+0{,}05)^6} + \frac{175300}{(1+0{,}05)^7} + \frac{208035}{(1+0{,}05)^8} + \frac{226450}{(1+0{,}05)^8} + \frac{256900}{(1+0{,}05)^{10}} = 981213.18 \text{ (грн.)}$$

На період в 10 років, ПП складатиме 981213.18 грн.

Оскільки $ПП > 0$, то вкладання коштів на виконання та впровадження результатів наукової роботи буде доцільним.

Далі розраховують величину початкових інвестицій PV, які розробник (замовник) має вкласти для здійснення науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{розр}} \cdot ЗВ \qquad (5.16)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 29280 грн.

$$PV = 2 \cdot 29280 = 58560$$

Тоді розрахуємо $E_{\text{абс}}$:

$$\text{Е}_{\text{абс}} = 981213{,}18 - 58560 = 922\,653{,}18 \text{ грн.}$$

Оскільки $E_{\text{абс}} > 0$, то вкладання коштів на виконання та впровадження результатів наукової роботи буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $\text{Е}_\text{в}$ за формулою:

$$\text{Е}_\text{в} = \sqrt[\text{Т}]{1 + \frac{\text{Е}_{\text{абс}}}{\text{PV}}} - 1\,, \qquad (5.17)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[10]{1 + \frac{922\,653{,}18}{58560}} - 1 = 0{,}33 \text{ або } 33\%.$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, , \tag{5.18}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3.$$

$E_в$ = 33%, що більшу ніж $\tau_{мін}$ (30%), це означає, що інвестор може бути зацікавлений у вкладені у розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}, , \tag{5.19}$$

$$T_{ок} = \frac{1}{0{,}33} = 3 \text{ роки}.$$

$T_{ок}$ = 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

ВИСНОВКИ

У магістерській кваліфікаційній роботі були розроблені методи для покращення ігрових механік, працездатність яких демонстрована за топомогою розробленого ігрового Telegram-бота «FireBox» в середовищі розробки Pycharm за допомогою СКБД MySQL та інструментарію SqlAlchemy.

Було розглянуто актуальний стан питання програмних реалізацій Telegram-ботів, а також існуючі аналоги та їх недоліки, зроблено порівняння з власним програмним продуктом та визначено, що є доцільним розробити новий програмний додаток цього типу. Виконано постановку задач для розробки.

При реалізації програми було обрано елементи інтерфейсу надані Telegram API. Розроблено методи та алгоритми роботи програмного продукту, а саме метод автоматизації обчислення характеристик ігрових предметів, який дозволив прискорити балансування характеристик предметів різних типів на різних рівнях, метод генерації варіативних квестів на основі орієнтованого графу, який дозволив покращити їх варіативність, виведено формулу для балансування характеристик пресонажів, яка дозволила оптимально збалансувати межі їх характеристик. Розроблено основні модулі програми.

Було виконано варіантний аналіз та обґрунтування вибору програмних засобів при розробці програми, обрано мову програмування Python, середовище розробки Pycharm, СКБД MySQL, та інструментарій SQLAlchemy для реалізації поставленої задачі. В процесі розробки було описано програмну реалізацію основних модулів додатку.

Було розглянуто види та методики тестування, обрано методику тестування «чорної скриньки», складено набір тест-кейсів для тестування програми, які довели повну працездатність даного програмного продукту та відповідність поставленому технічному завданню. Розроблено інструкцію користувача та вимоги до персонального комп’ютера користувача.

Проведено оцінювання комерційного потенціалу розробки, яке показало, що розробка має достатній рівень комерційного потенціалу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Massively multiplayer online role-playing game [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/mmo_rpg

2. Найпопулярніші жанри мобільних ігор 2020-2021 [Електронний ресурс] – Режим доступу до ресурсу: https://koloro.ua/blog/brending-i-marketing/samye-populyarnye-zhanry-mobilnykh-igr-2020-2021.html

3. Цукрук В. І., Романюк О.В. Розрахунок бойових характеристик персонажів ігрового Telegram-боту [Електронний ресурс] // Тези доповідей XIV науково-практичній конференції «Інформаційні технології і автоматизація» – Одеса 2021 – Режим доступу до ресурсу: https://www.onaft.edu.ua/download/konfi/2021/Collection-of-abstracts-of-the-conference-ITIA-2021.pdf

4. Цукрук В. І. Розробка алгоритму генерації варіативних квестів на основі орієнтованого графу [Електронний ресурс] // Міжнародна науково-практична інтернет конференція «Електронні інформаційні ресурси: створення, використання, доступ» – Вінниця 2021 – Режим доступу: https://conferences.vntu.edu.ua/index.php/eir/all-ebmd-2021/index

5. Цукрук В. І., Романюк О.В. Розрахунок бойових характеристик персонажів ігрового Telegram боту [Електронний ресурс] // Тези доповідей XI Міжнародної науково-технічної конференції «Інформаційно-комп’ютерні технології – 2020 (ІКТ-2020)», м. Житомир, 09 - 11 квітня 2020 р. – Житомир: Житомирська політехніка, 2020. – Режим доступу до ресурсу: https://conf.ztu.edu.ua/informatsijno-kompyuterni-tehnologiyi/

6. Ігровий баланс [Електронний ресурс] – Режим доступу до ресурсу: [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%86%D0%B3%D1%80%D0%BE%D0%B2%D0%B8%D0%B9_%D0%B1%D0%B0%D0%BB%D0%B0%D0%BD%D1%81

7. Метагейм [Електронний ресурс] – Режим доступу до ресурсу: http://wiki.rpgverse.ru/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%B3%D0%B5%D0%B9%D0%BC

8. 50 игровых механик, заставляющих возвращаться в MMO [Електронний ресурс] – Режим доступу до ресурсу: https://gamesmaker.ru/industry/common/50-igrovyh-mehanik-zastavlyayuschih-vozvraschatsya-v-mmo/

9. Chat Wars [Електронний ресурс] – Режим доступу до ресурсу: https://chatwars.me/

10. Бот Telegram RF Telegram [Електронний ресурс] – Режим доступу до ресурсу: https://telegram.org.ru/8406-rf-telegram.html

11. Игра Hyperion [Beta] [Електронний ресурс] – Режим доступу до ресурсу: https://telegram.me/HyperionGameBot?start=XRAS

12. Identifying MMORPG Bots [Електронний ресурс] – Режим доступу до ресурсу: https://link.springer.com/article/10.1155/2009/797159

13. Методология балансирования игр [Електронний ресурс] – Режим доступу до ресурсу: https://gdcuffs.com/balance-methods/

14. Обзор существующих подходов к динамической балансировке сложности игрового процесса [Електронний ресурс] – Режим доступу до ресурсу: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/Npdntu_inf_2016_2_14.pdf

15. Балансируем механику в играх [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/268033/

16. Процедурна генерація [Електронний ресурс] – Режим доступу до ресурсу: https://uk.freejournal.org/4015703/1/protsedurna-generatsiya.html

17. Playliner: какими бывают ежедневные задания в мобильных играх [Електронний ресурс]. Режим доступу: https://app2top.ru/game_development/playliner-kakimi-by-vayut-ezhednevny-e-zadaniya-v-mobil-ny-h-igrah-87921.html.

18. Особливості розробки Telegram бота с Google API в Docker [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/319016/

19. Огляд і основи мови програмування C++ [Електронний ресурс] – Режим доступу до ресурсу: http://www.znannya.org/?view=Cpp_basics

20. Stroustrup B. The C++ Programming Language, 4th edition / B. Stroustrup – Addison-Wesley, 2013 – 1368 ст.

21. Кравець П.О. Об'єктно-орієнтоване програмування: навч. посібник / П.О. Кравець - Львів:Видавництво Львівської політехніки, 2012– 624с.

22. Герберт Шилдт. Java Полное руководство: навч. Посібник / Шилдт Герберт – Диалектика Вильямс, 2019. – 44 с.

23. C Sharp [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/C_Sharp

24. Васильєв Олексій. Програмування мовою Python: навч. посібник / О. Васильєв – Київ:Навчальна книга – Богдан, 2019. – 504с.

25. Алгоритмы: построение и анализ / Т.Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн; пер. с англ. И. Красникова – Москва: ООО "И.Д. Вильямс", 2013. – 1328 с.

26. MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MongoDB

27. Тейлор Дж. Ален. SQL для чайников: навч. посібник / Аллен Дж. Тейлор – Москва: МСК, 2019. – 544с.

28. Романюк О. Н. Організація баз даних і знань. / О.Н. Романюк // Навчальний посібник. – Вінниця: УНІВЕРСУМ – Вінниця. – 2003. – 123 с.

29. SQLAlchemy [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQLAlchemy

30. Michael Kennedy & Matt Harrison. Effective PyCharm / Kennedy Michael, Harrison Matt – Copyrighted Material, 2019. – 12 с.

31. Spyder – The Scientific Python Development Enviroment [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/spyder/

32. Брюс Джонсон. Professional Visual Studio: навч. Посібник / Джонсон Брюс – Лондон:Wiley/Wrox, 2013. – 403с.

33. Калбертсон Р. Быстрое тестирование. / Р. Калбертсон, К. Браун, Г. Кобб. – М.: «Вильямс»,2002. – 374 с.–ISBN 5-8459-0336-X.

34. Тестирование програ́ммного обеспечения. [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Тестирование_ программного _обеспечения

35. Бейзер Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. / Б. Бейзер — СПб.: Питер, 2004. — 320 с.

36. «Летающий» мессенджер Telegram — обзор и инструкция по установке [Електронний ресурс] – Режим доступу до ресурсу: https://vkoshelek.com/letayushhij-messendzher-telegram-obzor-i-instruktsiya-po-ustanovke/

ДОДАТКИ

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"____ "_____2021 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методів і програмних засобів для підвищення ефективності ігрових механізмів на основі Telegram-ботів» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ д.т.н., проф. О.Н.Романюк

"____"_____2021 р.

Виконав:

_____студент гр.2ПІ-20м В.І. Цукрук

"____"_____2021 р.

Вінниця – 2021 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і програмних засобів для підвищення ефективності ігрових механізмів на основі Telegram-ботів».

Галузь застосування – системи миттєвих повідомлень.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ №277 ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності ігрових механізмів та їх балансування.

Призначення роботи – розробка методів і алгоритмів для підвищення ефективності ігрових механізмів і ігрового Telegram-бота.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Ігровий баланс [Електронний ресурс] – Режим доступу до ресурсу: [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%86%D0%B3%D1%80%D0%BE%D0%B2%D0%B8%D0%B9_%D0%B1%D0%B0%D0%BB%D0%B0%D0%BD%D1%81

2. Процедурна генерація [Електронний ресурс] – Режим доступу до ресурсу: https://uk.freejournal.org/4015703/1/protsedurna-generatsiya.html

3. Особливості розробки Telegram бота с Google API в Docker [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/319016/

4. Романюк О. Н. Організація баз даних і знань. / О.Н. Романюк // Навчальний посібник. – Вінниця: УНІВЕРСУМ – Вінниця. – 2003. – 123 с.

4. Технічні вимоги

Вихідні дані до роботи:_модель розробки – водоспадна; метод передачі повідомлень між серверами – LongPolling; метод балансування характеристик – транзитивний; метод створення ігрових об'єктів – від характеристик, метод генерації квестів – на основі орієнтованого графу; вхідні дані – інформація про неконтрольованих гравцями персонажів, локації та шаблони предметів у форматі «.xlsx»; ідентифікаційний номер Telegram-бота у відповідному форматі; ідентифікатор користувача; вихідні дані – результати виконання алгоритмів, як відповідь на команди гравця у текстовому фоматі

5. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред’являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

7. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

8. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз стану питання та постановка задач дослідження	15.09.2021-26.09.2021
2	Розробка методів та алгоритмів для підвищення ефективності ігрових механізмів	27.092021-15.10.2021
3	Розробка програмних компонент для ігрового telegram-боту	16.10.2021-7.11.2021
4	Тестування програми	8.11.2021-21.11.2021
5	Економічна частина	22.11.2021-30.11.2021

9. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка методів і програмних засобів для підвищення ефективності ігрових механізмів на основі Telegram-ботів.**

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 2ПІ – 20м

Науковий керівник: к.т.н. доц. Романюк О. В.

Unicheck	
Оригінальність	**86,7%**
Схожість	13,5%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлена з повним звітом подібності, який був згенерований Системою щодо роботи «Розробка методів і програмних засобів для підвищення ефективності ігрових механізмів на основі Telegram-ботів».

Автор _____ Цукрук Валентин Іванович

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку _____ Черноволик Г. О.
(підпис) (прізвище, ініціали)

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)

