

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПОШУКУ КОМП'ЮТЕРНИХ ІГОР
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
SEARCHING COMPUTER GAMES

Виконав: студент(ка) 2 курсу, групи КНТ-219м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

Гонтаренко М.О.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 122 Комп'ютерні науки
(код і найменування)
 Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2020 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Гонтаренка Микити Олексійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів пошуку комп'ютерних ігор
Research and Software Implementation of Methods for Searching Computer Games

керівник проєкту (роботи) Скрупський Степан Юрійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “3” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) 15 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Реферат, Перелік скорочень та умовних познач, Вступ, 1. Аналіз проблеми, 2. Матеріали і методи, 3. Опис програми, 4. Розробка програми, 5. Експлуатація та тестування програми, 6. Економіко-організаційна частина, 7. Охорона праці та безпека в надзвичайних ситуаціях, Висновки, Перелік джерел посилання, Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Скрупський С.Ю., доцент		
6 Економіко-організаційна частина	Пожуєва Т.О., професор		
7 Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтроль	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “ 7 ” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програми	7-9 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	10 тиждень	Розділ 5
6	Розробка економіко-організаційної частини	11 тиждень	Розділ 6
7	Розробка заходів з охорони праці та безпеки в надзвичайних ситуаціях	11 тиждень	Розділ 7
8	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
9	Захист роботи	13 тиждень	

Студент(ка)

(підпис)

Гонтаренко М..О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Скрупський С.Ю.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
118 с., 30 рис., 7 табл., 3 дод., 31 джерело.

КОМП'ЮТЕРНА ГРА, ІНФОРМАЦІЙНИЙ ПОШУК,
ПЕРСОНАЛІЗОВАНИЙ ПОШУК, КЛАСТЕРНИЙ АНАЛІЗ.

Об'єкт дослідження – процес пошуку комп'ютерних ігор. Предмет дослідження – методи пошуку комп'ютерних ігор. Мета роботи полягала в розробленні програмного забезпечення агрегації комп'ютерних ігор для забезпечення можливостей персоналізованого пошуку комп'ютерних ігор за запитом користувача.

Матеріали, методи та технічні засоби: мова програмування C#, об'єктно-орієнтоване програмування, кластерний аналіз, система управління базою даних Microsoft SQL Server.

Результати. Виконано проектування та розробку програмного забезпечення агрегації комп'ютерних ігор, що дозволяє вносити дані про ігри, оцінювати їх, створювати власні коментарі та переглядати існуючі, виконувати звичайний пошук та персоналізований пошук ігор.

Висновки. Наукова новизна роботи полягає в тому, що в ній розроблено метод пошуку комп'ютерних ігор, який використовує метод k-середніх для розділення комп'ютерних ігор на кластери, визначає прихильність користувача до ігор відповідного кластеру за відповідями на поставлені запитання або за уподобаними комп'ютерними іграми, що дозволяє виконувати персоналізований пошук у відповідності з тими значеннями характеристик комп'ютерних ігор, яким віддає перевагу користувач.

Галузь використання – ігрові платформи, програмні каталоги комп'ютерних ігор.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 118 pages, 30 figures, 7 tables, 3 appendixes, 31 sources.

COMPUTER GAME, INFORMATION SEARCH, PERSONALIZED SEARCH, CLUSTER ANALYSIS.

The object of research is the process of finding computer games. The subject of research is methods of searching for computer games. The goal of the research was to develop computer game aggregation software to provide personalized search for computer games at the user's request.

Materials, methods and technical means: C# programming language, object-oriented programming, cluster analysis, Microsoft SQL Server database management system.

Results. Computer game aggregation software has been designed and developed. The software allows user to enter data about games, evaluate games, create user own comments and view existing ones, perform regular search and personalized search for games.

Conclusions. The scientific novelty of the work is in the developed method of searching for computer games, which uses the method of k-means to divide computer games into clusters, determines the user commitment to the games of the cluster by answering questions or favorite computer games, which allows user to perform personalized search in accordance with the concepts of the characteristics of computer games, which are preferred by the user.

The field of use is gaming platforms, software catalogs of computer games.

ЗМІСТ

	С.
Реферат	4
Abstract	5
Перелік скорочень та умовних познач	8
Вступ.....	9
1 Аналіз проблеми.....	11
1.1 Комп'ютерні ігри	11
1.2 Огляд програмних засобів доступу до комп'ютерних ігор	19
1.3 Висновки за розділом 1	21
2 Матеріали і методи.....	23
2.1 Методи інформаційного пошуку	23
2.2 Метод k-середніх.....	26
2.3 Розроблений метод пошуку комп'ютерних ігор.....	29
2.4 Висновки за розділом 2	30
3 Опис програми.....	32
3.1 Вибір мови програмування	32
3.2 Моделювання функціональних вимог до програмного забезпечення...	35
3.3 Проектування структури бази даних.....	37
3.4 Проектування структури застосунку	41
3.5 Висновки за розділом 3	43
4 Розробка програми	44
4.1 Алгоритм функціонування програми.....	44
4.2 Опис розроблених модулів.....	46
4.3 Висновки за розділом 4	50
5 Експлуатація та тестування програми	51
5.1 Призначення програми	51
5.2 Умови виконання програми	51
5.3 Виконання програми.....	51
6 Економіко-організаційна частина.....	59

6.1	Планування розробки програмного продукту.....	59
6.2	Визначення витрат на розробку програми	61
6.2.1	Основна заробітна плата	62
6.2.2	Розрахунок додаткової заробітної плати	62
6.2.3	Відрахування на єдиний соціальний внесок	63
6.2.4	Визначення витрат на матеріали	63
6.2.5	Витрати на спеціальне обладнання	64
6.2.6	Розрахунок накладних витрат.....	67
6.3	Визначення економічної ефективності програмного продукту	68
7	Охорона праці та безпека в надзвичайних ситуаціях.....	71
7.1	Аналіз потенційних небезпек.....	71
7.2	Заходи щодо забезпечення безпеки.....	71
7.3	Заходи щодо забезпечення виробничої санітарії та гігієни праці.....	72
7.4	Заходи щодо забезпечення пожежної безпеки.....	78
7.5	Заходи щодо забезпечення безпеки в надзвичайних ситуаціях	80
	Висновки	81
	Перелік джерел посилання	83
	Додаток А Технічне завдання	86
	Додаток Б Текст програми	90
	Додаток В Слайди презентації.....	110

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

RPG	– Role-Playing Game;
UML	– Unified Modeling Language;
ВДТ	– візуальний дисплейний термінал;
ЕОМ	– електронно-обчислювальна машина;
ОЦ	– обчислювальний центр;
ПК	– персональний комп'ютер.

ВСТУП

Останнім часом комп'ютерні ігри стали дуже популярні в зв'язку з величезними відкриттями в індустрії графіки: сфера розробки ігор стала одним величезним механізмом, що об'єднав професіоналів з усіх куточків планети, специфіка їх роботи різна, але всі вони переслідують одну і ту ж мету – зробити світ веселіше і яскравіше, швидше за все 50 років тому ніхто не міг подумати, що середньостатистична людина буде проводити близько шести годин в день за відеоіграми, тому не дивно, що індустрія ігор розвивається стрімкими темпами [1].

Такий активний розвиток індустрії комп'ютерних ігор є з одного боку її плюсом, а з іншого призводить до недоліку – активний розвиток призводить до фрагментації, ігор стає все більше, але знайти потрібну гру, якщо вона не є найпопулярнішою, стає складніше. З іншого боку це проблема не тільки для гравців, але і для розробників ігор, адже за умов відсутності персоналізованого пошуку їм достатньо складно знайти аудиторію для своїх гри. Тому створення програмного забезпечення, що накопичує в собі інформацію про різні ігри, а головне – дозволяє за інформацією від користувача знаходити адекватні для нього ігри, є важливим практичним завданням, для якого має бути забезпечена адекватна теоретична частина з визначенням потрібного методу пошуку.

Метою роботи було визначено розроблення програмного забезпечення агрегації комп'ютерних ігор для забезпечення можливостей персоналізованого пошуку комп'ютерних ігор за запитом користувача.

Для досягнення цієї мети було визначено наступні завдання роботи:

- аналіз проблеми;
- дослідження методів пошуку комп'ютерних ігор;
- розробка технічного завдання;
- вибір засобів розробки;
- проєктування структури програмного забезпечення;

- розробка програмного забезпечення;
- тестування та відлагодження програми;
- економічне обґрунтування доцільності розробки.

1 АНАЛІЗ ПРОБЛЕМИ

1.1 Комп'ютерні ігри

Комп'ютерні ігри – це різновид прикладного програмного забезпечення, популярний спосіб проведення дозвілля і високододаткова галузь в сфері інформаційних технологій, залежно від тих чи інших ознак ігри відносять до певних жанрів, при чому типова класифікація комп'ютерних ігор заснована на понятті жанр, який визначає сукупність характерних рис, притаманних тій чи іншій грі, цей підхід не стандартизований, але популярний і його дотримуються як представники ігрової індустрії, так і споживачі ігрових застосунків, тож до основних жанрів належать [2]:

а) аркади – категорія комп'ютерних ігор з простим ігровим процесом, що зводиться до виконання обмеженого набору дій, націлених на досягнення певного результату, завдання ігор цього жанру, як правило, зводяться до набору максимальної кількості очок і/або мінімального часу проходження гри [1];

б) пригоди: пригоди (також відомі як квести) – сюжетні ігри, в яких герой виконує різні завдання, для чого йому потрібно спілкування з іншими персонажами, пошук і використання різних предметів, рішення задач і головоломок, піджанри [3]:

1) текстовий квест: перші текстові квести називали «інтерактивними книгами», як і випливає з назви, геймплей заснований на текстових повідомленнях, тобто у відповідь на запрограмовані в грі розгалуження сюжету або ситуації гравці вводять текстові команди, при створенні текстових квестів програмісти витрачали чимало часу на розробку різних варіантів відповіді на команди гравця [2];

2) графічний квест: у міру того, як комп'ютери ставали все більш потужними і здатними разом з текстом виводити графіку, розвивалися і самі ігри, наприклад, в перших графічних квестах прості

картинки лише доповнювали текстові пригоди, надалі, коли миша стала повноцінним ігровим контролером, ігри стали замінювати текстові команди на дії мишею [1];

3) візуальна новела: у більшості візуальних новел, шалено популярних насамперед в Японії, гравець повинен вибрати особливості або параметри персонажа, які впливають на геймплей, зазвичай в цих іграх безліч кінцівок, обумовлених діями гравця в певних точках сюжету, популярними темами багатьох візуальних новел є симуляції побачень або судових процесів (наприклад, *Асе Attorney* для Nintendo DS) [4];

4) технологія лазерних дисків дозволила створити інтерактивні фільми: у них використовуються заздалегідь зняті сцени з живими акторами або мультиплікацією, пригода зазвичай представляється з видом від третьої особи, а гравець контролює вибір в певних вузлових точках сюжету [2];

в) рольові ігри (або Role-Playing Game (RPG)) – багатопланові ігри з нелінійним сюжетом, де гравець, керуючи персонажем, має можливість покращувати його здатності, для цього потрібно перемагати ворогів, виконувати завдання, продавати і купувати артефакти тощо, на характеристики персонажа можуть впливати своєчасний прийом їжі, сон тощо, те, на скільки цікаві такі ігри, багато в чому визначається адекватністю штучного інтелекту, керуючого поведінкою інших персонажів, піджанри [3]:

1) екшен-RPG запозичують елементи жанрів action і action-adventure: визначальною характеристикою action-RPG можна назвати бойову систему в реальному часі, де все часто залежить від швидкості і реакції гравця, а не тільки від високих параметрів його персонажа на кшталт спритності і харизми [5];

2) roguelike – ще один жанр, який отримав назву від гри *Rogue*, яка була двомірною комп'ютерною грою в жанрі «зачистки підземель» 1980 року, у ній був текстовий інтерфейс і випадкова генерація рівнів,