ДОДАТОК В

ЛІСТИНГ КОДУ

В.1 Файл server.py

```
# -*- coding: utf-8 -*-
import logging
import re
import time


import os


from config import session, bot, engine, APP_ROOT, dp
from keyboards import keyboard_main_town, keyboard_forest, keyboard_location,
keyboard_arena


from models.fight import PvP, big_PvP, enemy_found, PvE


from models.location import Location, Time
from models.npc import NPC, mob_found
from models.parser import all_loader
from models.person import Base, Person, Progress
from models.Items import show_backpack, wear, PersonalItem,
generate_personal_item, Drop, DefaultItem, Resource, \
    add_res, use_item
from tasks import move_to, send_message_per_time
```

```
from aiogram import types
from aiogram.utils import executor
from aiogram.dispatcher import filters


def get_person(user_id):
    return session.query(Person).filter_by(user_id=user_id).first()


def extract_arg(arg):
    return arg.split('_')[1]


@dp.message_handler(commands=['start'])
async def send_welcome(message):
    if get_person(message.from_user.id):
        await bot.send_message(message.chat.id, u"Так ты уже существуешь")
    else:
        await bot.send_message(message.chat.id, u"Для начала игры выбирите свой ник (команда /name и ник через пробел))")


@dp.message_handler(commands=['name'])
async def name(message):
    name = message.text.split()[1:]
    if name:
```

```
        if get_person(message.from_user.id):
            await bot.send_message(message.chat.id, u"Так ты уже существуешь")
        else:
            active_person = Person(user_id=message.from_user.id, name=name,
chat_id=message.chat.id)
            session.add(active_person)
            session.commit()
            await bot.send_message(message.chat.id, u"Всё, салага, теперь тебя зовут
'{0}'".format(name),
                                   reply_markup=keyboard_main_town)
    else:
        await bot.send_message(message.chat.id, u"Недопустимый ник")



@dp.message_handler(commands=['backpack'])
async def backpack(message):
    await show_backpack(get_person(message.from_user.id))



@dp.message_handler(filters.RegexpCommandsFilter(regexp_commands=['(^/use_\
w{22}$)']))
async def wear_command(message, regexp_command):
    pers = get_person(message.from_user.id)
    if not pers.is_die:
        await use_item(pers, extract_arg(message.text))
    else:
        await bot.send_message(message.chat.id, u"Мертвые лежат смирно, скотина")
```

```
@dp.message_handler(filters.RegexpCommandsFilter(regexp_commands=['(^/wear_\w{22}$)']))
async def wear_command(message, regexp_command):
    pers = get_person(message.from_user.id)
    if not pers.is_die:
        await wear(pers, extract_arg(message.text))
    else:
        await bot.send_message(message.chat.id, u"Мертвые лежат смирно, скотина")


@dp.message_handler(filters.RegexpCommandsFilter(regexp_commands=['(^/unwear_\w{22}$)']))
async def unwear_command(message, regexp_command):
    pers = get_person(message.from_user.id)
    if not pers.is_die:
        await wear(pers, extract_arg(message.text))
    else:
        await bot.send_message(message.chat.id, u"Мертвые лежат смирно, скотина")


@dp.message_handler(filters.RegexpCommandsFilter(regexp_commands=['(^/info_\w{22}$)']))
async def info_command(message, regexp_command):
    uuid = extract_arg(message.text)
    if uuid[0] == 'i':
```

```
        item =
session.query(PersonalItem).filter_by(uuid=extract_arg(message.text)).first()
    else:
        item = session.query(Resource).filter_by(uuid=extract_arg(message.text)).first()
    info, image = item.get_info()
    if image:
        try:
            await bot.send_photo(chat_id=message.chat.id, caption=info,
photo=open(image, 'rb'))
        except:
            await bot.send_message(message.chat.id, info)
    else:
        await bot.send_message(message.chat.id, info)



@dp.message_handler(commands=['new_info'])
async def new_info(message):
    all_loader()



@dp.message_handler(commands=['check_info'])
async def check_info(message):
    for t in session.query(Time):
        await bot.send_message(message.chat.id, t.from_)
    for item in session.query(DefaultItem):
        await bot.send_message(message.chat.id, item.name)
```

```
@dp.message_handler(commands=['add_mirror'])
async def add_mirror(message):
    active_person = Person(user_id='mirror', name='mirror', max_hp=220000, hp=220000, damage_min_physic=7200,
                           damage_max_physic=10000, dodge=35, defence_physic=5000, crit=21, accuracy=31,
                           penetration_physic=680, temp_attack=15, count_for_combo=2, coef_for_combo=1.2)
    session.add(active_person)
    session.commit()


@dp.message_handler(commands=['Chit_admin_1232145325423532'])
async def cheat(message):
    pers = get_person(message.from_user.id)
    pers.max_hp *= 10
    pers.is_die = False
    pers.hp = pers.max_hp
    session.add(pers)
    session.commit()


@dp.message_handler(commands=['buttons'])
async def start_buttons(message):
    pers = get_person(message.from_user.id)
```

```
    if pers.location == u'Город':
        await bot.send_message(message.chat.id, u"Ты находишься в локации 'Город'", reply_markup=keyboard_main_town)
    elif pers.location == u'Лес':
        await bot.send_message(message.chat.id, u"Ты находишься в локации 'Лес'", reply_markup=keyboard_forest)
    elif pers.location == u'Арена':
        await bot.send_message(message.chat.id, u"Ты находишься в локации 'Арена'", reply_markup=keyboard_arena)
    else:
        pers.location = u'Город'
        await bot.send_message(message.chat.id, u"Ты перемещен в локацию 'Город'", reply_markup=keyboard_main_town)



async def show_persons(pers):
    # our_users = session.query(Person).filter_by(name='tester')
    await bot.send_message(pers.chat_id, '\nYour User:')
    await bot.send_message(pers.chat_id, "'{0}' id:{1} lvl={2} exp={3} hp={4}/{5}".format(pers.name,
                                                                                          pers.user_id,
                                                                                          pers.lvl,
                                                                                          pers.experience, pers.hp,
                                                                                          pers.max_hp))



@dp.message_handler(commands=['big_mirror_fight'])
```

```
async def big_mirror_fight(message):
    # our_users = session.query(Person).filter_by(name='tester')
    pers = get_person(message.from_user.id)
    await big_PvP(pers.user_id, pers.user_id, message.chat.id)



@dp.message_handler(filters.RegexpCommandsFilter(regexp_commands=['(^/test_fight_\w+$)']))
async def test_fight_command(message, regexp_command):
    try:
        # pers = get_person(message.from_user.id)
        # if not pers.is_die and not pers.is_in_action:
        #     PvP(pers, get_person(extract_arg(message.text)))
        # else:

        lvl = message.text.split('_')[2]
        lvl_2 = message.text.split('_')[3]
        counts = message.text.split('_')[4]
        progress = session.query(Progress).filter_by(lvl=lvl).first()
        progress_2 = session.query(Progress).filter_by(lvl=lvl_2).first()
        temp_pers_1 = session.query(Person).filter_by(user_id="tester_1").first()
        temp_pers_2 = session.query(Person).filter_by(user_id="tester_2").first()
        try:
            session.delete(temp_pers_1)
            session.delete(temp_pers_2)
            session.commit()
```

```
    except Exception: pass

    temp_pers_1 = Person(user_id="tester_1", name="tester_1",
max_hp=progress.max_hp, damage_min_physic=progress.damage_min_physic,

                         damage_max_physic=progress.damage_max_physic,
dodge_physic=progress.dodge_physic,

                         dodge_magic=progress.dodge_magic,
defence_physic=progress.defence_physic,

                         defence_magic=progress.defence_magic,
crit_physic=progress.crit_physic,

                         crit_magic=progress.crit_magic,
penetration_magic=progress.penetration_magic,

                         penetration_physic=progress.penetration_physic,
chat_id=message.chat.id, accuracy_physic=0,

                         accuracy_magic=0)

    temp_pers_2 = Person(user_id="tester_2", name="tester_2",
max_hp=progress_2.max_hp, damage_min_physic=progress_2.damage_min_physic,

                         damage_max_physic=progress_2.damage_max_physic,
dodge_physic=progress_2.dodge_physic,

                         dodge_magic=progress_2.dodge_magic,
defence_physic=progress_2.defence_physic,

                         defence_magic=progress_2.defence_magic,
crit_physic=progress_2.crit_physic,

                         crit_magic=progress_2.crit_magic,
penetration_magic=progress_2.penetration_magic,

                         penetration_physic=progress_2.penetration_physic,
chat_id=message.chat.id, accuracy_physic=0,

                         accuracy_magic=0)

    session.add(temp_pers_1, temp_pers_2)

    session.commit()

    if int(counts) == 1:
```

```
            await PvP(temp_pers_1, temp_pers_2)
        else:
            await big_PvP(temp_pers_1, temp_pers_2, int(counts))
    except:
        await bot.send_message(message.chat.id, "Проверь команду еще раз")


# @dp.message_handler(reg_exp='/fight_\w{12}')
# def fight_command(message):
#     # pers = get_person(message.from_user.id)
#     # if not pers.is_die and not pers.is_in_action:
#     #     PvP(pers, get_person(extract_arg(message.text)))
#     # else:
#     bot.send_message(message.chat.id, u"Ты видимо чем-то занят")


@dp.message_handler(content_types=['text'])
async def button_analys(message):
    # try:
        pers = get_person(message.from_user.id)
        if message.text == u"Профиль" or message.text == u'/hero':
            await show_persons(pers)
        elif message.text == u"Инвентарь":
            await show_backpack(pers)
        elif message.text == u"Локации":
```

```
            if not pers.is_die and not pers.is_in_action:
                await bot.send_message(message.chat.id, u"Пора выдвигаться",
reply_markup=keyboard_location)
            else:
                await bot.send_message(message.chat.id, u"Ты видимо чем-то занят")
        elif session.query(Location).filter_by(name=message.text).first():
            await move_to(message.text, message.from_user.id, message.chat.id)
        elif message.text == u"Поиск противника" and pers.location == u'Арена':
            if not pers.is_die and not pers.is_in_action:
                await enemy_found(pers.user_id, pers.location, pers.chat_id)
            else:
                await bot.send_message(message.chat.id, u"Ты видимо чем-то занят")
        elif message.text == u"Поиск монстра":
            if not pers.is_die and not pers.is_in_action:
                await mob_found(pers)
            else:
                await bot.send_message(message.chat.id, u"Ты видимо чем-то занят")
        elif message.text == u"Пополнить здоровье❤□":
            if pers.location == u"Город":
                pers.hp = pers.max_hp
                await bot.send_message(message.chat.id, "{0}/{0}".format(pers.max_hp))
            else:
                await bot.send_message(message.chat.id, u"Думал самый хитрый,
да?".format(pers.max_hp))
        elif message.text == u"Атаковать":
            if not pers.is_die and not pers.is_in_action:
```

```
            if pers.monster_for_fight and session.query(NPC).filter_by(
                    name=pers.monster_for_fight).first().location == pers.location:
                await PvE(pers, pers.monster_for_fight)
            else:
                await bot.send_message(message.chat.id, u"Ты с уверенностью разрубил ветер и почувствовал прилив сил")
        else:
            await bot.send_message(message.chat.id, u"Ты видимо чем-то занят")
        session.add(pers)
        session.commit()
    # except:
    #     await bot.send_message(message.chat.id, u"Что-то не так, возможно стоит нажать команду /start")


if __name__ == '__main__':
    Base.metadata.create_all(engine)
    executor.start_polling(dp)
```

B.2 Файл tasks.py

```
# -*- coding: utf-8 -*-
import asyncio

from config import BROKER_URL, session, bot
from keyboards import keyboard_main_town
from models.location import Time, wait
from models.person import Person
from aiogram import types
```

```
async def pers_death(user_id):
    pers = session.query(Person).filter_by(user_id=user_id).first()
    await bot.send_message(pers.chat_id, u"Ты пагип")
    pers.is_die = True
    session.add(pers)
    session.commit()
    await asyncio.sleep(60)
    await rise(pers)


async def send_message_per_time(time, chat_id, message, reply_markup=None):
    await asyncio.sleep(time)
    await send_modified_messge(chat_id, message, reply_markup)


async def send_modified_messge(chat_id, message, reply_markup):
    if reply_markup:
        reply_markup = types.ReplyKeyboardMarkup(True, True).row(reply_markup)
    await bot.send_message(chat_id, message, parse_mode="Markdown", reply_markup=reply_markup)


async def move_to(location_name, user_id, chat_id):
    pers = session.query(Person).filter_by(user_id=user_id).first()
    previous_loc = pers.location if pers.location else u"Город"
```

```
        time_to = session.query(Time).filter_by(from_=previous_loc, to_=location_name).first()
        if not time_to:
            await bot.send_message(chat_id, u"Такой локации либо не существует, либо пройти в неё тебе не судьба")
        await bot.send_message(chat_id, u"Ты отправился в путь")
        pers.monster_for_fight = None
        pers.is_in_action = True
        session.add(pers)
        session.commit()
        await asyncio.sleep(time_to.time)
        await wait(pers, location_name, chat_id)


    async def rise(pers):
        pers.location = u"Город"
        pers.is_die = False
        pers.hp = pers.max_hp
        session.add(pers)
        session.commit()
        await bot.send_message(pers.chat_id, u"Ты воскрес", reply_markup=keyboard_main_town)
        return
```

В.3 Файл keyboards.py

```
# -*- coding: utf-8 -*-

from aiogram import types

keyboard_main_town = types.ReplyKeyboardMarkup(True, True)
```

```
keyboard_main_town.row(u"Профиль", u"Локации", u"Пополнить здоровье ❤□", u"Инвентарь")
keyboard_location = types.ReplyKeyboardMarkup(True, True)
keyboard_location.row(u"Лес", u"Арена")
keyboard_forest = types.ReplyKeyboardMarkup(True, True)
keyboard_forest.row(u"Профиль", u"Город", u"Поиск монстра", u"Атаковать")
keyboard_arena = types.ReplyKeyboardMarkup(True, True)
keyboard_arena.row(u"Профиль", u"Поиск противника", u"Город")
```

В.4 Файл curve.py

```
def getMinMax(lvl):
    temp = 64
    Min = 0
    for i in range(1, lvl):  # 121
        print("(", i, ";", temp, ")", ",")
        temp = round(temp * 1.1)
        if i == lvl-1:
            Min = temp
    Max = temp
    return Min, Max
```

В.5 Файл config.py

```
import os

import sqlalchemy
from sqlalchemy import create_engine
from sqlalchemy.engine.url import URL
```

```
from sqlalchemy.ext.declarative import declarative_base

from aiogram import Bot
from aiogram.dispatcher import Dispatcher

token = '818620538:AAFqpU78oibiiXMH6iFGukVgX2tQodp88R0'
bot = Bot(token=token)
dp = Dispatcher(bot)

APP_ROOT = os.path.dirname(os.path.abspath(__file__))

Base = declarative_base()

db_url = {
    'database': "testI",
    'drivername': 'mysql',
    'username': 'root',
    'password': 'jazabilparol',
    'host': '127.0.0.1',
    'query': {'charset': 'utf8'},  # the key-point setting
}

engine = create_engine(URL(**db_url), encoding="utf8")

# engine = create_engine("mysql://root:jazabilparol@localhost:3306/testI?charset=utf8mb4")

Base.metadata.create_all(engine)
Session = sqlalchemy.orm.sessionmaker()
Session.configure(bind=engine)
```

```
session = Session()
session = None
BROKER_URL = 'pyamqp://guest@localhost//'



# For balance  constants

coefCrit = 2
```

B.6 Файл choices.py

```
# -*- coding: utf-8 -*-
rarity = {u'Обычный': u'□',
          u'Необычный': u'□',
          u'Редкий': u'',
          u'Эпический': u'□',
          u'Легендарный': u'□'}

for_info_type = {
    u"weapon": u"оружие",
    u"body": u"на торс",
    u"head": u"на голову",
    u"hands": u"на руки",
    u"legs": u"но ноги",
    u"feet": u"на ступни",
    u"neck": u"на шею",
    u"finger": u"на палец",
}
```

```
class_eng_ru = {
    u"sword": u"Меч",
    u"light armor": u"Легкая броня",
    u"potion": u"зелье",
}

attr_emotion ={
    u"hp": u"❤"
}
```