для прокачування параметрів своїх персонажів гравці повинні були долати перешкоди і перемогати ворогів, окремо варто згадати механіку перманентної смерті персонажа, завдяки якій ці ігри по праву вважаються одними з найбільш хардкорних представників жанру, ці базові принципи збереглися до наших днів, отримавши покращене графічне і геймплейне оформлення [2];

3) тактичні RPG: ігровий процес тактичних RPG ближче до їх настільних корінь і часто являє собою покрокові дії на ігровому полі, ізометричному або повністю тривимірному, для перемоги над ворогами у гравців є обмежені ресурси (армії, зброя) і потрібна практично шахова стратегія [1];

4) RPG з відкритим світом: рольові ігри з відкритим світом, або «пісочниці», дозволяють гравцям вільно подорожувати по своєму ігровому середовищу в пошуках пригод, це одні з найбільш атмосферних і захоплюючих ігор, тому що величезна кількість ігрових персонажів і ситуацій, необхідних для створення побічних квестів і додаткових сюжетних ліній, дозволяє розробникам створювати живі віртуальні світи [3];

5) партійні RPG від першої особи: популярний раніше піджанр, який нині здав свої позиції action-RPG і тактичній ізометрії, являє собою рольові ігри, де гравець управляє командою шукачів пригод, яка повністю створюється на початку гри або збирається по ходу подорожі, дія зазвичай відбувається в підземеллях, вся партія пересувається як єдине ціле, а битви, як правило, покрокові, хоча іноді можуть відбуватися і в реальному часі, серед найбільш відомих представників жанру можна назвати серію Might and Magic і старі Bard's Tale [4];

г) розвиток широкосмугового Інтернету призвів до появи різновиду рольових ігр – масових багатокористувацьких рольових онлайн-ігор: у цьому

виді ігрових програм віртуальні персонажі, керовані реальними гравцями, здатні взаємодіяти один з одним [5];

д) симулятор – ігрові програми, що імітують управління будь-яким процесом або пристроєм, в основі хороших симуляторів – математичні моделі об'єктів для огляду і процесів, чим повніше описана модель і її оточення, тим реалістичніше симулятор, хороші комп'ютерні симулятори можуть використовуватися в якості тренажерів, піджанри [2]:

1) симулятори будівництва і менеджменту: найпопулярнішою грою цього жанру по праву є SimCity – у ній гравець повинен будувати і розвивати місто, а також керувати ним, зокрема, планувати вулиці, розподіляти простір і стягувати податки з городян, в інших іграх охоплення може бути обмежене більшими чи меншими рамками, наприклад, кафе, хмарочосом, парком розваг або цілою планетою [3];

2) симулятори життя: The Sims – найпопулярніший симулятор життя і одна з найбільш продаваних ігор всіх часів, як і в SimCity (також створеної Уіллом Райтом) гравець управляє окремими аспектами штучного життя, такі симулятори можуть дозволяти гравцям управляти генетикою персонажа або середовищем їхнього життя, у деяких випадках можна контролювати навіть реакцію персонажа на певні ситуації, ігри з віртуальними вихованцями, наприклад Nintendogs, також можна віднести до цієї категорії – по суті, це симулятори догляду за тваринами, таким чином, The Sims можна назвати соціальним симулятором, і кожна з цих ігор в певному аспекті є симуляцією життя, нарешті, деякі найбільш масштабні ігри цієї категорії надають персонажам значний контроль над персонажами і їх оточенням і тому називаються «симуляторами бога» [1];

3) симулятори техніки: у цій категорії досить важко виділити найбільш популярні ігри, тому що продажі приблизно порівну діляться між авіасимуляторами і гонками, симулятор техніки намагається відтворити відчуття польоту на літаку або вертольоті,

водіння гоночною машиною і навіть керування трактором на фермі, бій в цих іграх теж зустрічається (найчастіше в авіаційних і танкових симуляторах), також сюди відносяться симулятори поїздів і космічні симулятори [4];

е) стратегічні ігри: завдання ігор цього жанру зводиться до отримання переваг над противником, досяжних шляхом вироблення і реалізації певного плану, об'єктами управління є не окремі персонажі, а цілі корпорації, армії, держави і навіть цивілізації, розрізняють покрокові стратегічні ігри, де гравці ходять по черзі і стратегії реального часу, в яких всі гравці діють асинхронно, піджанри [2]:

1) 4X-стратегії: 4X можна назвати будь-яку стратегію, де можна виділити чотири основні цілі: досліджувати, розвиватися, експлуатувати і знищувати, найвідомішою грою цієї категорії є Civilization Сіда Мейєра, завдяки масштабу закладених цілей у багатьох іграх цього жанру дія охоплює цілі епохи життя цивілізації (людської або інопланетної) [5];

2) стратегії в реальному часі: у стратегіях в реальному часі від гравця потрібно збирати і розподіляти ресурси, будувати бази, а також розвивати економіку і військову силу, найбільш популярною стратегією в реальному часі в даний час можна назвати Starcraft, варто згадати також серії Warcraft, Age of Empires і Command & Conquer [3];

3) тактика в реальному часі іноді вважається піджанром стратегії в реальному часі, але зосереджена на управлінні підрозділом в бою, а не на менеджменті ресурсів та окремих юнітів [2];

4) tower defense: в обороні веж гравці повинні відображати хвилі керованих комп'ютером ворогів, конкретні здатності веж ті ворогів в різних іграх можуть відрізнятися, але зазвичай є декілька веж з різними здібностями, наприклад, уповільненням або отруєнням, знищивши певну кількість ворогів, гравець отримує ресурси на

покупку нових веж або оновлення існуючих, наприклад, збільшення шкоди або дальності [1];

5) покрокові стратегії – ця категорія відрізняє покрокові стратегії від їх аналогій в реальному часі, у покроковій стратегії гравець отримує певний проміжок часу (хід), в який він виконує деякі дії, як правило, під час ходу гравця противники (інші гравці або керовані комп'ютером) не рухаються, але, як і у випадку і стратегією в реальному часі, в цьому жанрі можливі відхилення від класичної покрокової схеми, наприклад, переміщення по глобальній карті в реальному часі з переходом в покроковий режим в бою, перехоплення дій і різні концепції одночасного ходу, найбільш відомий представник жанру – серія Heroes of Might and Magic [3];

б) варгейми: геймплей цього піджанру стратегій зосереджений на тактичних або стратегічних дослідженнях з прив'язкою до карти, найчастіше без економічної складової і видобутку ресурсів, ігри можуть бути покроковими і в реальному часі, дуже часто використовуються історичні сетінги (в переважній більшості випадків – Друга світова війна), серед найбільш яскравих представників можна назвати серії Panzer General, Close Combat, Sudden Strike [2];

ж) бойовики – цей жанр комп'ютерних ігор, більш відомий під калькою «екшен» – один з найпопулярніших видів комп'ютерних ігор: швидкість реакції і здатність оперативно приймати рішення - основні складові успішного проходження таких ігор, гравець знаходиться в самому центрі дії і керує процесом, головним чином пов'язана з фізичними випробуваннями, які потрібно подолати, до цього жанру належали найперші відеоігри, наприклад, Galaga і Donkey Kong, зазвичай в екшенах дуже легко приступити до гри, тому вони до сих пір залишаються найпопулярнішими відеоіграми, до піджанрів належать [2]:

1) платформери отримали свою назву завдяки тому, що в процесі гри її головний персонаж взаємодіє з платформами – зазвичай бігає, стрибає або падає: різноманітних платформер дуже багато, найвідомішим з них можна назвати серію Super Mario, а Donkey Kong був одним з найперших [4];

2) шутери дозволяють гравцям використовувати в своїх діях зброю, зазвичай з метою знищити ворогів або протиборчих гравців, шутери діляться по ракурсу огляду гравця: у шутерах від першої особи вид з очей головного персонажа (типовими прикладами є Call of Duty, Half-Life і Halo), у шутерах від третьої особи (Fortnite і Splatoon) дію показано так, що гравець може бачити головного персонажа зазвичай трохи зверху і ззаду, шутери з видом зверху, наприклад, Galaga, Space Invaders і Raiden V: Director's Cut, як і впливає з назви, демонструють вид повністю зверху, і якщо в шутерах від третьої особи зазвичай використовується шкала здоров'я, яка росте або зменшується в залежності від рівня здоров'я або стану персонажа, то в шутерах з видом зверху часто застосовується принцип з деякою кількістю «життів», коли напис «Game Over» з'являється після вичерпання їх запасу [5];

3) у файтингах, зокрема, Mortal Kombat і Street Fighter II, вся дія зосереджена на боях, які найчастіше проходять у формі єдиноборств: зазвичай в файтингах великий набір іграбельних персонажів, кожен з яких спеціалізується на своїх унікальних здібностях або бойовому стилі, у більшості традиційних файтингів гравець повинен битися, просуваючись все вище і перемагаючи все більш і більш сильних супротивників [2];

4) ігри жанру Beat-em up, або броулери, також зосереджені на бійках, але замість бою з одним суперником гравці стикаються з цілими хвилями ворогів, однією з перших ігор цього жанру була

Double Dragon, серед більш недавніх можна назвати серію Batman Arkham або Castle Crashers [1];

5) такий піджанр бойовиків, як survival-action, в останні кілька років завоював значну популярність: одним з перших його представників була гра в жанрі survival-horror Resident Evil, хоч вона і була лінійною, тоді як дія більш сучасних ігор на виживання типу Fortnite зазвичай відбувається у відкритому світі, а гравці отримують можливості і ресурси для створення інструментів, зброї і укриттів з метою вижити максимально довго [3];

з) інші: сюди віднесемо ті ігри, які не потрапляють в перераховані вище жанри або до яких взагалі складно застосувати поняття «жанр», прикладами таких ігор є комп'ютерні версії шахів, шашок, карткових ігор, кросвордів тощо [2]-[3].

За даними компанії NewZoo, яка є лідуючим глобальним провайдером комп'ютерних ігор, у 2019 році світовий дохід від мобільних ігор зріс до 68,5 млрд. доларів, а за оцінками Newzoo, до кінця 2020 року ця цифра досягне 76,7 млрд. доларів – зростання відбудеться на 12%, при цьому загалом всі види комп'ютерних ігор дозволили зібрати протягом минулого року 151.9 млрд. доларів (рис. 1.1), що більше ніж на 10 % перевищує позаминулорічний показник [4]-[5].

Ігри стали повсюдним джерелом розваги – якщо скласти всі годинники активної гри і її перегляду, то вийде, що це найулюбленіше заняття в світі, хоча учасники цього ком'юніті і відрізняються за способом взаємодії з контентом, Newzoo виділила три головних категорії: гравці, глядачі і власники, третя група позначає людей, які скуповують продукти, призначені для ігор або пов'язані з ними, раніше всіх учасників ком'юніті називали «геймерами», але тепер Newzoo позбавляється від цього терміна, замінивши його на більш широке поняття – «любителі ігор» [6].

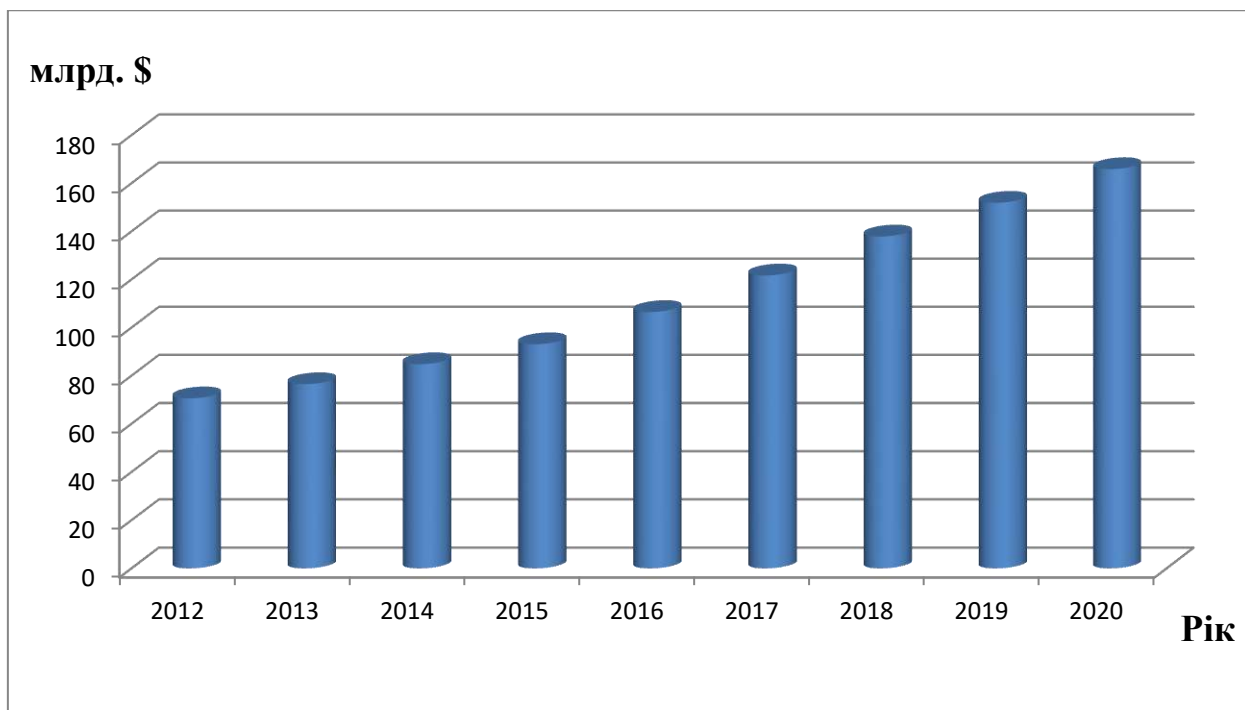


Рисунок 1.1 – Розподіл фінансів та аудиторії комп'ютерних ігор за роками

Число геймерів в світі перевищило 2,3 млрд людей, з них майже 95% грають в мобільні ігри, розвиток сегменту підігріло розширення лінійки доступних китайських смартфонів, що збільшило кількість мобільних користувачів з невеликим доходом з країн, що розвиваються, у них немає можливості витратити гроші на мобільні ігри, в зв'язку з чим в цих регіонах розвивається реклама в ігрових застосунках [6].

1.2 Огляд програмних засобів доступу до комп'ютерних ігор

Як видно з результатів, представлених у підрозділі 1.1, комп'ютерна індустрія зараз активно розвивається і загалом має величезні прибутки. Одночасно варто відзначити, що існує величезна кількість ігор, їх жанрів, платформ, на яких ці ігри розвиваються, відповідно важливим є створення накопичувального майданчика, звідки можна отримати інформацію про будь-яку гру, а вже звідти перейти на веб-сайт цієї гри. У всіх цих умовах важливим стає також не тільки наявність такого майданчика, але і наявність в

ньому засобів персоналізованого пошуку. Для цього потрібно виконати огляд програмних засобів, які використовуються на даний момент в цій задачі.

Одним з вебсайтів, які забезпечують можливості агрегації комп'ютерних ігор є вебсайт metacritic.com (рис. 1.2).

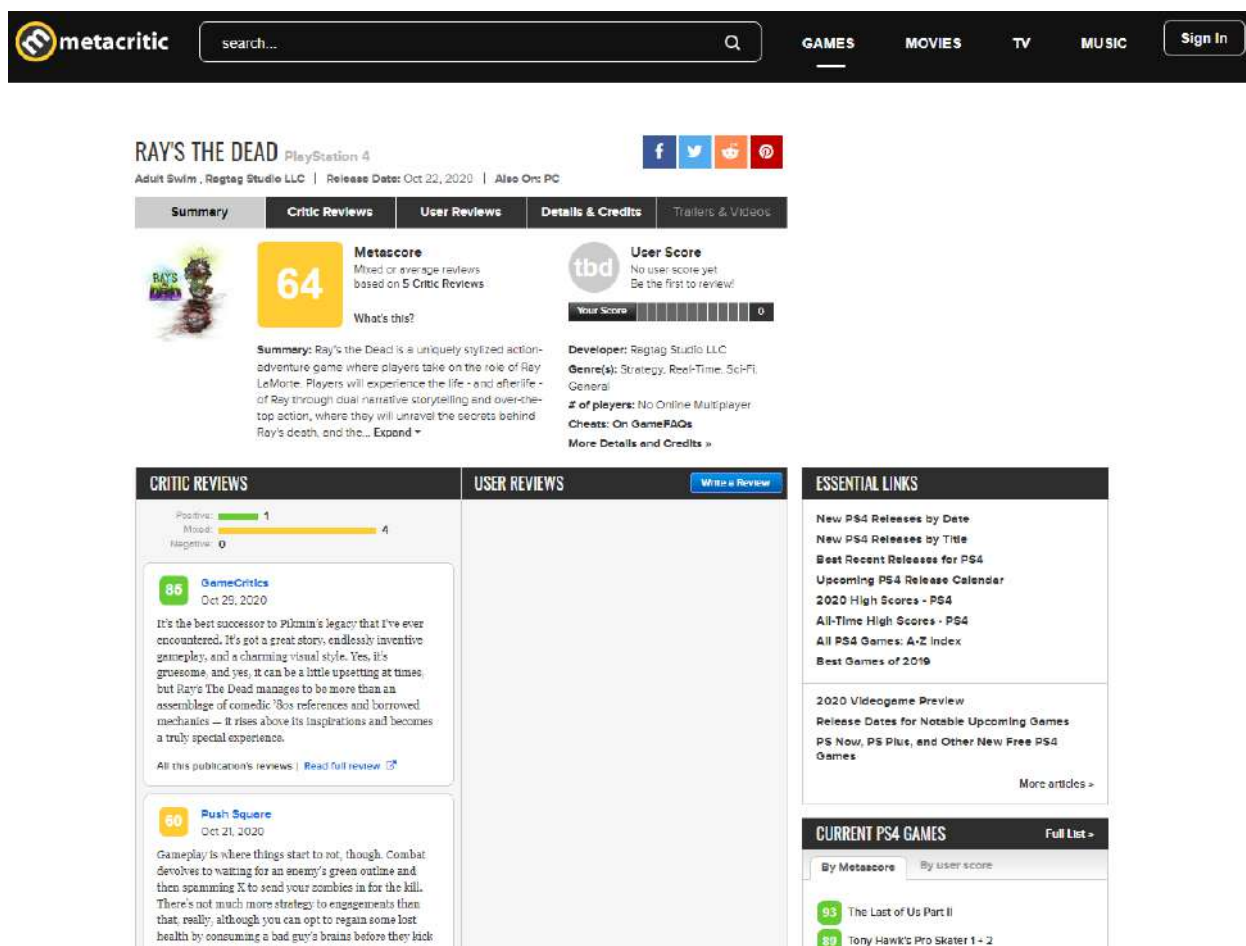


Рисунок 1.2 – Вебсайт metacritic

Він надає можливості виконання наступних функцій:

- переглядати існуючі ігри та отримувати інформацію про них;
- переглядати ігри за жанрами, платформами, нові ігри;
- переглядати рейтинг ігор;
- переглядати та додавати власні рецензії на ігри з виставленням оцінок;
- переглядати і додавати власні відгуки;
- переходити на зовнішні ресурси для придбання ігор;

- переглядати трейлери та відеороліки про ігри;
- виконувати пошук ігор.

При цьому важливо відзначити, що пошук виконується тільки за назвою. Можливостей персоналізованого пошуку для користувачів (за інтересами) немає.

Вебсайт xbox.com (рис. 1.3) дозволяє переглядати рекламну інформацію про ігри, які можна використовувати на платформі xbox, надає можливість купувати підписки на ігри, виконувати пошук ігор, отримувати доступ до переліків ігор за платформою. Загалом вебсайт організований за принципом інтернет-магазину, де в якості товарів виступають комп'ютерні ігри. Особливих налаштувань пошуку для можливостей персоналізованого дослідження сфери наявних ігор вебсайт не має.

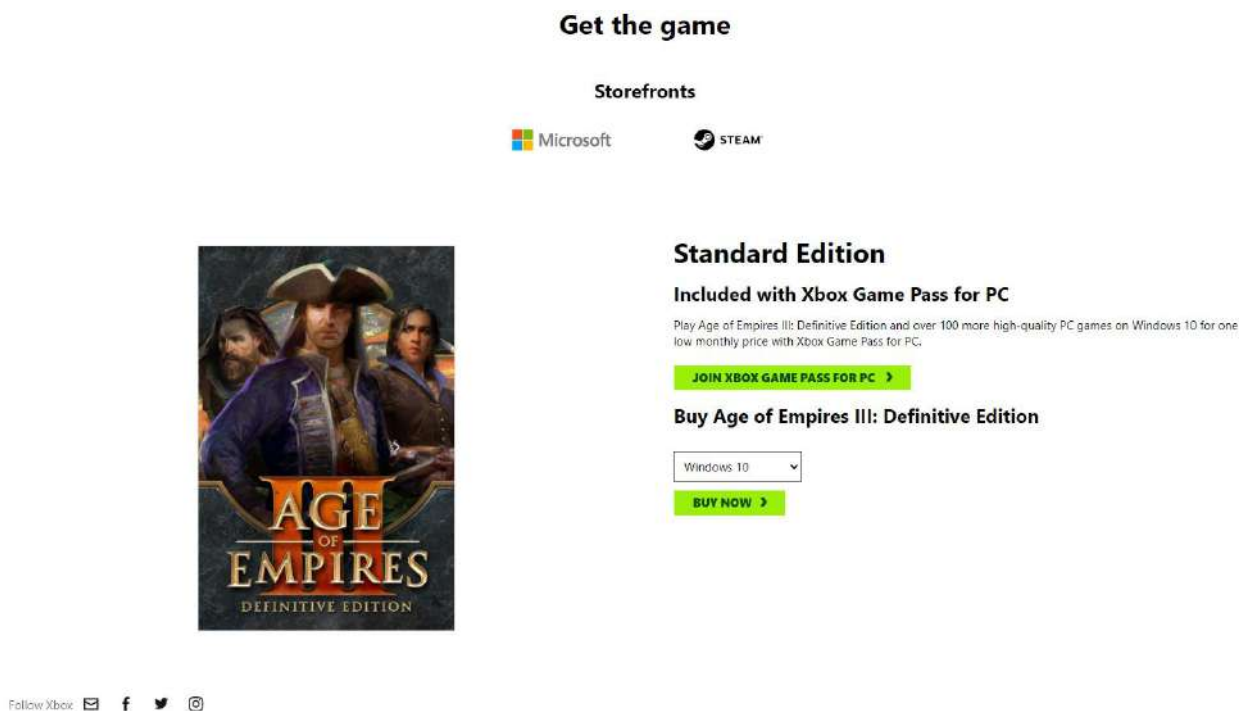


Рисунок 1.3 – Вебсайт xbox

1.3 Висновки за розділом 1

У першому розділі виконано аналіз проблеми. Для цього розглянуто поняття комп'ютерних ігор, особливості жанрового різноманіття, масштаби

та тенденції розвитку ринку комп'ютерних ігор. З цього можна зробити висновок, що величезна кількість ігор, жанрова фрагментація та фрагментація за платформами призводять до необхідності створення платформ для доступу до даних різних ігор, тобто агрегації комп'ютерних ігор. В якості таких платформ було розглянуто і проаналізовано функціональні можливості платформ metacritic та xbox.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Методи інформаційного пошуку

Пошук інформації найчастіше відбувається за релевантністю. Якщо є деяка множина електронних текстових документів і існує користувач, який, взаємодіючи з системою і формулюючи пошукові запити, отримує множину документів, які задовольняють запиту, то у цьому випадку завданням релевантного пошуку є якомога повніша й точніша вибірка підмножини документів найбільш близьких за змістом (релевантних) пошуковому запиту, при цьому під поняттям близькості документа пошуковому запиту слід розуміти деяку функцію кореляції між кожним документом і запитом, обчислення якої дозволяє визначити ступінь близькості інформаційного змісту документа і запиту, вид функції релевантності залежить від конкретної реалізації інформаційно-пошукових систем, у більшості інформаційно-пошукових систем функція релевантності документа запиту є ймовірнісною (тобто про відповідність документа пошуковому запиту можна говорити тільки з якоюсь мірою вірогідності) і аналізується на деякій інформації про документи, автоматично одержуваній системою в ході аналізу їх інформаційного контексту документів (аж до аналізу синтаксису і семантики) [7].