В.7 Файл person.py

```
# -*- coding: utf-8 -*-

import sqlalchemy
import sqlalchemy.orm as ORM
from config import Base, bot, session
from keyboards import keyboard_main_town, keyboard_forest, keyboard_arena


proff_person_association = sqlalchemy.Table(
    'proff_person', Base.metadata,
    sqlalchemy.Column('user_id', sqlalchemy.String(length=12), sqlalchemy.ForeignKey('persons.user_id')),
    sqlalchemy.Column('proff_name', sqlalchemy.String(length=12), sqlalchemy.ForeignKey('proff.name'))
)
```

```
    class Proffesion(Base):
        __tablename__ = 'proff'

        name = sqlalchemy.Column(sqlalchemy.String(length=12), primary_key=True)


    class Person(Base):
        __tablename__ = 'persons'

        # main information
        user_id = sqlalchemy.Column(sqlalchemy.String(length=12), primary_key=True)
        chat_id = sqlalchemy.Column(sqlalchemy.String(length=12))
        name = sqlalchemy.Column(sqlalchemy.String(length=50))
        # race_name = sqlalchemy.Column(sqlalchemy.String(length=12), sqlalchemy.ForeignKey('race.name'))
        # race = ORM.relationship("Race", backref=ORM.backref("persons"))
        # class_name = sqlalchemy.Column(sqlalchemy.String(length=12), sqlalchemy.ForeignKey('class.name'))
        # clas_s = ORM.relationship("Clas_s", backref=ORM.backref("persons"))
        # lvl information
        lvl = sqlalchemy.Column(sqlalchemy.Integer, default=1)
        current_energy = sqlalchemy.Column(sqlalchemy.Integer, default=5)
        max_energy = sqlalchemy.Column(sqlalchemy.Integer, default=5)
        experience = sqlalchemy.Column(sqlalchemy.Float, default=0)
        experience_to_next_lvl = sqlalchemy.Column(sqlalchemy.Integer, default=100)
        # skills = sqlalchemy.Column(sqlalchemy.ARRAY(sqlalchemy.String(length=12)))
```

```
        # professions = ORM.relationship("Proffesion", secondary=proff_person_association)
        # stats
        hp = sqlalchemy.Column(sqlalchemy.Float, default=100)
        max_hp = sqlalchemy.Column(sqlalchemy.Float, default=100)
        damage_min_physic = sqlalchemy.Column(sqlalchemy.Float, default=10)
        damage_max_physic = sqlalchemy.Column(sqlalchemy.Float, default=15)
        damage_min_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
        damage_max_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
        defence_physic = sqlalchemy.Column(sqlalchemy.Float, default=0)
        defence_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
        penetration_physic = sqlalchemy.Column(sqlalchemy.Float, default=0)
        penetration_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
        accuracy_physic = sqlalchemy.Column(sqlalchemy.Float, default=0)
        accuracy_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
        dodge_magic = sqlalchemy.Column(sqlalchemy.Float, default=5)
        dodge_physic = sqlalchemy.Column(sqlalchemy.Float, default=5)
        crit_magic = sqlalchemy.Column(sqlalchemy.Float, default=5)
        crit_physic = sqlalchemy.Column(sqlalchemy.Float, default=5)
        temp_attack = sqlalchemy.Column(sqlalchemy.Float, default=0)
        count_for_combo = sqlalchemy.Column(sqlalchemy.Integer, default=5)
        coef_for_combo = sqlalchemy.Column(sqlalchemy.Integer, default=1)
        temp_count_for_combo = sqlalchemy.Column(sqlalchemy.Integer, default=0)
        # money
        gold = sqlalchemy.Column(sqlalchemy.Integer, default=1000)
        donat_value = sqlalchemy.Column(sqlalchemy.Integer, default=0)
        # body
        weapon = sqlalchemy.Column(sqlalchemy.String(length=40))
        head = sqlalchemy.Column(sqlalchemy.String(length=40))
```

```
        body = sqlalchemy.Column(sqlalchemy.String(length=40))
        legs = sqlalchemy.Column(sqlalchemy.String(length=40))
        finger_1 = sqlalchemy.Column(sqlalchemy.String(length=40))
        finger_2 = sqlalchemy.Column(sqlalchemy.String(length=40))
        neck = sqlalchemy.Column(sqlalchemy.String(length=40))
        hands = sqlalchemy.Column(sqlalchemy.String(length=40))
        feet = sqlalchemy.Column(sqlalchemy.String(length=40))
        backpack_amount = sqlalchemy.Column(sqlalchemy.Integer, default=0)
        backpack_amount_max = sqlalchemy.Column(sqlalchemy.Integer,
default=50)
        # pets
        maunt = sqlalchemy.Column(sqlalchemy.String(length=12))
        pet = sqlalchemy.Column(sqlalchemy.String(length=12))
        # other info
        group = sqlalchemy.Column(sqlalchemy.String(length=12), default=0)
        location = sqlalchemy.Column(sqlalchemy.String(length=12))
        monster_for_fight = sqlalchemy.Column(sqlalchemy.String(length=50))
        is_in_action = sqlalchemy.Column(sqlalchemy.Boolean(), default=False)
        is_die = sqlalchemy.Column(sqlalchemy.Boolean(), default=False)
        # global_status = sqlalchemy.Column(sqlalchemy.String(length=3),
default="001")

        def __repr__(self):
            return "<Person(name='{0} lvl={1} exp={2}')>".format(self.name,
self.lvl, self.experience)

        def add_experience(self, exp):
            self.experience += exp
            # while (self.experience >= self.experience_to_next_lvl):
            #     self.lvl += 1
```

```
            #                                   self.experience_to_next_lvl =
session.query(Progress).filter_by(user_id=user_id).first()


        # def get_global_status(message):
        #                                                person =
session.query(Person).filter_by(user_id=message.from_user.id).first()
        #     return person.global_status
        #

        def wait(pers, location_name=None, chat_id=None):
            pers.is_in_action = False
            pers.location = location_name
            session.add(pers)
            session.commit()
            keyboard = keyboard_main_town
            if location_name == u'Город':
                keyboard = keyboard_main_town
            elif location_name == u'Лес':
                keyboard = keyboard_forest
            elif location_name == u'Арена':
                keyboard = keyboard_arena
            bot.send_message(chat_id, u"Ты пришел в локацию
'{}'".format(location_name), reply_markup=keyboard)


        class Progress(Base):
            __tablename__ = 'progress'

            lvl = sqlalchemy.Column(sqlalchemy.Integer(), primary_key=True)
```

```
    experience_to_next_lvl = sqlalchemy.Column(sqlalchemy.Integer())
    max_hp = sqlalchemy.Column(sqlalchemy.Float, default=100)
    damage_min_physic = sqlalchemy.Column(sqlalchemy.Float, default=15)
    damage_max_physic = sqlalchemy.Column(sqlalchemy.Float, default=15)
    defence_physic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    defence_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    penetration_physic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    penetration_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    dodge_physic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    dodge_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    crit_physic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    crit_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)

    def __repr__(self):
        return "lvl:{0} exp:{1} hp: {2}".format(self.lvl, self.experience_to_next_lvl, self.max_hp)
```

B.8 Файл parser.py

```
import os

import shortuuid
from openpyxl import load_workbook

from config import APP_ROOT, session
from models.Items import Resource, DefaultItem, Drop, Resource
from models.location import Location, Time
from models.npc import NPC
from models.person import Progress
```

```
def all_loader():
    locations = os.path.join(APP_ROOT, 'Data/Locations.xlsx')
    npc = os.path.join(APP_ROOT, 'Data/NPC.xlsx')
    progress = os.path.join(APP_ROOT, 'Data/Progress.xlsx')
    items = os.path.join(APP_ROOT, 'Data/Items.xlsx')
    drop = os.path.join(APP_ROOT, 'Data/Drop.xlsx')
    resources = os.path.join(APP_ROOT, 'Data/Resources.xlsx')
    wb = load_workbook(locations)
    ws = wb.active
    for i, row in enumerate(ws.iter_rows()):
        if i == 0:
            continue
        loc_from = None
        loc_to = None
        if not session.query(Location).filter_by(name=row[0].value).first():
            loc_from = Location(name=row[0].value)
            session.add(loc_from)
            loc_to = Location(name=row[1].value)
            if not session.query(Location).filter_by(name=row[1].value).first():
                session.add(loc_to)
        elif not session.query(Location).filter_by(name=row[1].value).first():
            loc_from = Location(name=row[0].value)
            loc_to = Location(name=row[1].value)
            session.add(loc_to)
        if loc_to and loc_from:
            time_to = row[2].value
            time_from = row[3].value
            time_1 = None
            time_2 = None
```

```
                time_ = session.query(Time).filter_by(from_=loc_from.name,
to_=loc_to.name).first()
                if time_ and time_.time == time_to:
                    pass
                elif time_:
                    time_.time = time_to
                else:
                    time_1 = Time(from_=loc_from.name, to_=loc_to.name,
time=time_to)
                time_ = session.query(Time).filter_by(from_=loc_to.name,
to_=loc_from.name).first()
                if time_ and time_.time == time_from:
                    pass
                elif time_:
                    time_.time = time_from
                else:
                    time_2 = Time(from_=loc_to.name, to_=loc_from.name,
time=time_from)
                if time_1:
                    session.add(time_1)
                if time_2:
                    session.add(time_2)
            wb = load_workbook(npc)
            ws = wb.active
            for i, row in enumerate(ws.iter_rows()):
                if i == 0:
                    continue
                if not session.query(NPC).filter_by(name=row[0].value).first():
                    mob = NPC(name=row[0].value)
                    mob.location = row[1].value
```

```
            mob.chance = row[2].value
            mob.fraction = row[3].value
            mob.lvl = row[4].value
            mob.max_hp = row[5].value
            mob.damage_min_physic = row[6].value
            mob.damage_max_physic = row[7].value
            mob.damage_min_magic = row[8].value
            mob.damage_max_magic = row[9].value
            mob.defence_physic = row[10].value
            mob.defence_magic = row[11].value
            mob.penetration_physic = row[12].value
            mob.penetration_magic = row[13].value
            mob.dodge_physic = row[14].value
            mob.dodge_magic = row[15].value
            mob.crit_physic = row[16].value
            mob.crit_magic = row[17].value
            mob.gold_max = row[18].value.split("-")[1]
            mob.experience = row[19].value
            mob.chance_to_physic = row[20].value
            mob.gold_min = row[18].value.split("-")[0]
            session.add(mob)
    wb = load_workbook(progress)
    ws = wb.active
    for i, row in enumerate(ws.iter_rows()):
        if i == 0:
            continue
        if not session.query(Progress).filter_by(lvl=row[0].value).first():
            progress = Progress(lvl=row[0].value)
            progress.experience_to_next_lvl = row[1].value
            progress.max_hp = row[2].value
```

```
            progress.damage_min_physic = row[3].value
            progress.damage_max_physic = row[4].value
            progress.defence_physic = row[5].value
            progress.defence_magic = row[5].value
            progress.dodge_physic = row[6].value
            progress.dodge_magic = row[6].value
            progress.crit_physic = row[7].value
            progress.crit_magic = row[7].value
            session.add(progress)
            session.commit()
    wb = load_workbook(items)
    ws = wb.active
    for i, row in enumerate(ws.iter_rows()):
        if i == 0:
            continue
        if not session.query(DefaultItem).filter_by(name=row[0].value).first():
            item = DefaultItem(name=row[0].value)
            item.type = row[1].value
            item.clas_s = row[2].value
            item.damage_physic_min = row[3].value
            item.damage_physic_max = row[4].value
            item.damage_magic_min = row[5].value
            item.damage_magic_max = row[6].value
            item.defence_physic_min = row[7].value
            item.defence_physic_max = row[8].value
            item.defence_magic_min = row[9].value
            item.defence_magic_max = row[10].value
            item.dodge_physic = row[11].value
            item.dodge_magic = row[12].value
            item.crit_physic = row[13].value
```

```
                item.crit_magic = row[14].value
                item.rarity = row[15].value
                item.price = row[16].value
                item.lvl = row[17].value
                item.chance_to_physic = row[18].value
                item.coef_for_combo = row[19].value
                item.count_for_combo = row[20].value
                item.info = row[21].value
                session.add(item)
        wb = load_workbook(drop)
        ws = wb.active
        for i, row in enumerate(ws.iter_rows()):
            if i == 0:
                continue
            if not session.query(Drop).filter_by(monster=row[0].value, model =
row[1].value).first():
                drop = Drop(monster=row[0].value)
                drop.model = row[1].value
                drop.chance_for_drop = row[2].value
                session.add(drop)
        wb = load_workbook(resources)
        ws = wb.active
        for i, row in enumerate(ws.iter_rows()):
            if i == 0:
                continue
            if not session.query(Resource).filter_by(name=row[0].value).first():
                res = Resource(name=row[0].value)
                res.type = row[1].value
                res.price = row[2].value
                res.effect = row[3].value
```

```
            res.info = row[4].value
            res.uuid = shortuuid.uuid()
            res.uuid = 'r' + res.uuid[1:]
            session.add(res)
        session.commit()
```

B.9 Файл npc.py

```
# -*- coding: utf-8 -*-

import random

import sqlalchemy


from config import Base, session, bot


class NPC(Base):
    __tablename__ = 'npc'

    name = sqlalchemy.Column(sqlalchemy.String(length=50), primary_key=True)
    chance = sqlalchemy.Column(sqlalchemy.Integer)
    fraction = sqlalchemy.Column(sqlalchemy.String(length=12))
    lvl = sqlalchemy.Column(sqlalchemy.Integer)
    max_hp = sqlalchemy.Column(sqlalchemy.Float, default=300)
    damage_min_physic = sqlalchemy.Column(sqlalchemy.Float, default=30)
    damage_max_physic = sqlalchemy.Column(sqlalchemy.Float, default=40)
    damage_min_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
```

```
        damage_max_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
        defence_physic = sqlalchemy.Column(sqlalchemy.Float, default=1)
        defence_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
        penetration_physic = sqlalchemy.Column(sqlalchemy.Float, default=0)
        penetration_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
        dodge_magic = sqlalchemy.Column(sqlalchemy.Float, default=5)
        dodge_physic = sqlalchemy.Column(sqlalchemy.Float, default=5)
        crit_magic = sqlalchemy.Column(sqlalchemy.Float, default=5)
        crit_physic = sqlalchemy.Column(sqlalchemy.Float, default=5)
        chance_to_physic = sqlalchemy.Column(sqlalchemy.Float, default=100)
        gold_min = sqlalchemy.Column(sqlalchemy.Integer, default=0)
        gold_max = sqlalchemy.Column(sqlalchemy.Integer, default=10)
        experience = sqlalchemy.Column(sqlalchemy.Integer, default=50)
        location = sqlalchemy.Column(sqlalchemy.String(length=12))

        def __repr__(self):
            return u"<NPC(name='{0} lvl={1} exp={2}')>".format(self.name,
self.lvl, self.experience)


    class Fraction(Base):
        __tablename__ = 'frac'

        name = sqlalchemy.Column(sqlalchemy.String(length=12),
primary_key=True)


    async def mob_found(pers):
        mobs = session.query(NPC).filter_by(location=pers.location)
        range_for_random = []
```

```
            temp_range = 0
            for m in mobs:
                temp_range += m.chance
                range_for_random.append(temp_range)
            rand = random.uniform(0, 100)
            iterate_mob = 0
            for i, t_r in enumerate(range_for_random):
                if rand <= t_r:
                    iterate_mob = i
                    break
            for i, m in enumerate(mobs):
                if i == iterate_mob:
                    pers.monster_for_fight = m.name
                    await bot.send_message(pers.chat_id, u"Перед вами стоит {}".format(m.name))
                    return

            await bot.send_message(pers.chat_id, u"А в ответ тишина")
            return
```

B.10 Файл location.py

```
# -*- coding: utf-8 -*-

import sqlalchemy

from config import Base, session, bot
from keyboards import keyboard_main_town, keyboard_forest, keyboard_arena
```

```
class Location(Base):
    __tablename__ = 'location'

    name = sqlalchemy.Column(sqlalchemy.String(length=12),
primary_key=True)


class Time(Base):
    __tablename__ = 'time'

    id_ = sqlalchemy.Column(sqlalchemy.Integer, primary_key=True)
    from_ = sqlalchemy.Column(sqlalchemy.String(length=12))
    to_ = sqlalchemy.Column(sqlalchemy.String(length=12))
    time = sqlalchemy.Column(sqlalchemy.Integer, default=15)


async def wait(pers, location_name=None, chat_id=None):
    pers.is_in_action = False
    pers.location = location_name
    session.add(pers)
    session.commit()
    keyboard = keyboard_main_town
    if location_name == u'Город':
        keyboard = keyboard_main_town
    elif location_name == u'Лес':
        keyboard = keyboard_forest
    elif location_name == u'Арена':
        keyboard = keyboard_arena
```

```
        await bot.send_message(chat_id, u"Ты пришел в локацию '{}'".format(location_name), reply_markup=keyboard)
```