Інформаційний пошук текстів – одне з найбільш затребуваних завдань обробки текстів, центральна проблема – допомогти користувачеві знайти ту інформацію, в якій він зацікавлений, завдання пошуку полягає у визначенні за запитом користувача множини текстів з деякої фіксованої бази, релевантних запиту, запит подається набором ключових слів, основним питанням при цьому є визначення релевантних текстів запиту і сортування документів за цими значеннями, класична задача інформаційного пошуку, з якої почався розвиток цієї галузі, – це пошук документів, що задовольняють запиту в рамках деякої статичної (на момент виконання пошуку) колекції документів, це завдання вирішується в рамках більшості сучасних

довідкових систем, включаючи Windows, моделі інформаційного пошуку поділяються на три класи [8]:

а) теоретико-множинні моделі, які спираються на апарат теорії множин, класичний приклад – булева модель: в рамках цієї моделі документи і запити представляються у вигляді множин термів, запити будуються на основі елементарних термінів документів (слів або словоформ), що знаходяться між собою у відносинах, визначених предикатами, такими як диз'юнкція, кон'юнкція і заперечення, булеві запити зазвичай використовуються в пошуку на точну відповідність, в якому документи перевіряються на наявність того чи іншого терміна; логічний пошук спирається на використання інвертованого індексу ключових слів, тобто таблиці, в якій для кожного ключового слова перераховуються всі документи, де воно зустрічається [7]:

- 1) головною перевагою цього алгоритму є можливість зв'язування слів запиту логічними операціями і отримання в результаті об'єднання множин документів, що містять шукані слова;

- 2) до недоліків слід віднести неможливість визначення релевантності запиту отриманої вибірки документів;

- 3) при пошуку з інвертованого індексу беруться списки документів, що відповідають кожному слову запиту, над отриманими множинами проводяться операції, відповідні логічним операціям, що зв'язують слова запиту, в результаті чого утворюється список знайдених документів, як правило, даний алгоритм використовується спільно з іншими алгоритмами;

б) імовірнісні моделі, які спираються на теорію ймовірностей: як оцінки релевантності документа запиту користувача використовується ймовірність того, що користувач визнає документ істинно релевантним [8]:

- 1) загалом імовірнісна модель інформаційного пошуку заснована на теорії ймовірності та використовує статистичні

показники, що характеризують ймовірність відповідності проіндексованих текстових ресурсів запиту користувача;

2) перевага в тому, що модель розташовує документи в порядку зменшення ймовірності документу опинитися релевантним, проте на практиці ці моделі не набули великого поширення;

3) в рамках моделей обчислюється умовна ймовірність події, що документ відповідає даному запиту, а для розрахунку використовується формула Байєса;

4) як фактори, що впливають на безумовну релевантність документа, можна розглядати його розмір, джерело, дату публікації;

5) імовірність запиту за умови релевантності документа залежить головним чином від ваги ключових слів запиту в документі: для її розрахунку зазвичай приймають гіпотезу незалежності слів документа і запиту;

в) алгебраїчні моделі, в яких документи і запити описуються у вигляді векторів в багатовимірному просторі: апарат спирається на алгебраїчні методи, які широко застосовуються в сучасних інформаційно-пошукових системах, так векторна модель є класичним представником класу алгебраїчних моделей, реалізована Джерардом Солтоном в пошуковій системі, при цьому скорочене позначення $TF*IDF$ – синонім найбільш поширеної сучасної векторної моделі, заснованої на математичному апараті геометрії, в якій індексовані текстові ресурси і запити користувачів розглядаються як вектори в просторі слів, а релевантність – як відстань між ними, векторні моделі, на відміну від булевих, дозволяють ранжувати результуючу множину документів запиту, у векторній моделі кожному документу ставиться у відповідність вектор, кожен елемент якого визначений вагою кожного ключового слова у відповідному документі, що обчислюється зазвичай за формулою нормованого уявлення $TF*IDF$, відповідно, аналогічно вводиться вектор для запиту, міра схожості документа і запиту обчислюється як косинус кута між відповідними векторами [7];

г) у реальних пошукових системах, як правило, використовується комбінація розглянутих методів, при цьому логічний пошук використовується для виділення з усього масиву тих документів, які містять всі слова запиту, для визначення релевантності документів і сортування отриманої вибірки використовуються алгоритми векторного і непевного пошуку, булева складова індексування, яка сильно прискорює процес пошуку, – невід'ємна частина пошукових систем, що говорить про необхідність створення і підтримки інвертованого індексу [8].

Більшість сучасних інформаційно-пошукових систем для здійснення пошуку будують на основі вихідної інформації логічні і фізичні структури даних, що представляють собою пошуковий індекс, який дозволяє реалізувати деяку задану модель інформаційного пошуку, перетворення інформації в інформаційно-пошукових системах, що будують пошуковий індекс, зазвичай складається з наступних базових етапів:

- аналіз даних вихідної множини текстових документів і їх перетворення в вид, зручний для побудови повнотекстового індексу обчислювальною машиною, виділення з документів змістовної інформаційної основи;

- аналіз даних отриманих на попередньому кроці і подальша побудова пошукового індексу: в даному випадку індекс є поданням даних, логічна модель якого визначає спосіб обробки і інтерпретації даних і дозволяє здійснювати інформаційний пошук;

- перетворення пошукових запитів в формат, який дозволяє використовувати пошуковий індекс для обчислення функції релевантності запитів і документів та вибірки релевантних запиту документів [7].

2.2 Метод k-середніх

Кластерний аналіз – сукупність багатовимірних статистичних методів класифікації об'єктів за ознаками, які характеризують їх, поділ сукупності

об'єктів на однорідні групи, близькі за визначальними критеріями, виділення об'єктів певної групи [9].

При великій кількості спостережень ієрархічні методи кластерного аналізу не придатні, у таких випадках використовують неієрархічні методи, засновані на поділі, які представляють собою ітеративні методи дроблення вихідної сукупності, у процесі поділу нові кластери формуються до тих пір, поки не буде виконано правило зупинки, така неієрархічна кластеризація полягає в поділі набору даних на певну кількість окремих кластерів, при цьому існує два підходи:

- перший полягає у визначенні меж кластерів як найбільш щільних ділянок в багатовимірному просторі вихідних даних, тобто визначення кластера там, де є велике згущення точок;

- другий підхід полягає в мінімізації міри відмінності об'єктів [10].

Алгоритм k -середніх є найбільш поширеним серед неієрархічних методів, даний алгоритм також званий швидким кластерним аналізом, на відміну від ієрархічних методів, які не вимагають попередніх припущень щодо числа кластерів, для можливості використання вимагає мати гіпотезу про найбільш ймовірну кількість кластерів:

- алгоритм k -середніх будує k кластерів, розташованих на можливо великих відстанях один від одного;

- основний тип задач, які вирішує алгоритм k -середніх, – наявність припущень (гіпотез) щодо числа кластерів, при цьому вони повинні бути різні настільки, наскільки це можливо;

- вибір числа k може базуватися на результатах попередніх досліджень, теоретичних міркуваннях або інтуїції [10].

Метод k -середніх застосовується до масиву значень точок в d -вимірному векторному просторі, таким чином, це кластери набору d -вимірних векторів, $D = \{x_i \mid i = 1, \dots, N\}$, де $x_i \in S$ позначає i -ий об'єкт або точку даних: k -середніх є алгоритмом кластеризації, який розділяє D на k кластерів точок, тобто алгоритм k -середніх об'єднує всі точки даних в D так,

що кожна точка x_j потрапляє в один і тільки один з k кластерів, можна відстежити, яка точка знаходиться в якому кластері, призначивши кожній точці номер кластера, точки з таким же номером кластера знаходяться в одному і тому ж кластері, в той час як точки з різними номерами кластера знаходяться в різних кластерах [11].

Алгоритм діє за принципом мінімізації сумарного квадратичного відхилення точок кластерів від центрів цих кластерів (центроїдів – точок, які є центрами кластерів), в алгоритмах кластеризації угруповання точок відбувається підбором подібних точок самим собі, схожим за більшістю ознак, алгоритм k -середніх використовує міру близькості – Євклідову відстань, що обчислюється за формулою (2.1):

$$V = \sum_{i=1}^k \sum_{x_j \in S_i} (x_j - \mu_i)^2, \quad (2.1)$$

де k – число кластерів;

x_j – кожен вектор, представлений точкою, $i = 1, 2, \dots, k$;

S – отримані кластери;

μ_i – центри мас векторів x_j (центроїди) [11].

Алгоритм k -середніх призводить до мінімізації підсумкового квадрата Євклідової відстані між кожним вектором x_i і подібною точкою кластера μ_j , відповідно рівняння (2.1) є цільовою функцією методу k -середніх, який виконується на основі наступної послідовності кроків:

- випадковим чином вибирається k точок (центроїди) з початкової множини точок;
- використовуючи формулу (2.1) розподіляються точки за кластерами відносно центроїдів;
- знайти нове положення центроїдів, обчисливши центр кожного кластера;

– виконувати попередні два пункти до тих пір, поки центроїди не перестануть змінювати своє положення або до певного порогу зміни положення центроїдів [11].

2.3 Розроблений метод пошуку комп'ютерних ігор

Логіка роботи методу полягає в створенні персоналізованого пошуку, коли замість того, щоб тільки вибирати комп'ютерні ігри, які співпадають з введеним користувачем запитом, або визначати релевантність кожної гри запиту користувача, спочатку з'ясовується, яким комп'ютерним іграм віддає перевагу користувач, а вже за близькістю кожної гри до відповідності вимогам користувача сортуються всі ігри, які відповідають запиту. Це дозволяє виконувати персоналізований пошук, коли результати головним чином направлені на те, щоб користувач першим чином бачив у результатах ті комп'ютерні ігри, які задовольняють його смакам.

На першому етапі виконується кластерний аналіз всіх комп'ютерних ігор на основі використання методу k-середніх. В результаті буде отримано центроїди для всіх кластерів.

В якості ознак екземплярів можна використати:

- жанр;
- платформу;
- вартість;
- розробника (значення кодується);
- рік виходу гри;
- мови;
- кількість гравців;
- кількість гравців онлайн.

На другому етапі забезпечується персоналізація пошуку, для чого мають бути визначені значення ознак, яким віддає перевагу користувач.

Для виконання цього етапу реалізується один з двох наступних варіантів:

- поставити користувачу набір запитань, кожне з яких визначає значення відповідної ознаки, в результаті можна буде знайти точку в n -мірному просторі, розмірність якого відповідає кількості ознак комп'ютерних ігор;

- якщо користувач вже має уподобані ігри, то можна використати множину його уподобань і усереднивши відповідні значення уподобаних ігор, розрахувати координати точки в n -мірному просторі.

Після цього потрібно визначити, до якого кластера ближче за Євклідовою відстанню належить знайдена точка.