B.11 Файл items.py

```
# -*- coding: utf-8 -*-

import random
from uuid import uuid4

import os
import sqlalchemy
from sqlalchemy import ForeignKey
import shortuuid
from choices import rarity, for_info_type, class_eng_ru, attr_emotion
from config import Base, session, bot, APP_ROOT



class Resource(Base):
    __tablename__ = 'resource'

    uuid = sqlalchemy.Column(sqlalchemy.String(length=40), primary_key=True)
    name = sqlalchemy.Column(sqlalchemy.String(length=50))
    type = sqlalchemy.Column(sqlalchemy.String(length=12))
    price = sqlalchemy.Column(sqlalchemy.Integer)
    effect = sqlalchemy.Column(sqlalchemy.String(length=20))
    info = sqlalchemy.Column(sqlalchemy.String(length=1000))

    def get_info(self):
```

```
            result = u'//Инфо:\n'
            result += u'[{0}]\n'.format(self.name)
            result += u'Класс: {0}\n'.format(class_eng_ru[self.type])
            if self.info:
                result += self.info
            image = image = os.path.join(APP_ROOT,
u'static/other/{0}/{1}.PNG'.format(self.type, self.name))
            print(image)
            return result, image

        def __unicode__(self):
            return u"[{0}]".format(self.name)


    class Person_Resource(Base):
        __tablename__ = 'person_resource'

        _id = sqlalchemy.Column(sqlalchemy.Integer(), primary_key=True)
        uuid = sqlalchemy.Column(sqlalchemy.String(length=40))
        user_id = sqlalchemy.Column(sqlalchemy.String(length=12))
        count = sqlalchemy.Column(sqlalchemy.Integer, default=1)


    async def use_item(person, res):
        resource = session.query(Resource).filter_by(uuid=res).first()
        res_pers = session.query(Person_Resource).filter_by(uuid=res,
user_id=person.user_id).first()
        # if not res_pers:
        #     bot.send_message(person.chat_id, u"Ох уж эти читеры")
        #     return
```

```
            resource = resource.effect.split("#")
            print(resource)
            action = resource[1]
            count = resource[2]
            attr = resource[3]
            max_attr = None
            set_attr = getattr(person, attr)
            if attr == "hp":
                max_attr = "max_hp"
            if max_attr:
                if action == "+":
                    set_attr = getattr(person, attr) + int(count)
                    if set_attr > getattr(person, max_attr):
                        set_attr = getattr(person, max_attr)
            setattr(person, attr, set_attr)
            await bot.send_message(person.chat_id,
                                   getattr(person, attr) + (u"/" + getattr(person, max_attr)
                                                            + attr_emotion[attr]) if max_attr else
attr_emotion[attr])
            if res_pers.count == 1:
                session.delete(res_pers)
            else:
                print(res_pers.count)
                res_pers.count -= 1
            session.add(person)
            session.commit()


        def add_res(person_id, res, count=1):
```

```
        pers_res = session.query(Person_Resource).filter_by(user_id=person_id, uuid=res).first()
        if pers_res:
            pers_res.count += count
        else:
            pers_res = Person_Resource(uuid=res, user_id=person_id, count=count)
        session.add(pers_res)
        session.commit()


    class PersonalItem(Base):
        __tablename__ = 'personal_item'

        uuid = sqlalchemy.Column(sqlalchemy.String(length=40), primary_key=True)
        name = sqlalchemy.Column(sqlalchemy.String(length=50))
        type = sqlalchemy.Column(sqlalchemy.String(length=12))
        clas_s = sqlalchemy.Column(sqlalchemy.String(length=50))
        price = sqlalchemy.Column(sqlalchemy.Integer, default=10)
        person = sqlalchemy.Column(sqlalchemy.String(length=12))
        damage_physic_min = sqlalchemy.Column(sqlalchemy.Integer)
        damage_physic_max = sqlalchemy.Column(sqlalchemy.Integer)
        damage_magic_min = sqlalchemy.Column(sqlalchemy.Integer)
        damage_magic_max = sqlalchemy.Column(sqlalchemy.Integer)
        defence_physic = sqlalchemy.Column(sqlalchemy.Integer)
        defence_magic = sqlalchemy.Column(sqlalchemy.Integer)
        penetration_physic = sqlalchemy.Column(sqlalchemy.Float)
        penetration_magic = sqlalchemy.Column(sqlalchemy.Float)
        dodge_magic = sqlalchemy.Column(sqlalchemy.Float)
        dodge_physic = sqlalchemy.Column(sqlalchemy.Float)
```

```
        crit_magic = sqlalchemy.Column(sqlalchemy.Float)
        crit_physic = sqlalchemy.Column(sqlalchemy.Float)
        chance_to_physic = sqlalchemy.Column(sqlalchemy.Float, default=100)
        count_for_combo = sqlalchemy.Column(sqlalchemy.Integer)
        coef_for_combo = sqlalchemy.Column(sqlalchemy.Integer)
        rarity = sqlalchemy.Column(sqlalchemy.String(length=12))
        lvl = sqlalchemy.Column(sqlalchemy.Integer)
        info = sqlalchemy.Column(sqlalchemy.String(length=1000))
        image = sqlalchemy.Column(sqlalchemy.String(length=20))
        weared = sqlalchemy.Column(sqlalchemy.Boolean, default=False)

        def __unicode__(self):
            result = u'[{0}|{1}|'.format(rarity[self.rarity], self.name)
            with_previous = False
            if self.damage_magic_max:
                result += u'◒{0}-{1}'.format(self.damage_magic_min,
self.damage_magic_max)
                with_previous = True
            if self.damage_physic_max:
                if with_previous:
                    result += u'/'
                result += u'□{0}-{1}'.format(self.damage_physic_min,
self.damage_physic_max)
                with_previous = True
            if self.defence_physic:
                if with_previous:
                    result += u'/'
                result += u'✶□♡{0}'.format(self.defence_physic)
                with_previous = True
            if self.defence_magic:
```

```
        if with_previous:
            result += u'/'
        result += u'Ⓜ□♡{0}'.format(self.defence_magic)
        with_previous = True
    if self.penetration_physic:
        if with_previous:
            result += u'/'
        result += u'✶□💕{0}'.format(self.penetration_physic)
        with_previous = True
    if self.penetration_magic:
        if with_previous:
            result += u'/'
        result += u'Ⓜ□💕{0}'.format(self.penetration_magic)
        with_previous = True
    if self.dodge_physic:
        if with_previous:
            result += u'/'
        result += u'✶□🌬{0}'.format(self.dodge_physic)
        with_previous = True
    if self.dodge_magic:
        if with_previous:
            result += u'/'
        result += u'Ⓜ□🌬{0}'.format(self.dodge_magic)
        with_previous = True
    if self.crit_physic:
        if with_previous:
            result += u'/'
        result += u'✶□🎯{0}'.format(self.crit_physic)
        with_previous = True
```

```
            if self.crit_magic:
                if with_previous:
                    result += u'/'
                result += u'Ⓜ□🎯{0}'.format(self.crit_magic)
            result += u'|{0}]'.format(rarity[self.rarity])
            return result

        def get_info(self):
            result = u'//Инфо:\n'
            result += u'[{0}|{1}|ур.{2}|{0}] {3}\n'.format(rarity[self.rarity],
self.name, self.lvl,
                                                for_info_type[self.type])
            result += u'Класс: {0}\n'.format(class_eng_ru[self.clas_s])
            if self.damage_magic_max:
                result += u'Маг. урон: {0}-{1}\n'.format(self.damage_magic_min,
self.damage_magic_max)
            if self.damage_physic_max:
                result += u'Физ. урон: {0}-{1}\n'.format(self.damage_physic_min,
self.damage_physic_max)
            if self.defence_physic:
                result += u'Физ. сопротивление: {0}\n'.format(self.defence_physic)
            if self.defence_magic:
                result += u'Маг. сопротивление: {0}\n'.format(self.defence_magic)
            if self.penetration_physic:
                result += u'Физ. проникновение:
{0}\n'.format(self.penetration_physic)
            if self.penetration_magic:
                result += u'Маг. проникновение:
{0}\n'.format(self.penetration_magic)
            if self.dodge_physic:
```

```
            result += u'Физ. защита: {0}\n'.format(self.dodge_physic)
        if self.dodge_magic:
            result += u'Маг. защита: {0}\n'.format(self.dodge_magic)
        if self.crit_physic:
            result += u'Шанс физ. крита: {0}\n'.format(self.crit_physic)
        if self.crit_magic:
            result += u'Шанс маг. крита: {0}\n'.format(self.crit_magic)
        if self.info:
            result += self.info
        image = image = os.path.join(APP_ROOT,
u'static/{0}/{1}/{2}.PNG'.format(self.type, self.clas_s, self.name))
        print(image)
        return result, image


class DefaultItem(Base):
    __tablename__ = 'default_item'

    _id = sqlalchemy.Column(sqlalchemy.Integer, primary_key=True)
    name = sqlalchemy.Column(sqlalchemy.String(length=50))
    clas_s = sqlalchemy.Column(sqlalchemy.String(length=50))
    type = sqlalchemy.Column(sqlalchemy.String(length=12))
    price = sqlalchemy.Column(sqlalchemy.Integer, default=10)
    damage_physic_min = sqlalchemy.Column(sqlalchemy.Integer, default=0)
    damage_physic_max = sqlalchemy.Column(sqlalchemy.Integer, default=0)
    damage_magic_min = sqlalchemy.Column(sqlalchemy.Integer, default=0)
    damage_magic_max = sqlalchemy.Column(sqlalchemy.Integer, default=0)
    defence_physic_min = sqlalchemy.Column(sqlalchemy.Integer, default=0)
    defence_physic_max = sqlalchemy.Column(sqlalchemy.Integer, default=0)
    defence_magic_min = sqlalchemy.Column(sqlalchemy.Integer, default=0)
```

```
    defence_magic_max = sqlalchemy.Column(sqlalchemy.Integer, default=0)
    penetration_physic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    penetration_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    dodge_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    dodge_physic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    crit_magic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    crit_physic = sqlalchemy.Column(sqlalchemy.Float, default=0)
    chance_to_physic = sqlalchemy.Column(sqlalchemy.Float, default=100)
    count_for_combo = sqlalchemy.Column(sqlalchemy.Integer)
    coef_for_combo = sqlalchemy.Column(sqlalchemy.Integer)
    rarity = sqlalchemy.Column(sqlalchemy.String(length=12))
    lvl = sqlalchemy.Column(sqlalchemy.Integer, default=1)
    info = sqlalchemy.Column(sqlalchemy.String(length=1000))
    image = sqlalchemy.Column(sqlalchemy.String(length=20))


class Drop(Base):
    __tablename__ = 'drop'

    _id = sqlalchemy.Column(sqlalchemy.Integer, primary_key=True)
    model = sqlalchemy.Column(sqlalchemy.String(length=50))
    monster = sqlalchemy.Column(sqlalchemy.String(length=50))
    chance_for_drop = sqlalchemy.Column(sqlalchemy.Float)


def drop_from_mob(monster):
    pers_item = None
    if random.uniform(0, 100) <= 100:
        drops = session.query(Drop).filter_by(monster=monster.name)
        range_for_random = []
```

```
            temp_range = 0
            for d in drops:
                temp_range += d.chance_for_drop
                range_for_random.append(temp_range)
            rand = random.uniform(0, 100)
            iterate_drop = 0
            for i, t_r in enumerate(range_for_random):
                if rand <= t_r:
                    iterate_drop = i
                    break
            for i, drop in enumerate(drops):
                if i == iterate_drop:
                    model = session.query(DefaultItem).filter_by(name=drop.model).first()
                    pers_item = generate_personal_item(model)
        return pers_item


    def generate_personal_item(model):
        pers_item = PersonalItem(name=model.name, price=model.price, type=model.type,

damage_physic_min=random.randint(model.damage_physic_min,
                                 (model.damage_physic_min + model.damage_physic_max) / 2),

damage_physic_max=random.randint((model.damage_physic_min + model.damage_physic_max) / 2,
                                 model.damage_physic_max),
```

```
damage_magic_min=random.randint(model.damage_magic_min,
                                (model.damage_magic_min +
model.damage_magic_max) / 2),

damage_magic_max=random.randint((model.damage_magic_min +
model.damage_magic_max) / 2,
                                model.damage_magic_max),
                    defence_physic=random.randint(model.defence_physic_min,
model.defence_physic_max),
                    defence_magic=random.randint(model.defence_magic_min,
model.defence_magic_max),
                    dodge_magic=model.dodge_magic,
                    dodge_physic=model.dodge_physic,
                    crit_magic=model.crit_magic,
                    crit_physic=model.crit_physic,
                    chance_to_physic=model.chance_to_physic,
                    lvl=model.lvl,
                    rarity=model.rarity,
                    clas_s=model.clas_s,
                    info=model.info,
                    image=model.image,
                    count_for_combo=model.count_for_combo,
                    coef_for_combo=model.coef_for_combo,
                    uuid=get_default_as_uuid())
        return pers_item


    async def show_backpack(pers):
        backpack = u"Инвентарь 🎒:\n"
```

```
            item = session.query(PersonalItem).filter_by(uuid=pers.weapon).first()
            backpack += u"Оружие: " + ((item.__unicode__() + u' /unwear_' +
item.uuid + u'\n' + u'/info_' + item.uuid + u'\n')
                                       if pers.weapon else u'пусто\n')
            item = session.query(PersonalItem).filter_by(uuid=pers.head).first()
            backpack += u"Голова: " + ((item.__unicode__() + u' /unwear_' + item.uuid
+ u'\n' + u'/info_' + item.uuid + u'\n')
                                       if pers.head else u'пусто\n')
            item = session.query(PersonalItem).filter_by(uuid=pers.body).first()
            backpack += u"Торс: " + ((item.__unicode__() + u' /unwear_' + item.uuid +
u'\n' + u'/info_' + item.uuid + u'\n')
                                     if pers.body else u'пусто\n')
            item = session.query(PersonalItem).filter_by(uuid=pers.hands).first()
            backpack += u"Руки: " + ((item.__unicode__() + u' /unwear_' + item.uuid +
u'\n' + u'/info_' + item.uuid + u'\n')
                                     if pers.hands else u'пусто\n')
            item = session.query(PersonalItem).filter_by(uuid=pers.legs).first()
            backpack += u"Ноги: " + ((item.__unicode__() + u' /unwear_' + item.uuid +
u'\n' + u'/info_' + item.uuid + u'\n')
                                     if pers.legs else u'пусто\n')
            item = session.query(PersonalItem).filter_by(uuid=pers.feet).first()
            backpack += u"Ступни: " + ((item.__unicode__() + u' /unwear_' + item.uuid
+ u'\n' + u'/info_' + item.uuid + u'\n')
                                       if pers.feet else u'пусто\n')
            item = session.query(PersonalItem).filter_by(uuid=pers.neck).first()
            backpack += u"Шея: " + ((item.__unicode__() + u' /unwear_' + item.uuid +
u'\n' + u'/info_' + item.uuid + u'\n')
                                    if pers.neck else u'пусто\n')
            item = session.query(PersonalItem).filter_by(uuid=pers.finger_1).first()
```

```
            backpack += u"Палец: " + ((item.__unicode__() + u' /unwear_' + item.uuid
+ u'\n' + u'/info_' + item.uuid + u'\n')
                                      if pers.finger_1 else u'пусто\n')
            item = session.query(PersonalItem).filter_by(uuid=pers.finger_2).first()
            backpack += u"Палец: " + ((item.__unicode__() + u' /unwear_' + item.uuid
+ u'\n' + u'/info_' + item.uuid + u'\n')
                                      if pers.finger_2 else u'пусто\n')
            items = session.query(PersonalItem).filter_by(person=pers.user_id)
            if items.count():
                backpack += u"\n Рюкзак:({}/9999999)🎒\n".format(items.count())
                for item in items:
                    if not item.weared:
                        backpack += item.__unicode__() + u'/wear_' + item.uuid + u'\n' \
                                    + u'/info_' + item.uuid + u'\n'
            resources = session.query(Person_Resource).filter_by(user_id=pers.user_id)
            if resources.count():
                backpack                                    +=                                    u"\n
Подсумок:({}/9999999)🎒\n".format(resources.count())
                for res in resources:
                    res_model = session.query(Resource).filter_by(uuid=res.uuid).first()
                    backpack += res_model.__unicode__() + ((u"({0})".format(res.count))
if res.count > 1 else u"") + (
                            (
                                u' /use_{0}\n'.format(
                                    res.uuid)) if res_model.type == u"potion" else u"") + u'/info_' +
res.uuid + u'\n'
            await bot.send_message(pers.chat_id, backpack)


        def get_default_as_uuid():
```

```
        uuid = "i" + shortuuid.uuid()[1:]
        return uuid


    async def wear(pers, item):
        item = session.query(PersonalItem).filter_by(uuid=item).first()
        if item.type == u'weapon':
            if pers.weapon:
                item_to_unwear =
session.query(PersonalItem).filter_by(uuid=pers.weapon).first()
                if item_to_unwear:
                    unwear_item(pers, item_to_unwear)
            pers.weapon = item.uuid
            wear_item(pers, item)
        elif item.type == u'body':
            if pers.body:
                item_to_unwear =
session.query(PersonalItem).filter_by(uuid=pers.body).first()
                if item_to_unwear:
                    unwear_item(pers, item_to_unwear)
            pers.body = item.uuid
            wear_item(pers, item)
        elif item.type == u'hands':
            if pers.hands:
                item_to_unwear =
session.query(PersonalItem).filter_by(uuid=pers.hands).first()
                if item_to_unwear:
                    unwear_item(pers, item_to_unwear)
            pers.hands = item.uuid
            wear_item(pers, item)
```

```
            elif item.type == u'head':
                if pers.head:
                    item_to_unwear                                          =
session.query(PersonalItem).filter_by(uuid=pers.head).first()
                    if item_to_unwear:
                        unwear_item(pers, item_to_unwear)
                pers.head = item.uuid
                wear_item(pers, item)
            elif item.type == u'legs':
                if pers.legs:
                    item_to_unwear                                          =
session.query(PersonalItem).filter_by(uuid=pers.legs).first()
                    if item_to_unwear:
                        unwear_item(pers, item_to_unwear)
                pers.legs = item.uuid
                wear_item(pers, item)
            elif item.type == u'feet':
                if pers.feet:
                    item_to_unwear                                          =
session.query(PersonalItem).filter_by(uuid=pers.feet).first()
                    if item_to_unwear:
                        unwear_item(pers, item_to_unwear)
                pers.feet = item.uuid
                wear_item(pers, item)
            elif item.type == u'neck':
                if pers.neck:
                    item_to_unwear                                          =
session.query(PersonalItem).filter_by(uuid=pers.neck).first()
                    if item_to_unwear:
                        unwear_item(pers, item_to_unwear)
```

```
            pers.neck = item.uuid
            wear_item(pers, item)
        elif item.type == u'finger':
            if pers.finger_1 and pers.finger_2:
                item_to_unwear =
session.query(PersonalItem).filter_by(uuid=pers.finger_1).first()
                if item_to_unwear:
                    unwear_item(pers, item_to_unwear)
                pers.finger_1 = item.uuid
                wear_item(pers, item)
            elif pers.finger_1:
                pers.finger_2 = item.uuid
                wear_item(pers, item)
            else:
                pers.finger_1 = item.uuid
                wear_item(pers, item)
        await show_backpack(pers)
        session.add(pers)
        session.commit()
        return


    def unwear(pers, item):
        item = session.query(PersonalItem).filter_by(uuid=item).first()
        if item.type == u'weapon':
            if pers.weapon and pers.weapon == item.uuid:
                unwear_item(pers, item)
            pers.weapon = None
        elif item.type == u'body':
            if pers.body and pers.body == item.uuid:
```

```
            unwear_item(pers, item)
        pers.body = None
    elif item.type == u'hands':
        if pers.hands and pers.hands == item.uuid:
            unwear_item(pers, item)
        pers.hands = None
    elif item.type == u'head':
        if pers.head and pers.head == item.uuid:
            unwear_item(pers, item)
        pers.head = None
    elif item.type == u'legs':
        if pers.legs and pers.legs == item.uuid:
            unwear_item(pers, item)
        pers.legs = None
    elif item.type == u'feet':
        if pers.feet and pers.feet == item.uuid:
            unwear_item(pers, item)
        pers.feet = None
    elif item.type == u'neck':
        if pers.neck and pers.neck == item.uuid:
            unwear_item(pers, item)
        pers.neck = None
    elif item.type == u'finger':
        if pers.finger_1 and pers.finger_1 == item.uuid:
            unwear_item(pers, item)
            pers.finger_1 = None
        elif pers.finger_2 and pers.finger_2 == item.uuid:
            unwear_item(pers, item)
            pers.finger_1 = None
    show_backpack(pers)
```

```
    session.add(pers)
    session.commit()
    return


def unwear_item(pers, item_to_unwear):
    pers.damage_min_physic -= item_to_unwear.damage_physic_min
    pers.damage_max_physic -= item_to_unwear.damage_physic_max
    pers.damage_min_magic -= item_to_unwear.damage_magic_min
    pers.damage_max_magic -= item_to_unwear.damage_magic_max
    pers.defence_physic -= item_to_unwear.defence_physic
    pers.defence_magic -= item_to_unwear.defence_magic
    pers.dodge_physic -= item_to_unwear.dodge_physic
    pers.dodge_magic -= item_to_unwear.dodge_magic
    pers.crit_physic -= item_to_unwear.crit_physic
    pers.crit_magic -= item_to_unwear.crit_magic
    if item_to_unwear.type == u'weapon':
        pers.count_for_combo = 5
        pers.chance_to_physic = 100
        pers.coef_for_combo = 2
    item_to_unwear.weared = False
    session.add(item_to_unwear)
    session.add(pers)
    session.commit()


def wear_item(pers, item_to_wear):
    pers.damage_min_physic += item_to_wear.damage_physic_min
    pers.damage_max_physic += item_to_wear.damage_physic_max
    pers.damage_min_magic += item_to_wear.damage_magic_min
```

```
    pers.damage_max_magic += item_to_wear.damage_magic_max
    pers.defence_physic += item_to_wear.defence_physic
    pers.defence_magic += item_to_wear.defence_magic
    pers.dodge_physic += item_to_wear.dodge_physic
    pers.dodge_magic += item_to_wear.dodge_magic
    pers.crit_physic += item_to_wear.crit_physic
    pers.crit_magic += item_to_wear.crit_magic
    if item_to_wear.type == u'weapon':
        pers.count_for_combo = item_to_wear.count_for_combo
        pers.chance_to_physic = item_to_wear.chance_to_physic
        pers.coef_for_combo = item_to_wear.coef_for_combo
    item_to_wear.weared = True
    session.add(item_to_wear)
    session.add(pers)
    session.commit()
```