На третьому етапі формується запит користувача на пошук ігор. За введеним запитом виконується пошук комп'ютерних ігор, які відповідають запиту. Пошук виконується за входженням введених слів в назву гри. В результаті формується підмножина комп'ютерних ігор, що відповідають запиту.

На четвертому етапі виконується сортування сформованої підмножини за розрахованим значенням Євклідової відстані для визначеної на третьому етапі точки. Відсортована підмножина і є результатом роботи методу.

Відповідно, якщо користувач не задасть сам запит, але виконає пошук, то він отримає перелік ігор, які найбільше будуть відповідати його власним смакам щодо комп'ютерних ігор.

2.4 Висновки за розділом 2

У другому розділі виконано огляд методів інформаційного пошуку, серед яких розглянуто теоретико-множинні моделі, імовірнісні моделі, алгебраїчні моделі та гібридні моделі. Розглянуто метод кластерного аналізу k -середніх.

Розроблено метод пошуку комп'ютерних ігор, який використовує метод k-середніх для розділення комп'ютерних ігор на кластери, визначає прихильність користувача до ігор відповідного кластеру за відповідями на поставлені запитання або за уподобаними комп'ютерними іграми, що дозволяє виконувати персоналізований пошук у відповідності з тими значеннями характеристик комп'ютерних ігор, яким віддає перевагу користувач.

3 ОПИС ПРОГРАМИ

3.1 Вибір мови програмування

Програмне забезпечення, в якому має бути виконано агрегацію комп'ютерних ігор, має бути розроблено у вигляді десктопного застосунку. Логічним є розроблення такого застосунку на одній з мов програмування: C# або C++.

За результатами порівняння мов програмування C# та C++ було знайдено наступні відмінності за наступними критеріями:

- тип мови: C++ – це мова програмування низького рівня та нейтральна до платформи, C# – мова високого рівня;
- компіляція: C++ компілюється до машинного коду, C# компілюється до Common Language Runtime;
- управління пам'яттю: у C++ потрібно керувати пам'яттю вручну, якщо динамічно розподіляється об'єкт, C# автоматично виконує управління пам'яттю;
- множинне спадкування: C++ підтримує множинне спадкування, C# не підтримує множинне спадкування;
- рівень складності: C++ включає більш складні функції, C# не має складних функцій, він має просту ієрархію і досить простий у розумінні;
- вказівник доступу за замовчуванням: у C++ для struct публічний, приватний для класів, приватний в C#;
- платформа: C++ – це мова, яка працює на різноманітних платформах, вона також однаково популярна в системах Unix та Linux, в той же час стандартизований C# рідко раніше використовувався за межами Windows, але зараз за рахунок кросплатформного рішення .NET Core розширив свій вплив і на інші платформи;
- об'єктно-орієнтованість: C++ не є чистою об'єктно-орієнтованою мовою, C# – це суто об'єктно-орієнтована мова;

- перевірка границь: в той час як C++ не підтримує перевірку границь масивів, C# підтримує таку перевірку;
- збір сміття: C++ не підтримує збір сміття, C# підтримує автоматичний збір сміття;
- цикл `foreach`: C++ не підтримує такий цикл, C# підтримує цикл `foreach`;
- використання покажчиків: в C++ можна використовувати покажчики в будь-якому місці програми, у C# покажчик можна використовувати лише в небезпечному режимі;
- галузь використання: C++ широко використовується в іграх, програмування на C# можна використовувати для створення Windows, мобільних та консольних застосунків;
- розмір бінарних файлів: код C++ набагато легший, C# має багато накладних витрат, і перед компіляцією слід включити всі такі бібліотеки;
- тип проєктів: програмісти на C++ зазвичай зосереджуються на застосунках, які працюють безпосередньо з апаратним забезпеченням або потребують кращої продуктивності, ніж інші мови, C# використовується для розробки сучасних застосунків;
- попередження компілятора: C++ дозволяє робити майже все, за умови правильного синтаксису, тому це гнучка мова, але можна за допомогою неї завдати серйозної шкоди операційній системі, C# є високо захищеною мовою, оскільки компілятор видає помилки та попередження на випадок написання коду, який може завдати шкоди;
- результат компіляції: після компіляції код C++ перетворюється на машинний код, після компіляції код C# перетворюється в код проміжної мови;
- оператор `switch`: у C++ в операторі `switch` тестова змінна не може бути рядком, у операторі `switch` в C# тестова змінна може бути або не бути рядком [12]-[14].

Отримані результати порівняння за основними критеріями, які було використано для виконання вибору, було зведено до табл. 3.1.

Таблиця 3.1 – Вибір мови програмування

Показник	C#	C++
Тип мови	мова програмування високого рівня	мова програмування низького рівня
Управління пам'яттю	автоматичне управління пам'яттю	керування пам'яттю вручну
Платформа	може запускатися на різних платформах за рахунок .NET Core	працює на різноманітних платформах
Чиста об'єктно-орієнтованість	+	—
Перевірка границь масивів	+	—
Автоматичний збір сміття	+	—
Цикл foreach	+	—
Тип проєктів	розробки сучасних застосунків	застосунки, які працюють безпосередньо з апаратним забезпеченням або потребують кращої продуктивності

Грунтуючись на описаних критеріях, мовою розробки було обрано мову програмування C#.

3.2 Моделювання функціональних вимог до програмного забезпечення

На основі виконаного огляду програмних аналогів у підрозділі 1.2, було виділено наступні функціональні вимоги до розроблюваного програмного забезпечення агрегації комп'ютерних ігор:

- додавання інформації про комп'ютерну гру;
- пошук комп'ютерних ігор за назвою;
- персоналізований пошук комп'ютерних ігор;
- виставлення власної оцінки за комп'ютерну гру;
- внесення коментаря про комп'ютерну гру;
- перегляд коментарів користувачів про комп'ютерну гру;
- перегляд комп'ютерних ігор за жанром;
- перегляд найкращих комп'ютерних ігор за оцінками користувачів;
- редагування інформації про раніше додану комп'ютерну гру;
- авторизація;
- реєстрація.

Моделювання роботи користувачів з програмним забезпеченням було виконано за допомогою діаграми прецедентів Unified Modeling Language (UML). Діаграми прецедентів є один з п'яти типів діаграм, що застосовуються в UML для моделювання динамічних аспектів системи, діаграми прецедентів грають основну роль в моделюванні поведінки системи, підсистеми або класу, кожна така діаграма показує множину прецедентів, акторів і відносини між ними, діаграми прецедентів застосовуються для моделювання виду системи з точки зору прецедентів (або варіантів використання), найчастіше це передбачає моделювання контексту системи, підсистеми або класу або моделювання вимог, що пред'являються до поведінки зазначених елементів, діаграми прецедентів мають велике значення для візуалізації, документування поведінки елемента, вони полегшують розуміння систем, підсистем або класів, представляючи погляд

ззовні на те, як дані елементи можуть бути використані у відповідному контексті, крім того, такі діаграми важливі для тестування виконуваних систем в процесі прямого проєктування і для розуміння їх внутрішнього устрою при зворотному проєктуванні [15].

На рис. 3.1 приведено діаграму прецедентів UML взаємодії адміністратора з програмою.

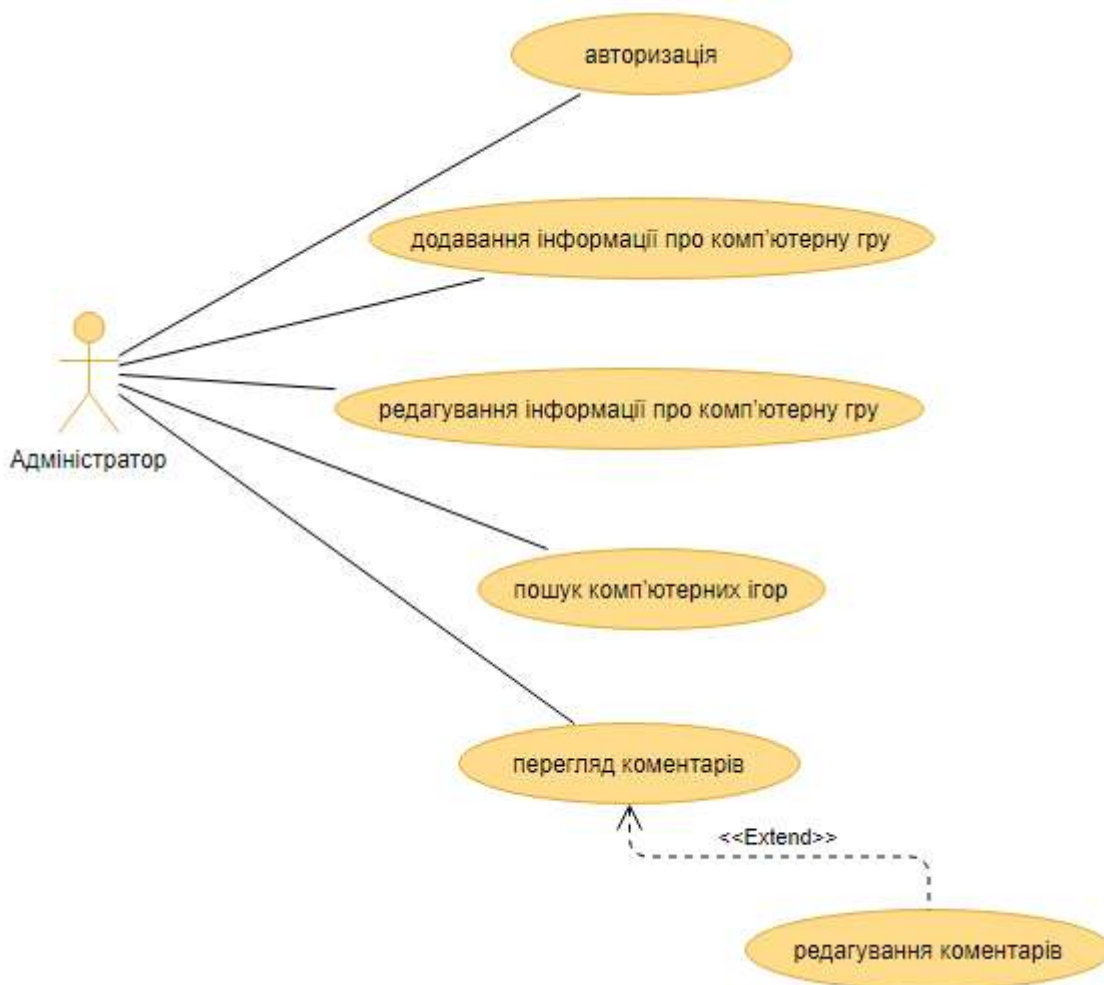


Рисунок 3.1 – Діаграма прецедентів UML взаємодії адміністратора з програмою

На рис. 3.2 приведено діаграму прецедентів UML взаємодії користувача з програмою.

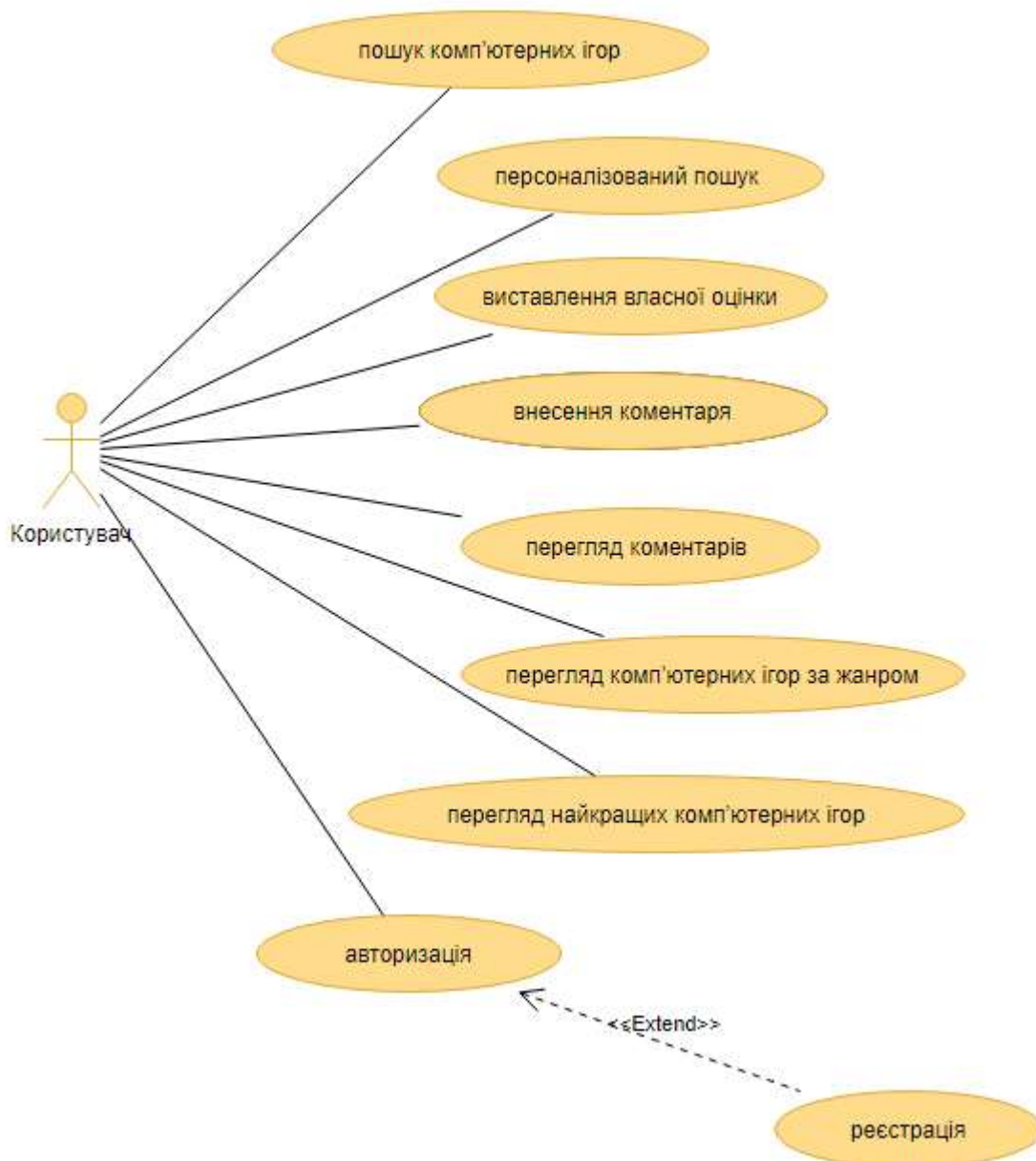


Рисунок 3.2 – Діаграма прецедентів UML взаємодії користувача з програмою

3.3 Проєктування структури бази даних

В якості системи керування базою даних було використано Microsoft SQL Server з екосистеми мови програмування C#, вибір якої було виконано на попередньому етапі.

Таблиці бази даних:

- комп'ютерна гра Game;
- платформа Platform;

- жанр Genre;
- жанри ігор GameGenre, що визначає жанр гри;
- платформи ігор GamePlatform, що визначає для яких платформ присутня версія гри;
- мова гри Language;
- мови ігор GameLanguage, що визначає для яких мов присутня версія гри;
- акаунт Account;
- оцінка Grade;
- коментар Comment;
- запитання Question;
- відповіді Answer;
- відповідь користувача на запитання UserAnswer.

Таблиця Game включає поля:

- ідентифікаційний номер Id;
- назва Title;
- опис Info;
- зовнішнє посилання Url;
- оцінка Rate;
- виробник Developer;
- максимальна кількість гравців Gnum;
- максимальна кількість гравців онлайн Ognum;
- середня вартість Price;
- рік створення Dyear;
- зображення гри Photo;
- кластер, до якого належить гра Cluster.

Таблиця Platform включає поля:

- ідентифікаційний номер Id;
- назва Title.

Таблиця Genre включає поля:

- ідентифікаційний номер Id;
- назва Title.

Таблиця Language включає поля:

- ідентифікаційний номер Id;
- назва Title.

Таблиця GamePlatform включає поля:

- ідентифікаційний номер Id;
- номер гри Game_id;
- номер платформи Platform_id.

Таблиця GameGenre включає поля:

- ідентифікаційний номер Id;
- номер гри Game_id;
- номер жанру Genre_id.

Таблиця GameLanguage включає поля:

- ідентифікаційний номер Id;
- номер гри Game_id;
- номер мови Language_id.

Таблиця Account включає поля:

- ідентифікаційний номер Id;
- логін Login;
- пароль Pass;
- кластер Cluster;
- тип Type (адміністратор або користувач).

Таблиця Grade включає поля:

- ідентифікаційний номер Id;
- номер акаунта Account_id;
- номер гри Game_id;
- оцінка Value;
- дата і час виставлення When.

Структура бази даних приведена на рис. 3.3.

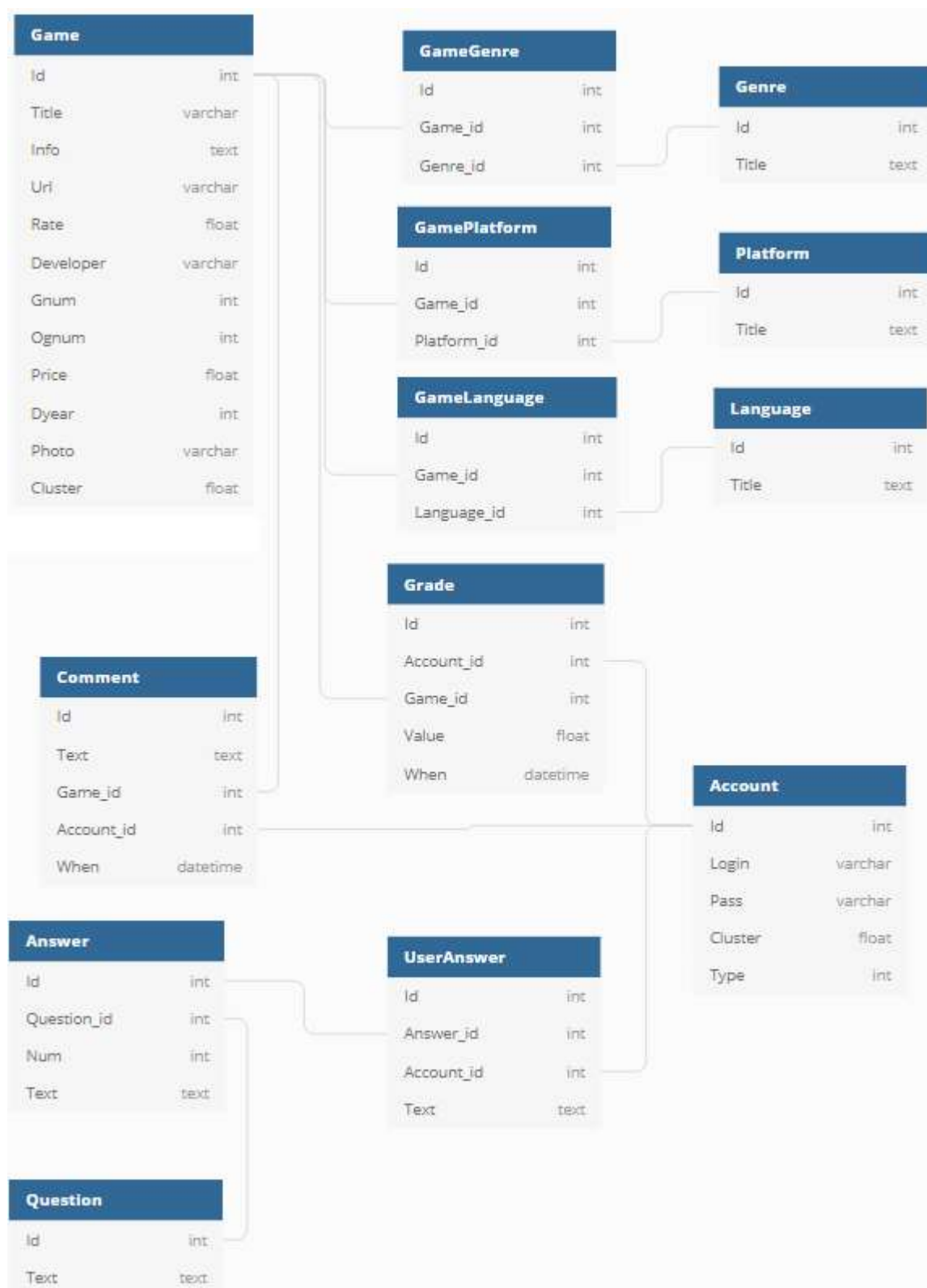


Рисунок 3.3 – Структурна схема бази даних

Таблиця Comment включає поля:

- ідентифікаційний номер Id;
- текст коментаря Text;

- акаунт Account_id;
- дата і час запису When.

Таблиця Question включає поля:

- ідентифікаційний номер Id;
- текст запитання Text.

Таблиця Answer включає поля:

- ідентифікаційний номер Id;
- номер запитання Question_id;
- номер варіанту відповіді Num;
- текст відповіді Text.

Таблиця UserAnswer включає поля:

- ідентифікаційний номер Id;
- номер відповіді Answer_id;
- номер акаунту Account_id;
- текст відповіді Text (для відповідей на запитання без варіантів відповіді).

3.4 Проектування структури застосунку

При проектуванні структури застосунку було використано підхід, який дозволяє повторно використовувати код, що належить до реалізації предметної області. Відповідні класи було виділено в окрему складову – підсистему роботи з даними.

У складі цієї підсистеми основною для зовнішньої взаємодії є модуль роботи з даними, до якого надано окремі інтерфейси доступу для адміністратора IAdmin, користувача IUser та для виконання авторизаційних функцій IAuthorization.

Модуль класів предметної області відповідає за реалізацію класів у відповідності зі структурою бази даних.

Модуль роботи з базою даних забезпечує зв'язування за базою даних та виконання запитів до неї.

Підсистема методу пошуку комп'ютерних ігор реалізує безпосередньо виконання запропонованого в роботі методу за допомогою набору відповідних модулів.

Структуру застосунку приведено у вигляді діаграми компонентів UML на рис. 3.4.