B.12 Файл generateQuest.py

```
import random

import shortuuid

from config import session
from models import Items
from models.location import Location
from models.npc import NPC
from models.quest import Quest



def getRandomTarget(lvl):
```

```
    targets = session.query(Items).filte_by(lvl)
    return targets[random.randint(0, len(targets))]


def getRandomEnemy(lvl):
    enemies = session.query(NPC).filte_by(lvl)
    return enemies[random.randint(0, len(enemies))]


def getRandomType():
    return ['kill', 'support', 'find', 'delivery'][random.randint(0, 5)]


def getRandomLocation(lvl):
    locations = session.query(Location).filter_by(lvl)
    return locations[random.randint(0, len(locations))]


def generateQuestByLocation(lvl, location, npc_quest):
    quest = Quest(id=shortuuid.uuid())
    quest.type = getRandomType()
    quest.location = location
    quest.npc = npc_quest
    if quest.type in ['find', 'delivery']:
        quest.target = getRandomTarget(lvl)
    elif quest.type in ['support', 'delivery']:
        quest.location_to = getRandomLocation(lvl)
    elif quest.type == 'kill':
        quest.enemy = getRandomEnemy(lvl)
    session.add(quest)
```

```
    session.commit()
    return quest


def generateHeader(player, tempNode):
    location = getRandomLocation(player.lvl)
    npc_list = session.query(NPC).filter_by(location)
    npc_quest = random.randint(0, len(npc_list))
    return generateQuestByLocation(player.lvl, location, npc_quest)
```

B.13 Файл generateItem.py

```
import random

from config import session
from curve import getMinMax
from models.Items import DefaultItem

# 0 - 70
def generateItem(name, type_, clas_s, lvl, koef_rosp_list, koef_divide_list):
    n = 6
    characteristic = list()
    Min, Max = getMinMax(lvl)
    for kr, i in enumerate(koef_rosp_list):
        x_max = Max * kr / n
        x_min = Min * kr / n
        characteristic.append(random.randint(x_min, x_max) / koef_divide_list[i])

    item = DefaultItem(name=name)
```

```
    item.type = type_
    item.clas_s = clas_s
    item.damage_physic_min = characteristic[0] * 0.95
    item.damage_physic_max = characteristic[0] * 1.05
    item.damage_magic_min = characteristic[1] * 0.95
    item.damage_magic_max = characteristic[1] * 1.05
    item.defence_physic_min = characteristic[2] * 0.95
    item.defence_physic_max = characteristic[2] * 1.05
    item.defence_magic_min = characteristic[3] * 0.95
    item.defence_magic_max = characteristic[3] * 1.05
    item.dodge_physic = characteristic[4]
    item.dodge_magic = characteristic[5]
    item.crit_physic = characteristic[6]
    item.crit_magic = characteristic[7]
    item.rarity = characteristic[8]
    item.price = characteristic[9]
    item.lvl = lvl

    session.add(item)
    return item
```

В.14 Файл fight.py

```
# -*- coding: utf-8 -*-

import random

from config import session, bot, coefCrit
from models.Items import drop_from_mob, PersonalItem
from models.npc import NPC
```

```
from models.person import Person
from tasks import pers_death


def mob_attack(mob, pers):
    with_crit = False
    if random.uniform(0, 100) > mob.chance_to_physic:
        if random.uniform(0, 100) > pers.dodge_magic:
            basic_damage = random.randint(mob.damage_min_magic,
mob.damage_max_magic)
            if random.uniform(0, 100) <= mob.crit_magic:
                with_crit = True
                damage = (coefCrit * basic_damage) - pers.defence_magic +
mob.penetration_magic
            else:
                damage = basic_damage - pers.defence_magic +
mob.penetration_magic
            pers.hp -= damage
            result = u"{0} нанес ".format(mob.name)
            if with_crit:
                result += u"□"
            result += u" ◌{0}урона□, у {1} осталось {2}❤□\n".format(damage,
pers.name, pers.hp)
        else:
            result = u"{0} увернулся 🌬\n".format(pers.name)
    else:
        if random.uniform(0, 100) > pers.dodge_physic:
            basic_damage = random.randint(mob.damage_min_physic,
mob.damage_max_physic)
            if random.uniform(0, 100) <= mob.crit_physic:
```

```
                with_crit = True
                damage = (coefCrit * basic_damage) - pers.defence_physic + mob.penetration_physic
            else:
                damage = basic_damage - pers.defence_physic + mob.penetration_physic
            pers.hp -= damage
            result = u"{0} нанес ".format(mob.name)
            if with_crit:
                result += u"□"
            result += u" □{0}урона□, у {1} осталось {2}❤□\n".format(damage, pers.name, pers.hp)
        else:
            result = u"{0} увернулся 🌬\n".format(pers.name)
    return result


async def PvE(pers, monster):
    monster = session.query(NPC).filter_by(name=monster).first()
    monster_temp_hp = monster.max_hp
    pers.temp_count_for_combo = 0
    pers_die = False
    monster_die = False
    first_fighter = random.randint(1, 2)
    result_of_fight = u"Сражение с {0} \n".format(monster.name)
    count = 0
    chance_to_physic = get_chance_to_physic(pers)
    while not pers_die and not monster_die:
        if first_fighter == 1:
```

```
                add_result_of_fight, monster_temp_hp = attack_mob(pers, monster, monster_temp_hp, chance_to_physic)
                result_of_fight += add_result_of_fight
                if monster_temp_hp <= 0:
                    monster_die = True
                else:
                    result_of_fight += mob_attack(monster, pers)
                    if pers.hp <= 0:
                        pers_die = True
            else:
                result_of_fight += mob_attack(monster, pers)
                if pers.hp <= 0:
                    pers_die = True
                else:
                    add_result_of_fight, monster_temp_hp = attack_mob(pers, monster, monster_temp_hp, chance_to_physic)
                    result_of_fight += add_result_of_fight
                    if monster_temp_hp <= 0:
                        monster_die = True
            count += 1


        if pers_die:
            result_of_fight += u"Поражение, у монстра осталось {0}❤□, бой длился {1} тиков".format(monster_temp_hp, count)
            await bot.send_message(pers.chat_id, result_of_fight)
            await pers_death(pers.user_id)
        else:
            pers_item = drop_from_mob(monster)
            pers.add_experience(monster.experience)
            if pers_item:
```

```
                pers_item.person = pers.user_id
                session.add(pers_item)
                result_of_fight += u"Игрок {0} победил с {1}, бой длился {2} тиков, вы получили {3}".format(pers.name,
                                                                                                            pers.hp,
count,

pers_item.__unicode__())
            else:
                result_of_fight += u"Игрок {0} победил с {1}, бой длился {2} тиков".format(pers.name, pers.hp, count)
            await bot.send_message(pers.chat_id, result_of_fight)

        pers.temp_count_for_combo = 0
        session.add(pers)
        session.commit()



    def attack_mob(pers, mob, mob_hp, chance_to_physic):
        with_combo = False
        with_crit = False
        if pers.weapon and random.uniform(0, 100) > chance_to_physic:
            if random.uniform(0, 100) > mob.dodge_magic - pers.accuracy_magic:
                basic_damage = random.randint(pers.damage_min_magic, pers.damage_max_magic)
                if pers.temp_count_for_combo == pers.count_for_combo:
                    with_combo = True
                    basic_damage *= pers.coef_for_combo
                    pers.temp_count_for_combo = 0
                else:
```

```
                pers.temp_count_for_combo += 1
            if random.uniform(0, 100) <= pers.crit_magic:
                with_crit = True
                damage = (coefCrit * basic_damage) - mob.defence_magic + pers.penetration_magic
            else:
                damage = basic_damage - mob.defence_magic + pers.penetration_magic
            mob_hp -= damage
            result = u"{0} нанес ".format(pers.name)
            if with_crit:
                result += u"💥"
            if with_combo:
                result += u"□"
            result += u" 🤕{0}урона□, у {1} осталось {2}❤□\n".format(damage, mob.name, mob_hp)
        else:
            if random.uniform(0, 100) > pers.temp_attack:
                pers.temp_count_for_combo = 0
            result = u"{0} увернулся 💨\n".format(mob.name)
    else:
        if random.uniform(0, 100) > mob.dodge_physic - pers.accuracy_physic:
            basic_damage = random.randint(pers.damage_min_physic, pers.damage_max_physic)
            if pers.temp_count_for_combo == pers.count_for_combo:
                with_combo = True
                basic_damage *= pers.coef_for_combo
                pers.temp_count_for_combo = 0
            else:
                pers.temp_count_for_combo += 1
```

```
                if random.uniform(0, 100) <= pers.crit_physic:
                    with_crit = True
                    damage = (coefCrit * basic_damage) - mob.defence_physic + pers.penetration_physic
                else:
                    damage = basic_damage - mob.defence_physic + pers.penetration_physic
                mob_hp -= damage
                result = u"{0} нанес ".format(pers.name)
                if with_crit:
                    result += u"💥"
                if with_combo:
                    result += u"□"
                result += u" □{0}урона□, у {1} осталось {2}❤□\n".format(damage, mob.name, mob_hp)
            else:
                if random.uniform(0, 100) > pers.temp_attack:
                    pers.temp_count_for_combo = 0
                result = u"{0} увернулся 💨\n".format(mob.name)
            return result, mob_hp


        async def PvP(pers_1, pers_2):
            pers_1.hp = pers_1.max_hp
            pers_2.hp = pers_2.max_hp
            pers_1.temp_count_for_combo = 0
            pers_2.temp_count_for_combo = 0
            pers_1_die = False
            pers_2_die = False
            first_fighter = random.randint(1, 2)
```

```
        result_of_fight = u"Сражение между игроками {0} {1}/{2}HP и {3} {4}/{5} HP\n".format(pers_1.name, pers_1.hp,
                                                                                              pers_1.max_hp,
pers_2.name,
                                                                                              pers_2.hp,
pers_2.max_hp)
        chance_to_physic_1 = get_chance_to_physic(pers_1)
        chance_to_physic_2 = get_chance_to_physic(pers_2)
        count = 0
        while not pers_1_die and not pers_2_die:
            if first_fighter == 1:
                result_of_fight += attack(pers_1, pers_2, chance_to_physic_1)
                if pers_2.hp <= 0:
                    pers_2_die = True
                else:
                    result_of_fight += attack(pers_2, pers_1, chance_to_physic_2)
                    if pers_1.hp <= 0:
                        pers_1_die = True
            else:
                result_of_fight += attack(pers_2, pers_1, chance_to_physic_2)
                if pers_1.hp <= 0:
                    pers_1_die = True
                else:
                    result_of_fight += attack(pers_1, pers_2, chance_to_physic_1)
                    if pers_2.hp <= 0:
                        pers_2_die = True
            count += 1

        if pers_1_die:
```

```
            result_of_fight += u"Игрок {0} победил с {1}❤, бой длился {2} тиков".format(pers_2.name, pers_2.hp, count)
        else:
            result_of_fight += u"Игрок {0} победил с {1}❤, бой длился {2} тиков".format(pers_1.name, pers_1.hp, count)
        if pers_1.chat_id:
            await bot.send_message(pers_1.chat_id, result_of_fight)
        if pers_2.chat_id:
            await bot.send_message(pers_2.chat_id, result_of_fight)
        pers_1.temp_count_for_combo = 0
        pers_2.temp_count_for_combo = 0
        session.add(pers_1)
        session.add(pers_2)
        session.commit()
        return


    def attack(atacker, defender, chance_to_physic):
        with_combo = False
        with_crit = False
        if atacker.weapon and random.uniform(0, 100) > chance_to_physic:
            if random.uniform(0, 100) > defender.dodge_magic - atacker.accuracy_magic:
                basic_damage = random.randint(atacker.damage_min_magic, atacker.damage_max_magic)
                if atacker.temp_count_for_combo == atacker.count_for_combo:
                    with_combo = True
                    basic_damage *= atacker.coef_for_combo
                    atacker.temp_count_for_combo = 0
                else:
```

```
                atacker.temp_count_for_combo += 1
            if random.uniform(0, 100) <= atacker.crit_magic:
                with_crit = True
                damage = (coefCrit * basic_damage) - defender.defence_magic + atacker.penetration_magic
            else:
                damage = basic_damage - defender.defence_magic + atacker.penetration_magic
            defender.hp-=damage
            result = u"{0} нанес ".format(atacker.name)
            if with_crit:
                result += u"💥"
            if with_combo:
                result += u"□"
            result += u" 🤕{0}урона□, у {1} осталось {2}❤□\n".format(damage, defender.name, defender.hp)
        else:
            if random.uniform(0, 100) > atacker.temp_attack:
                atacker.temp_count_for_combo = 0
            result = u"{0} увернулся 💨\n".format(defender.name)
    else:
        if random.uniform(0, 100) > defender.dodge_physic - atacker.accuracy_physic:
            basic_damage = random.randint(atacker.damage_min_physic, atacker.damage_max_physic)
            if atacker.temp_count_for_combo == atacker.count_for_combo:
                with_combo = True
                basic_damage *= atacker.coef_for_combo
                atacker.temp_count_for_combo = 0
            else:
```

```
                atacker.temp_count_for_combo += 1
            if random.uniform(0, 100) <= atacker.crit_physic:
                with_crit = True
                damage = (coefCrit * basic_damage) - defender.defence_physic + atacker.penetration_physic
            else:
                damage = basic_damage - defender.defence_physic + atacker.penetration_physic
            defender.hp -= damage
            result = u"{0} нанес ".format(atacker.name)
            if with_crit:
                result += u"✹"
            if with_combo:
                result += u"□"
            result += u" □{0}урона□, у {1} осталось {2}❤□\n".format(damage, defender.name, defender.hp)
        else:
            if random.uniform(0, 100) > atacker.temp_attack:
                atacker.temp_count_for_combo = 0
            result = u"{0} увернулся 🌬\n".format(defender.name)
        return result


    async def big_PvP(pers_1, pers_2, number):
        avarage_count = 0
        pers_1_wins = 0
        pers_2_wins = 0
        for i in range(0, number):
            count = 0
            pers_1.hp = pers_1.max_hp
```

```
pers_2.hp = pers_2.max_hp
pers_1.temp_count_for_combo = 0
pers_2.temp_count_for_combo = 0
pers_1_die = False
pers_2_die = False
first_fighter = random.randint(1, 2)
chance_to_physic_1 = get_chance_to_physic(pers_1)
chance_to_physic_2 = get_chance_to_physic(pers_2)
while not pers_1_die and not pers_2_die:
    if first_fighter == 1:
        attack(pers_1, pers_2, chance_to_physic_1)
        if pers_2.hp <= 0:
            pers_2_die = True
        else:
            attack(pers_2, pers_1, chance_to_physic_2)
            if pers_1.hp <= 0:
                pers_1_die = True
    else:
        attack(pers_2, pers_1, chance_to_physic_2)
        if pers_1.hp <= 0:
            pers_1_die = True
        else:
            attack(pers_1, pers_2, chance_to_physic_1)
            if pers_2.hp <= 0:
                pers_2_die = True
    count += 1
avarage_count += count
if pers_1_die:
    pers_2_wins += 1
else:
```

```
                pers_1_wins += 1
            avarage_count = avarage_count/number
            pers_1_wins = pers_1_wins * 100 / number
            pers_2_wins = pers_2_wins * 100 / number
            if pers_1.chat_id:
                await bot.send_message(pers_1.chat_id,
                                       u'Среднее количество тиков в бою:{0} \n Количество побед у первого игрока:{1}% \n'
                                       u'Количество побед у второго игрока:{2}%'.format(avarage_count, pers_1_wins,
                                                                                        pers_2_wins))
            pers_1.temp_count_for_combo = 0
            pers_2.temp_count_for_combo = 0
            session.add(pers_1)
            session.add(pers_2)
            session.commit()
            return


        def attack_for_bif_fight(atacker, defender, chance_to_physic):
            with_combo = False
            with_crit = False
            if atacker.weapon and random.uniform(0, 100) > chance_to_physic:
                if random.uniform(0, 100) > defender.dodge_magic - atacker.accuracy_magic:
                    basic_damage = random.randint(atacker.damage_min_magic, atacker.damage_max_magic)
                    if atacker.temp_count_for_combo == atacker.count_for_combo:
                        with_combo = True
                        basic_damage *= atacker.coef_for_combo
```

```
                atacker.temp_count_for_combo = 0
            else:
                atacker.temp_count_for_combo += 1
            if random.uniform(0, 100) <= atacker.crit_magic:
                with_crit = True
                damage = (coefCrit * basic_damage) - defender.defence_magic + atacker.penetration_magic
            else:
                damage = basic_damage - defender.defence_magic + atacker.penetration_magic
            defender.hp -= damage
        else:
            if random.uniform(0, 100) > atacker.temp_attack:
                atacker.temp_count_for_combo = 0
    else:
        if random.uniform(0, 100) > defender.dodge_physic - atacker.accuracy_physic:
            basic_damage = random.randint(atacker.damage_min_physic, atacker.damage_max_physic)
            if atacker.temp_count_for_combo == atacker.count_for_combo:
                with_combo = True
                basic_damage *= atacker.coef_for_combo
                atacker.temp_count_for_combo = 0
            else:
                atacker.temp_count_for_combo += 1
            if random.uniform(0, 100) <= atacker.crit_physic:
                with_crit = True
                damage = (coefCrit * basic_damage) - defender.defence_physic + atacker.penetration_physic
            else:
```

```
                damage = basic_damage - defender.defence_physic + atacker.penetration_physic
                defender.hp -= damage
            else:
                if random.uniform(0, 100) > atacker.temp_attack:
                    atacker.temp_count_for_combo = 0
            return 0


        def enemy_found(user_id, location, chat_id):
            persons = session.query(Person).filter_by(location=location)
            result_of_found = u'Доступные противники:\n'
            for p in persons:
                if p.user_id == user_id:
                    continue
                result_of_found += u'/fight_{}\n'.format(p.user_id)
            bot.send_message(chat_id, result_of_found)
            return


        def get_chance_to_physic(pers):
            chance_to_physic = session.query(PersonalItem).filter_by(uuid=pers.weapon).first()
            if chance_to_physic:
                return chance_to_physic.chance_to_physic
            else:
                return 100
```

ДОДАТОК Г

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ІГРОВИХ МЕХАНІЗМІВ НА ОСНОВІ TELEGRAM-БОТІВ

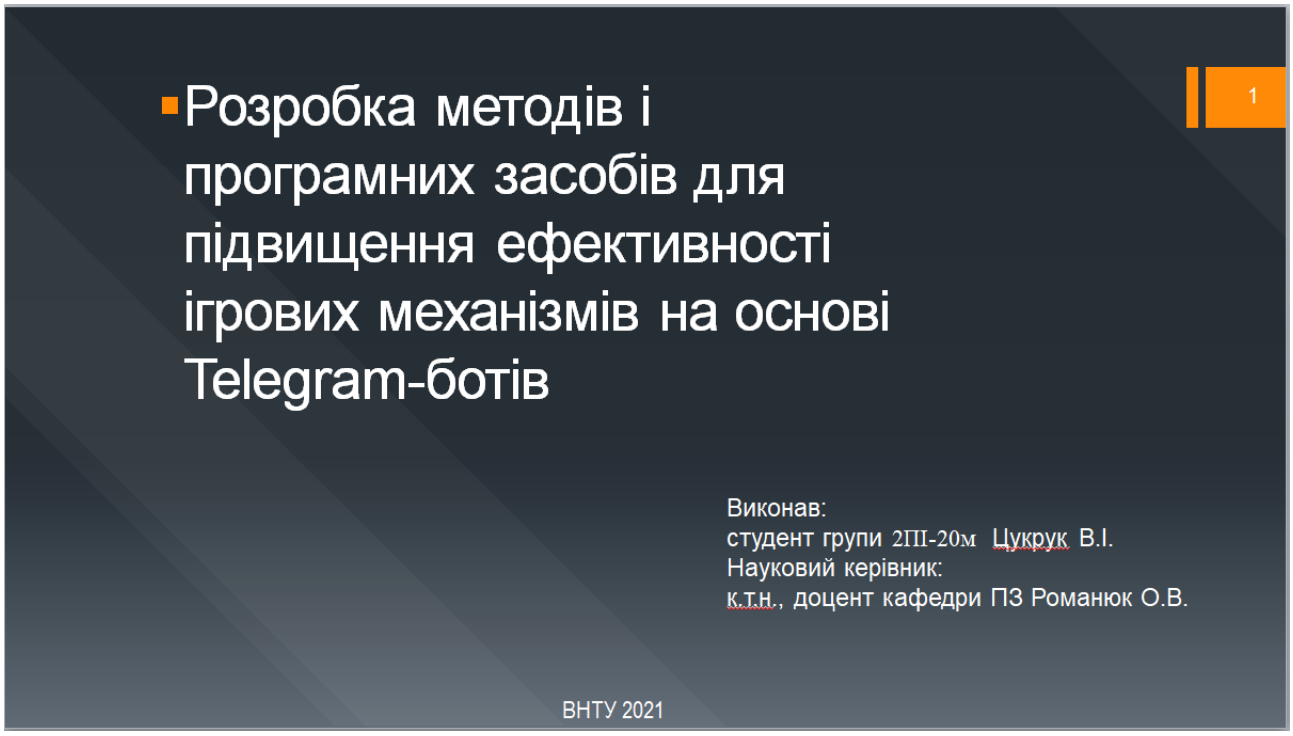


Рисунок Г.1 – Перший слайд

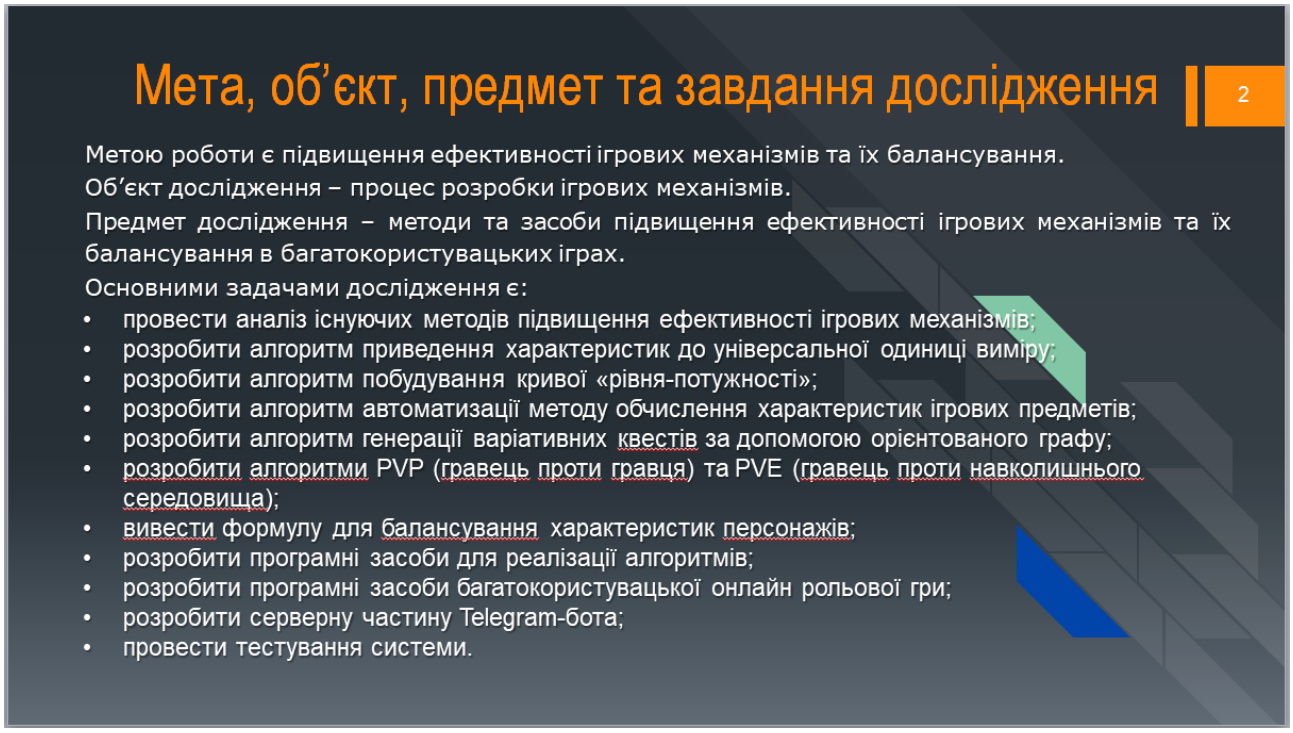


Рисунок Г.2 – Мета, об'єкт, предмет та завдання дослідження

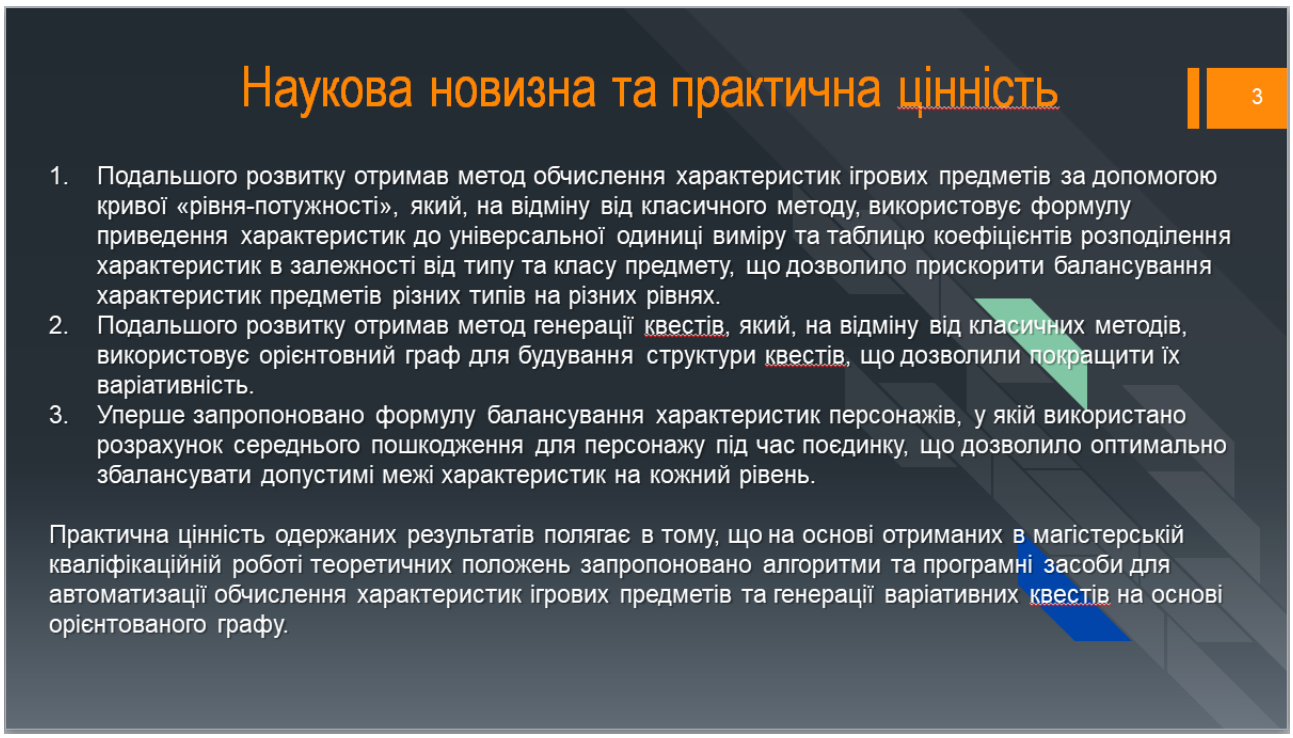


Рисунок Г.3 – Наукова новизна та практична цінність

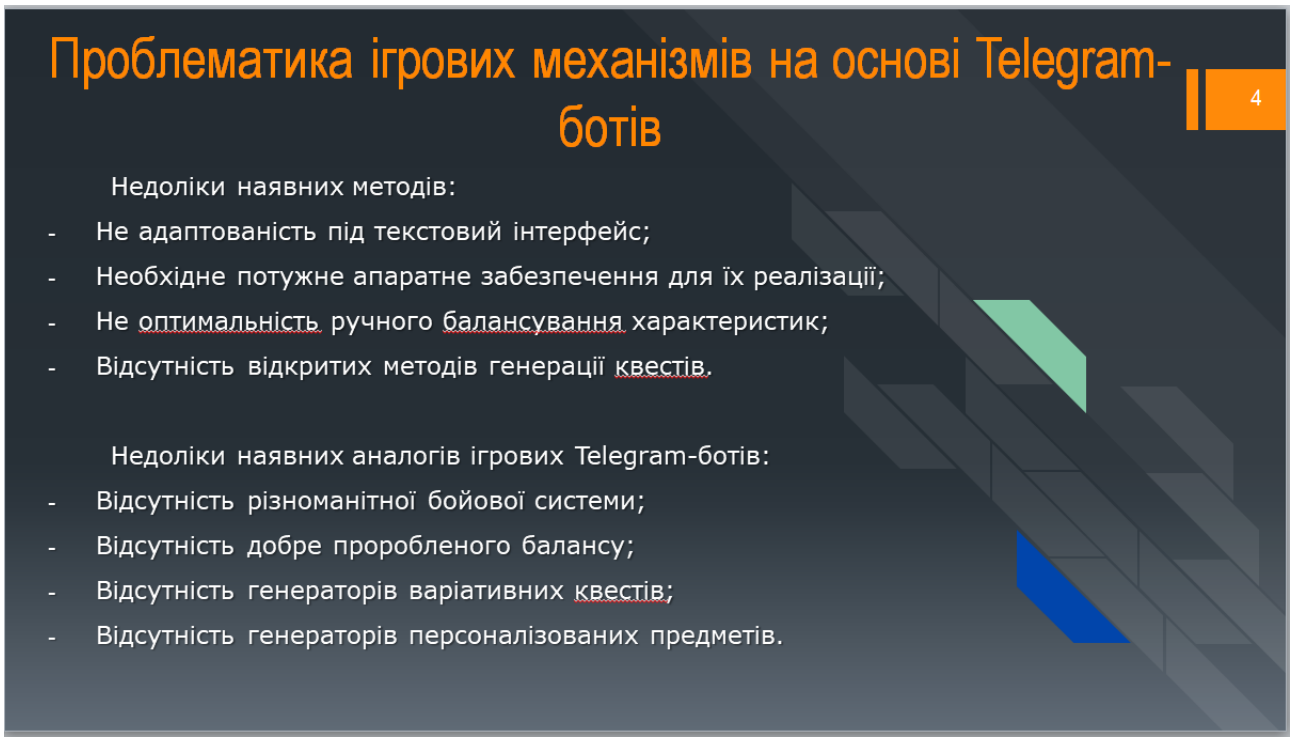


Рисунок Г.4 – Проблематика ігрових механізмів на основі Telegram-ботів

Результати аналізу методів реалізації ігрових механізмів

5

- Метод балансування характеристик – транзитивний (це метод прямого порівняння характеристик, транзитивний метод використовується у випадках, коли порівнювальні об'єкти мають однаковий, або схожий за сенсом, набір характеристик).
- Метод створення ігрових об'єктів – від характеристик (Спочатку описуються характеристики певного предмету, а потім на їх основі створюється образ предмету)
- Метод генерації квестів – на основі орієнтованого графу (де вузлами являються конкретні завдання, а ребрами – перехід з одного завдання до іншого, тобто послідовність виконання завдань).

Рисунок Г.5 – Результати аналізу методів реалізації ігрових механізмів

Метод автоматизації обчислення характеристик

6

Формула приведення до універсальної одиниці виміру:

$$P = a_1 * k_1 + .. + a_{n-1} * k_{n-1} + a_n * k_n$$

Формула кривої рівня-потужності:

$$x_y = kp * x_{y-1}$$

Формула підрахунку максимальної границі характеристики

$$X_{max} = \frac{M_{max} * kr}{n}$$

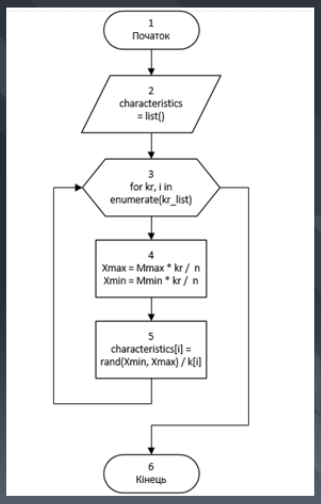


Рисунок Г.6 – Метод автоматизації обчислення характеристик

Метод генерації квестів на основі орієнтованого графу

7

Умови при створенні графа:

- З кожної вершини графа, крім кінцевої, повинна бути можливість дістатись хоча б у одну іншу.
- У графа обов'язково має бути початкова вершина, яка може містити умову початку квесту, та хоча б 1 кінцева.
- Кожний квест має бути здійсненний, тобто такий, який персонаж може виконати.