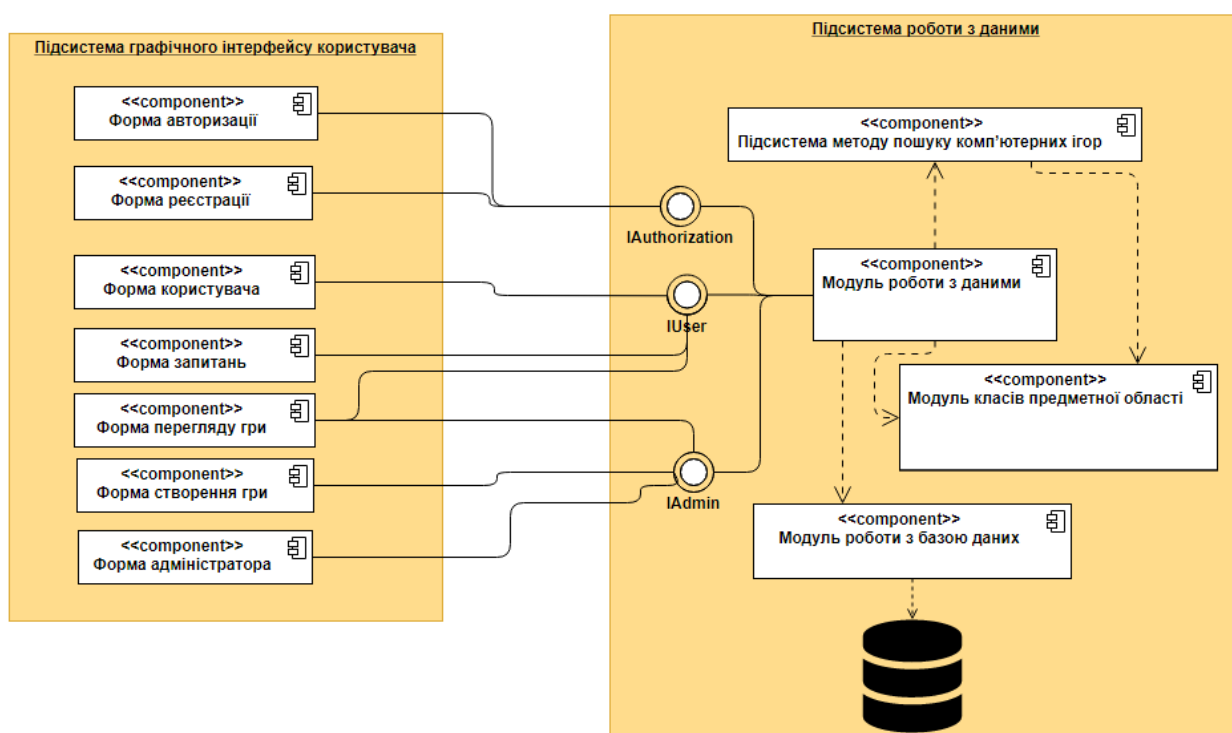


Рисунок 3.4 – Діаграма компонентів UML

Окрім того окремо було виділено складову – підсистему графічного інтерфейсу користувача, що включає набір форм для роботи над функціями, які було описано вище:

- форму авторизації;
- форму реєстрації;
- форму користувача;
- форму адміністратора;
- форму запитань;

- форму перегляду гри;
- форму створення гри.

3.5 Висновки за розділом 3

У третьому розділі виконано вибір мови програмування C# для реалізації програмного забезпечення, оскільки вона є мовою високого рівня, дозволяє виконувати автоматичне управління пам'яттю, може запускатися на різних платформах за рахунок .NET Core, є чистою об'єктно-орієнтованою мовою, дозволяє виконувати перевірку границь масивів, а також створювати широкі сучасні застосунки. Виконано моделювання функціональних вимог до програмного забезпечення у вигляді діаграм прецедентів UML для адміністратора та користувача. Виконано проєктування структури бази даних з вибором Microsoft SQL Server у якості системи керування базою даних та з приведенням структурної схеми бази даних. Виконано проєктування структури застосунку зі створенням діаграми прецедентів UML.

4 РОЗРОБКА ПРОГРАМИ

4.1 Алгоритм функціонування програми

Робота програми розпочинається з виведення форми авторизації, на якій користувач виконує вибір – чи він має на даний момент вже зареєстрований обліковий запис. Якщо має, то він використовує дану форму для авторизації. Якщо не має, то ініціалізується відкриття форми реєстрації, де користувач виконує реєстрацію облікового запису і одразу ініціалізується його авторизація.

Після отримання авторизованого облікового запису виконується визначення його типу.

У випадку авторизації за обліковим записом адміністратора надається можливість виконати одну з дій:

- додавання даних про нову гру;
- редагування даних про існуючу гру;
- пошук ігор, що використовується адміністратором для знаходження ігор з метою подальшого редагування даних про них або редагування коментарів;
- перегляд даних гри, звідки адміністратор може зокрема переглянути набір коментарів і виконати за потребу редагування існуючих.

У випадку авторизації за обліковим записом користувача надається можливість виконати одну з дій:

- пошук ігор;
- персоналізований пошук ігор;
- перегляд переліку найкращих ігор;
- перегляд ігор за жанром;
- вибір однієї з даних з виведеного переліку, після чого можна встановити власну оцінку грі, переглянути її дані, коментарі до неї, а також написати власний коментар.

На рис. 4.1 представлена відповідна функціональна схема програми.

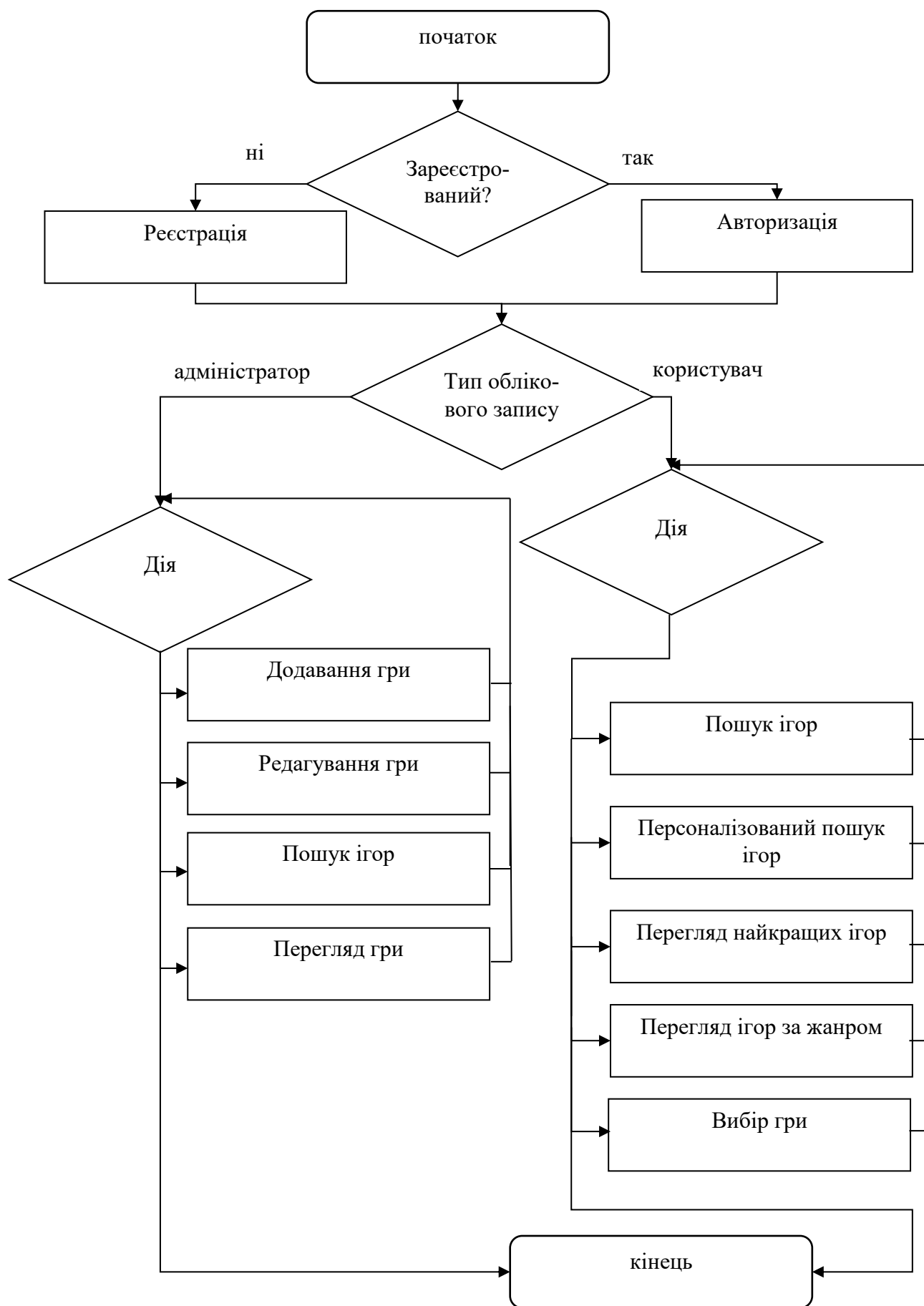


Рисунок 4.1 – Функціональна схема програми

4.2 Опис розроблених модулів

Для реалізації модулю роботи з даними було створено модуль `DataProcessorModule.cs`.

Даний модуль надає доступ до методів класу за допомогою наступних інтерфейсів:

- `IAuthorization` – інтерфейс авторизації облікового запису;
- `IAdmin` – інтерфейс доступу адміністратора до модуля роботи з даними;
- `IUser` – інтерфейс доступу користувача до модуля роботи з даними.

Дані інтерфейси забезпечують доступ до класу `DataProcessorModule`, який є основою даного модуля.

Інтерфейс `IAuthorization` надає доступ до наступних методів:

- `int Auth(string login, string pass)` – метод виконання аутентифікації облікового запису користувача з логіном `login` та паролем `pass`, який в результаті роботи повертає ідентифікаційний номер аутентифікованого облікового запису;
- `int GetAccountType(int num)` – метод визначення типу облікового запису за ідентифікаційним номером;
- `int Registrer(string login, string pass)` – метод виконання реєстрації облікового запису користувача з логіном `login` та паролем `pass`, який в результаті роботи повертає ідентифікаційний номер новоствореного облікового запису, при цьому реалізується перевірка наявності облікового запису з заданим логіном і якщо він вже наявний, то повертається значення - 1;
- `List<Account> getAllAccounts()` – метод, який дозволяє отримати перелік усіх облікових записів.

Інтерфейс `IAdmin` надає доступ до наступних методів:

– `int FillAllGames ()` – метод, який виконує завантаження даних про ігри з бази даних і забезпечує заповнення переліку `allgames`, який є одним з атрибутів класу `DataProcessorModule` даними;

– `bool AddGame(string title, string info[100], string url, string developer, int gnum, int ognum, float price, string rating, int dyear, string photo, List<String> genres, List<String> platforms, List<String> languages)` – виконує створення нової гри з назвою `title`, описом `info`, адресою для доступу `url`, розробником `developer`, максимальною кількістю гравців `gnum`, максимальною кількістю гравців онлайн `ognum`, середньою вартістю `price`, віковим рейтингом `rating`, роком створення `dyear`, зображенням гри `photo`, переліком жанрів `genres`, переліком платформ `platforms` та переліком мов `languages`: при цьому створюється новий запис в переліку `allgames`, який є одним з атрибутів класу `DataProcessorModule` даними, та ініціалізується додавання нового запису до бази даних;

– `List<Games> getAllGames()` – метод, який повертає повний перелік ігор, що міститься в переліку `allgames`, який є одним з атрибутів класу `DataProcessorModule` даними;

– `bool detectGameClusters()` – метод, який виконує визначення кластерів, до яких належать ігри переліку `allgames`, при цьому ініціалізуючи кластеризацію з класу `GonMethod` шляхом виклику методу `Clusterize`;

– `bool EditGame(int num, string title, string info[100], string url, float rate, string developer, int gnum, int ognum, float price, string rating, int dyear, string photo, List<String> genres, List<String> platforms, List<String> languages)` – виконує редагування гри з номером `num`, змінюючи її назву на `title`, опис на `info`, адресу для доступу на `url`, розробника на `developer`, максимальну кількість гравців на `gnum`, максимальну кількість гравців онлайн на `ognum`, середню вартість на `price`, віковий рейтинг на `rating`, рік створення на `dyear`, шлях до зображення гри на `photo`, перелік жанрів на `genres`, перелік платформ на `platforms` та перелік мов на `languages`: при цьому окрім зміни даних відповідної гри в переліку `allgames`, який є одним з атрибутів класу

DataProcessorModule даними, ініціалізується оновлення відповідного запису бази даних, якщо будь-яке з переданих значень є порожнім(порожній рядок або нульове значення для відповідних типів), то зміна значення відповідного параметра не відбувається;

- List<Game> SearchGame(string namegame) – метод, який реалізує пошук ігор, які містять в назві фразу namegame, повертаючи перелік таких ігор в результаті своєї роботи;

- List<Comment> getGameComments(int gamenum) метод, який повертає перелік всіх коментарів, які були залишені для гри з номером gamenum;

- bool EditComment (int commnum, int accnum, string text[100], DateTime when) – метод, який виконує редагування коментаря з номером commnum, який було створено користувачем з номером облікового запису accnum, змінюючи його текст на text та зі збереженням моменту часу when в якості часу його додавання;

- bool calcAllGamesRate() – метод, який виконує переобчислення рейтингу всіх ігор на основі розрахунку середньої оцінки за всіма оцінками, які було виставлено грі;

- bool calcAllGamesRate(int gamenum) – метод, який виконує переобчислення рейтингу гри gamenum на основі розрахунку середньої оцінки за всіма оцінками, які було виставлено грі.