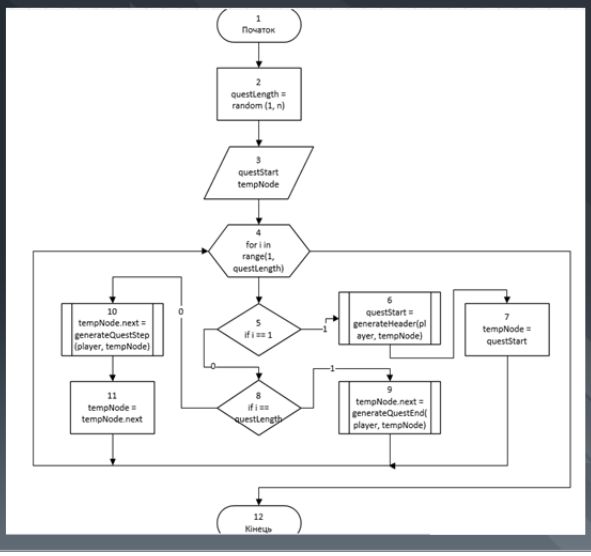


Рисунок Г.7 – Метод генерації квестів на основі орієнтованого графу

Алгоритм PVP ТА PVE Ч.1

8

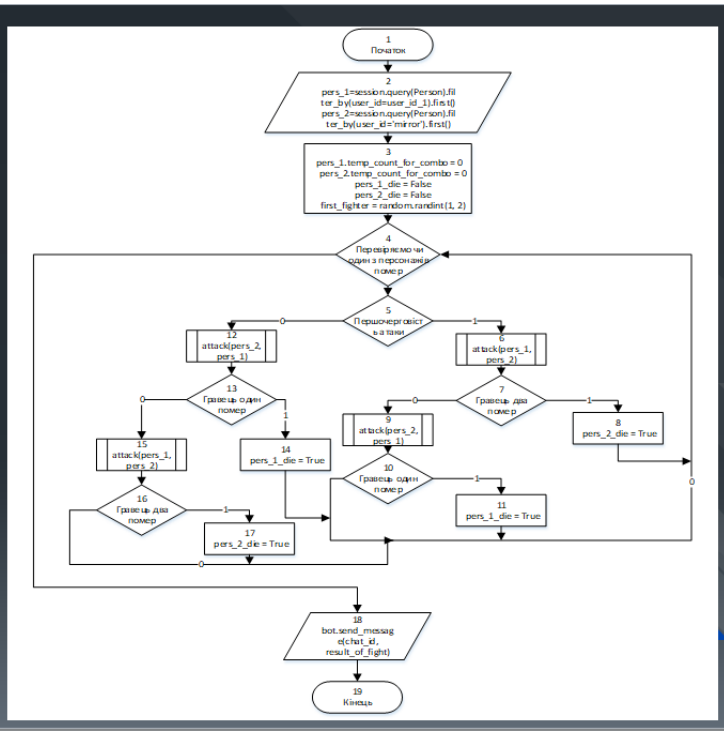


Рисунок Г.8 – Алгоритм PvP та PvE ч.1

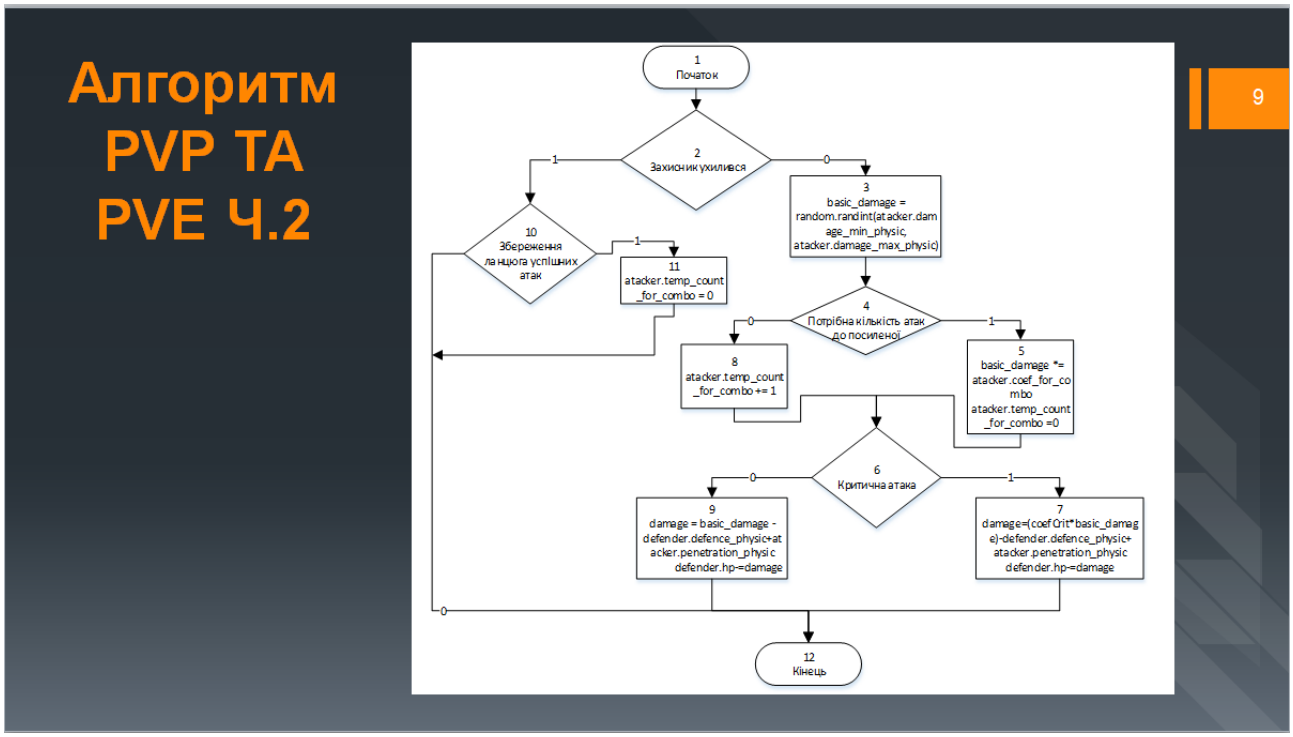


Рисунок Г.9 – Алгоритм PvP та PvE ч.2

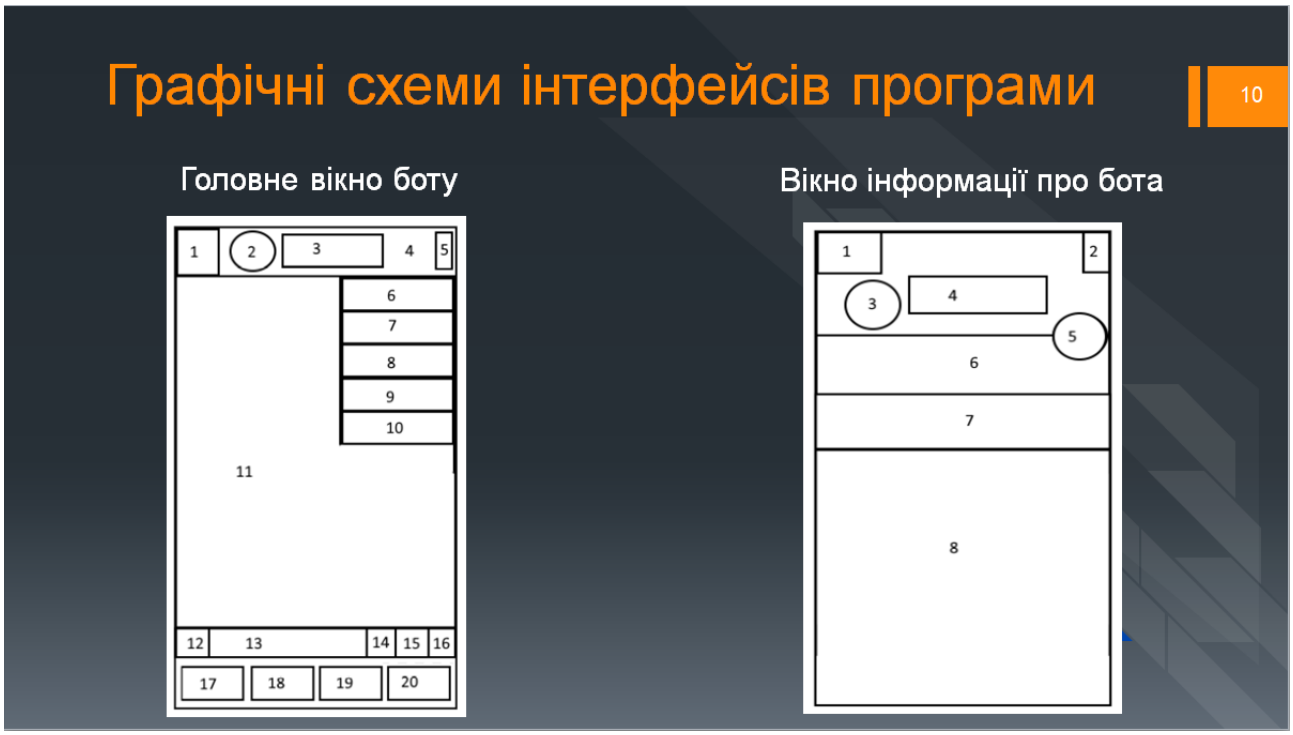


Рисунок Г.10 – Графічні схеми інтерфейсів програми

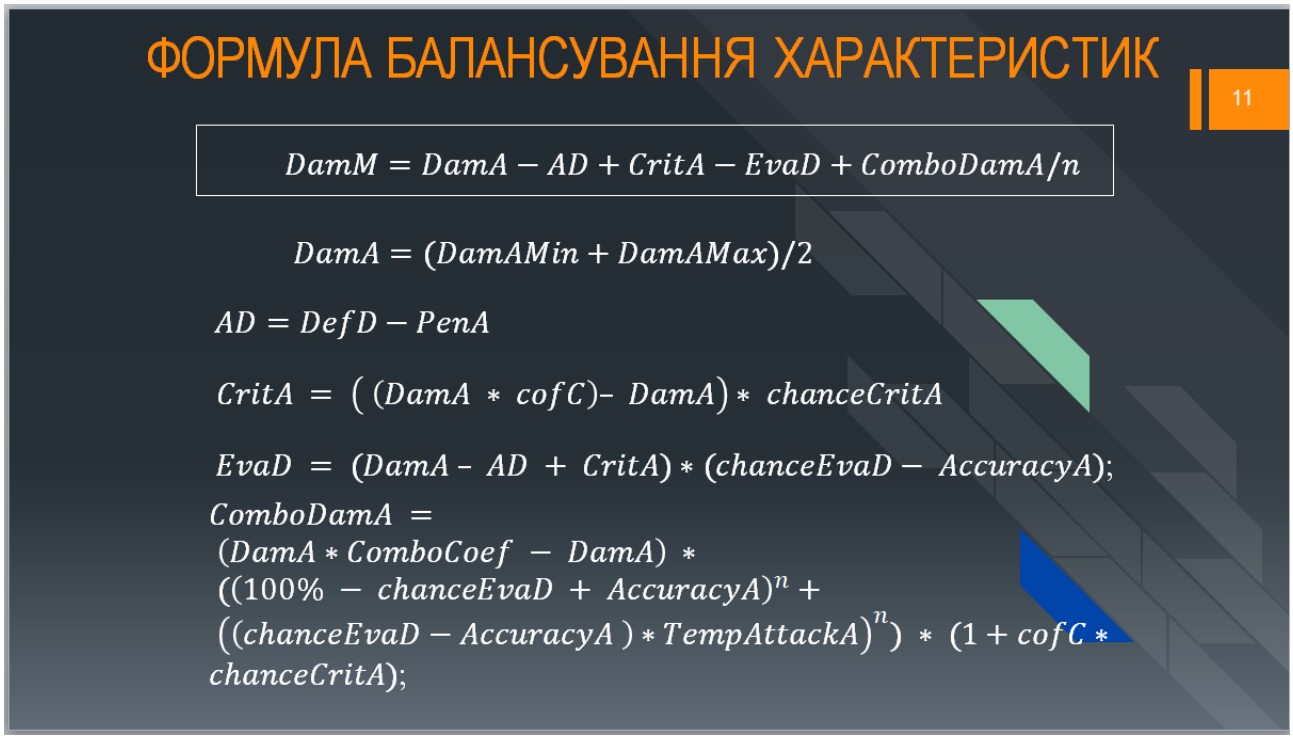


Рисунок Г.11 – Формула балансування характеристик

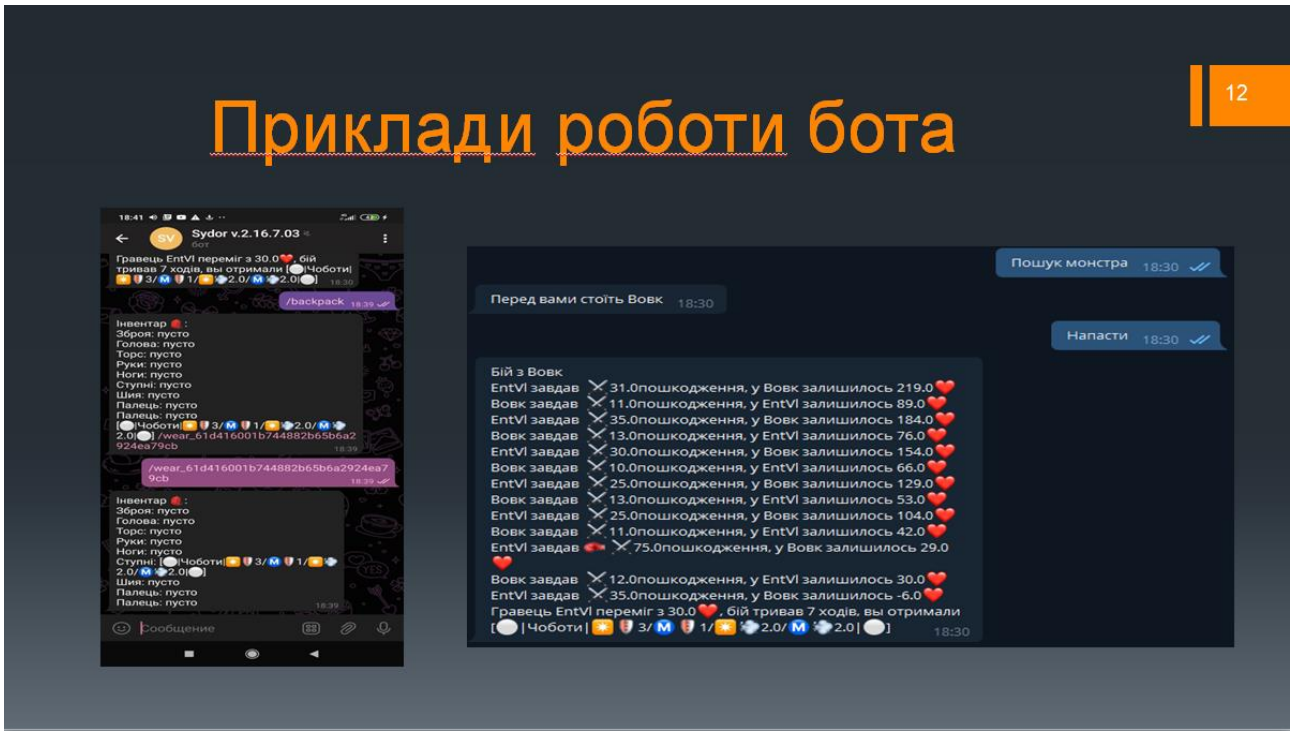


Рисунок Г.12 – Приклад роботи бота

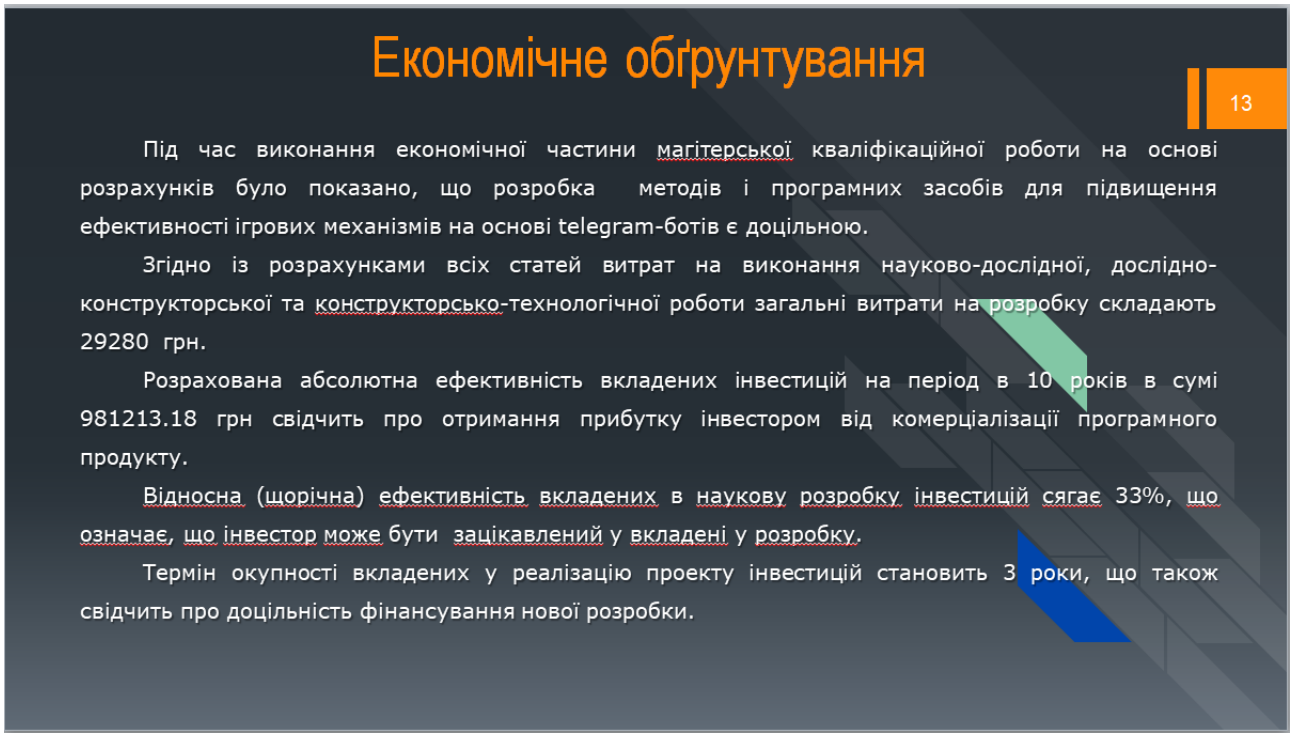


Рисунок Г.13 – Економічне обгрунтування

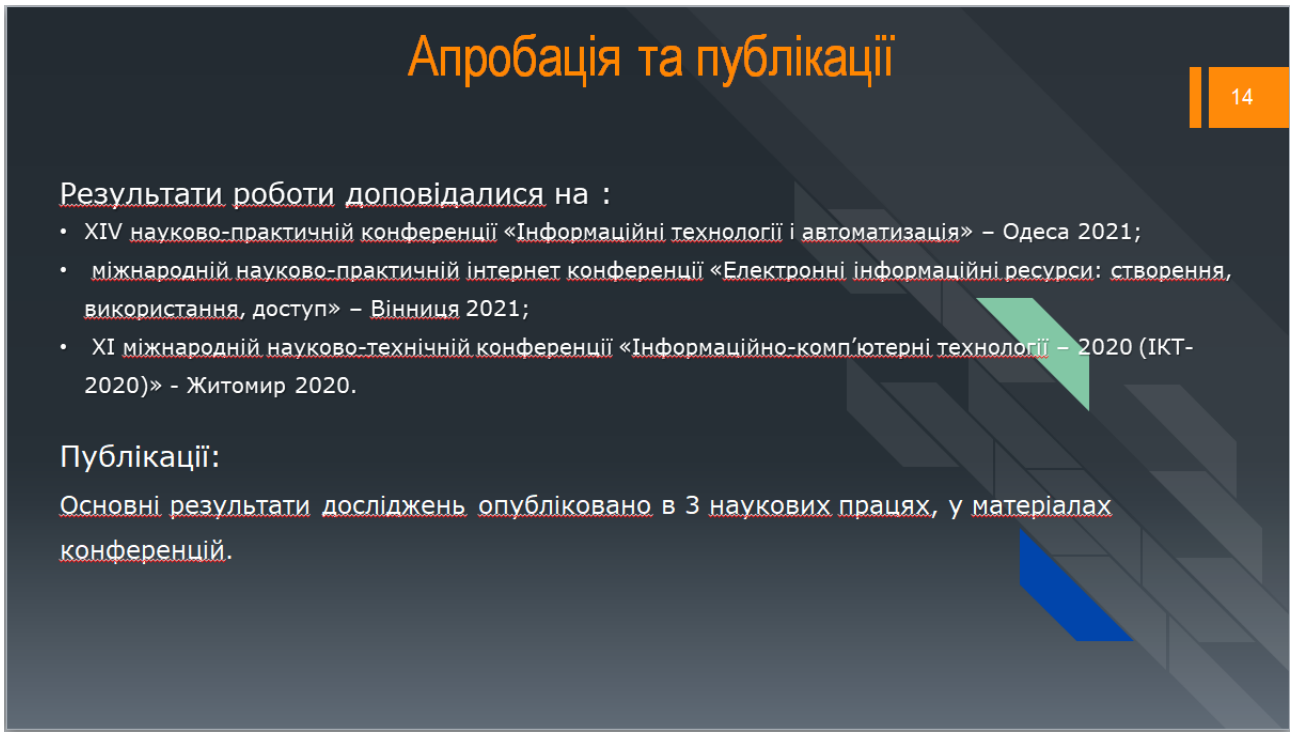


Рисунок Г.14 – Апробації та публікації

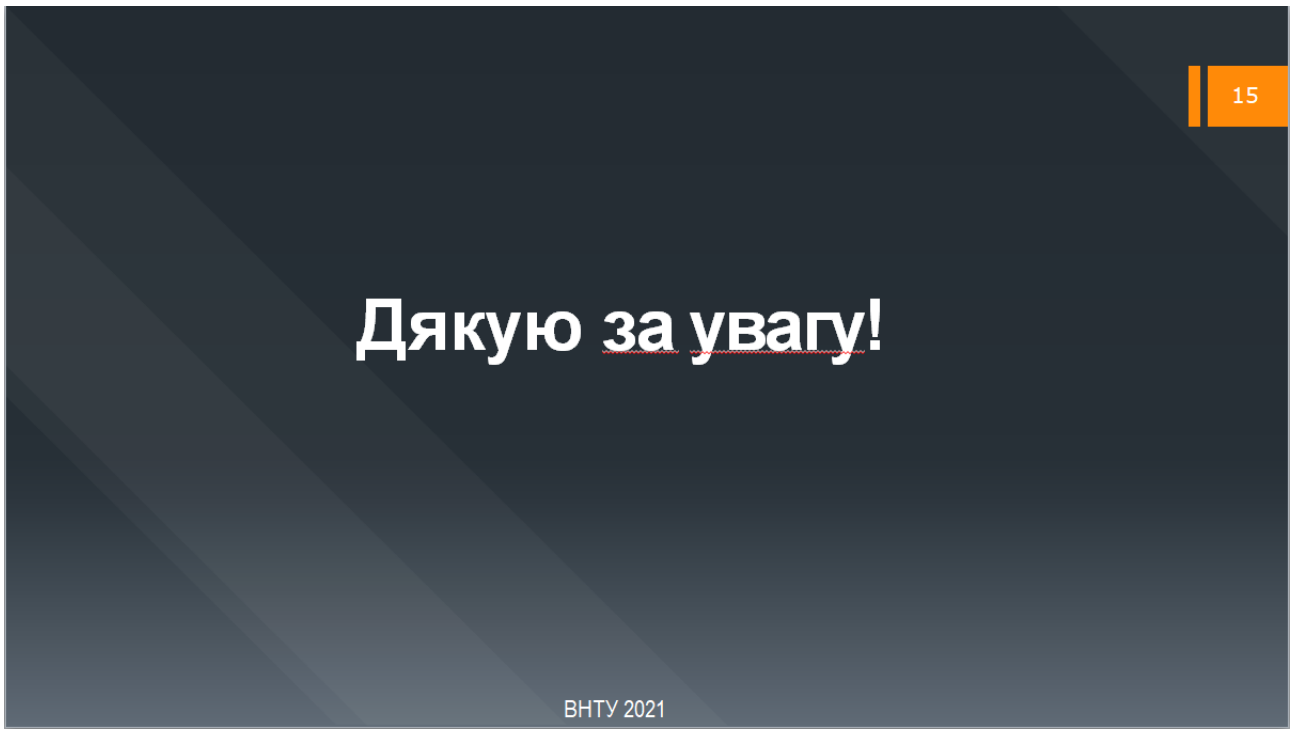


Рисунок Г.15 – Подяка