Інтерфейс IUser надає доступ до наступних методів:

- int GetAccountCluster(int num) – метод, який повертає номер кластеру для користувача з номером облікового запису num;

- int FillAllGames () – метод, який виконує завантаження даних про ігри з бази даних і забезпечує заповнення переліку allgames, який є одним з атрибутів класу DataProcessorModule даними;

- List<Game> SearchGame(string namegame) – метод, який реалізує пошук ігор, які містять в назві фразу namegame, повертаючи перелік таких ігор в результаті своєї роботи;

- `List<Game> PersonalSearchGame(int accnum, string namegame)` – метод, який виконує персоналізований пошук (з врахуванням кластерів приналежності ігор) для користувача з номером облікового запису `accnum` гри, що містить фразу `namegame` в назві, повертаючи перелік таких ігор в результаті своєї роботи;
- `List<Game> GameGenreBest (string genre)` – метод, який дозволяє отримати перелік ігор, що належать жанру `genre`, відсортованих за рейтингом, при цьому перед початком сортування забезпечується оновлення даних рейтингу;
- `bool AddComment (int gamenum, int accnum, string text[100], DateTime when)` – метод, який виконує додавання коментаря від користувача з номером облікового запису `accnum` до гри з номером `gamenum` з текстом `text` та зі збереженням моменту часу `when` в якості часу його додавання;
- `bool detectUserCluster(int accnum, List<string> answers)` – метод, який ініціалізує визначення кластеру, до якого належать відповіді користувача на запитання `answers` шляхом звернення до методу `GetCluster` класу `GonMethod`, що забезпечує реалізацію запропонованого методу, за результатами відбувається внесення даних як в перелік `allaccs`, так і в базу даних;
- `bool AddGrade (int gamenum, int accnum, float valuee, DateTime when)` – метод, який виконує додавання оцінки `valuee` від користувача з номером облікового запису `accnum` до гри з номером `gamenum` зі збереженням моменту часу `when` в якості часу його додавання;
- `List<Comment> getGameComments(int gamenum)` – метод, який повертає перелік всіх коментарів, які були залишені для гри з номером `gamenum`;
- `bool calcAllGamesRate()` – метод, який виконує переобчислення рейтингу всіх ігор на основі розрахунку середньої оцінки за всіма оцінками, які було виставлено гри;

- `bool calcAllGamesRate(int gamenum)` – метод, який виконує переобчислення рейтингу гри `gamenum` на основі розрахунку середньої оцінки за всіма оцінками, які було виставлено грі;
- `List<Game> BestGames()` – метод, який дозволяє отримати перелік ігор, відсортованих за рейтингом, при цьому перед початком сортування забезпечується оновлення даних рейтингу.

4.3 Висновки за розділом 4

У четвертому розділі описано розроблені класи для реалізації структури програмного забезпечення, проектування якої було виконано в розділі 3. Приведено алгоритм функціонування програмного забезпечення з наведенням відповідної схеми.

5 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

5.1 Призначення програми

Програмне забезпечення призначене для створення платформи для агрегації даних про різні комп'ютерні ігри. Дана програмна платформа забезпечує підбір комп'ютерних ігор за перевагами користувача у світі комп'ютерних ігор та надання йому інформації про відібрані ігри з можливістю наступного переходу на зовнішні ресурси.

5.2 Умови виконання програми

Для роботи програмного забезпечення на серверній стороні має забезпечуватися наступна мінімальна конфігурація апаратно-програмного забезпечення комп'ютера:

- двоядерний процесор з частотою не менше 2,4 ГГц;
- оперативна пам'ять – не менше 4 Гб;
- наявність не менше 2 Гб вільного простору на жорсткому диску.

Для використання програмного забезпечення необхідно мати персональний комп'ютер чи ноутбук з інстальованою операційною системою лінійки Windows, версії не нижче від Windows 7.

Для розробки необхідно використовувати середовище Visual Studio версії не нижче ніж 2017.

5.3 Виконання програми

Після запуску програми користувачу відкривається форма авторизації, на якій має бути введено коректні дані існуючого в системі облікового запису.

Після авторизації відкривається форма для роботи з програмою. На рис. 5.1 приведена форма адміністратора.

Дослідження та програмна реалізація методів пошуку комп'ютерних ігор

Власна
Спільнота
Геймерів

АДМІНІСТРАТИВНА ПАНЕЛЬ

ШУКАТИ

Результати

Номер	Назва	Виробник	Рік	Рейтинг
0	Doom Eternal	id Software	2020	9.3
1	Genshin Impact	miHoYo Limited	2020	8.7
2	Ace Combat 7: Skies Unknown	Project Aces	2019	8.4
3	FIFA 21	EA Sports	2020	9.2
4	Dota 2	Valve Corporation	2013	9.4
5	Sid Meier's Civilization VI	Firaxis Games	2016	8.9
6	Ring of Elysium	Aurora Studio	2018	7.1
7	Fallout 4	Bethesda Game S...	2015	8.2
8	NieR: Automata	PlatinumGamess	2017	8.6
9	The Witcher 3: Wild Hunt	CD Projekt RED	2015	9.4
10	Temtem	Crema	2020	6.2

Коментарі

27.11.2020 checker

Ідея гри була хороша, реалізовано так собі

ВПЕРЕД

ЗБЕРЕГТИ

ВІДКРИТИ

РЕДАГУВАТИ

СТВОРИТИ

Рисунок 5.1 – Форма адміністратора

Форма адміністратора містить поле для пошуку, куди можна вводити назву або частину назви комп'ютерної гри, а далі потрібно натиснути на кнопку “Шукати”, результати пошуку відображаються в таблиці “Результати”. Якщо обрати гру з таблиці, то в правому полі відобразиться один з коментарів. Між коментарями можна переходити за допомогою кнопок “Назад” і “Вперед” (у випадку рис. 5.1 кнопка “Назад” деактивована, оскільки це перший коментар з переліку).

Адміністратор може тут же редагувати коментар, а для збереження має натиснути на кнопку “Зберегти”.

Якщо обрати одну з ігор з таблиці, то можна за допомогою кнопки “Відкрити” перейти до детального перегляду інформації про гру або за допомогою кнопки “Редагувати” перейти до редагування даних обраної гри.

За допомогою кнопки “Створити” можна відкрити форму додавання гри (рис. 5.2).

Дослідження та програмна реалізація методів пошуку комп'ютерних ігор

Власна
Спільнота
Геймерів

СТВОРЕННЯ НОВОГО ЗАПИСУ ПРО КОМП'ЮТЕРНУ ГРУ

Назва

Зовнішнє посилання

Виробник

Максимальна кількість гравців

Максимальна кількість гравців онлайн

Середня вартість

Рік створення

Зображення **ОБРАТИ**

Опис

Мова **ДОДАТИ** **ВИЛУЧИТИ**

Жанр **ДОДАТИ** **ВИЛУЧИТИ**

Платформа **ДОДАТИ** **ВИЛУЧИТИ**

СТВОРИТИ **СКАСУВАТИ**

Рисунок 5.2 – Форма додавання гри

Форма додавання гри дозволяє задати текстом всі параметри гри, обрати зображення (шлях до фалу додається автоматично після вибору з провідника за допомогою кнопки “Обрати”). Для підтвердження збереження гри потрібно натиснути на кнопку “Створити”. Кнопка “Скасувати” дозволяє вийти без збереження.

Таблиці “Мова”, “Жанр”, “Платформа” дозволяють обрати мови інтерфейсу гри, жанри та платформи, для яких доступні версії гри. Для цього потрібно обрати з випадаючого переліку відповідний варіант та натиснути на кнопку “Додати” для додавання жанру, мови або платформи. Результат з’явиться в таблиці. За допомогою кнопки “Вилучити” можна навпаки прибрати вибраний жанр з таблиці.

Приклад заповнення даних про комп’ютерну гру наведено на рис. 5.3.

Дослідження та програмна реалізація методів пошуку комп'ютерних ігор

Власна
Спільнота
Геймерів

СТВОРЕННЯ НОВОГО ЗАПИСУ ПРО КОМП'ЮТЕРНУ ГРУ

Назва: Spider-Man

Зовнішнє посилання: <https://insomniac.games/game/spider-man-ps4/>

Виробник: Insomniac Games

Максимальна кількість гравців: 1

Максимальна кількість гравців онлайн: 1

Середня вартість: 1135

Рік створення: 2018

Зображення: d:/images/spiderman.jpg

ОБРАТИ

Мова: англійська, українська

Жанр: бойовик

ДОДАТИ, ВИЛУЧИТИ

Платформа: PlayStation, Xbox One

ДОДАТИ, ВИЛУЧИТИ

Опис: Spider-Man - це action-adventure від третьої особи з відкритим світом, події якого розгортатимуться в Нью-Йорку. Гравцям належить використовувати вміння Людини-павука, такі як польоти на павутині і прилипання до стін. У грі також є нововведення, які ще не з'являлися за всі ігри про Людину-павука. Одними з таких нововведень стануть паркур і використання оточення під час бою. Так само в грі є RPG складова, у міру отримання досвіду, герой буде відкривати нові навички і механіка бою буде істотно змінюватися. Арсенал Людини-павука буде складатися з «павукових гаджетів» - нових видів павутини, дронів і бомб. По ходу

СТВОРИТИ, СКАСУВАТИ

Рисунок 5.3 – Внесення даних про нову комп'ютерну гру

Якщо в програмі авторизувався звичайний користувач, то для нього відкривається відповідна форма (рис. 5.4).

Форма користувача також містить поле для введення пошукового запиту та таблицю для виведення результатів пошуку.

При цьому дана форма має набір додаткових налаштувань:

- якщо обрати з випадаючого переліку жанр гри, то результати будуть відображатися тільки для заданого жанру;
- якщо обрати прапорець “рейтинг”, то результати будуть відсортовані за рейтингом (якщо натиснути на кнопку “Шукати” без введення пошукового запиту, то відобразяться ігри за рейтингом).

Користувач може формувати потрібний йому набір ігор, звужуючи весь доступний перелік таким чином.

Дослідження та програмна реалізація методів пошуку комп'ютерних ігор

Власна
Спільнота
Геймерів

FIFA

Жанр ☐ персоналізований пошук ☐ рейтинг **ШУКАТИ**

Результати

Номер	Назва	Виробник	Рік	Рейтинг
3	FIFA 21	EA Sports	2020	9.2
37	FIFA 20	EA Sports	2019	9.1

ІНДИВІДУАЛІЗАЦІЯ **ВІДКРИТИ**

Рисунок 5.4 – Форма користувача

Якщо натиснути на прапорець “персоналізований пошук”, то під час виконання пошуку буде виведено ті ігри, що належать кластеру, який обрав для себе користувач за відповідями на запитання (рис. 5.5). У випадку рис. 5.5 за словом “wild” було знайдено лише обмежений набір ігор, не включаючи гру Відьмак 3, яку було показано в таблиці для адміністратора, хоча вона в назві також містить слово wild. Це пов’язано з тим, що у

відповідях на питання користувач обрав, що любить полювання, але ігри жанру RPG здаються йому нудними.

Дослідження та програмна реалізація методів пошуку комп'ютерних ігор

Власна
Спільнота
Геймерів

wild

Жанр ☒ персоналізований пошук ☐ рейтинг **ШУКАТИ**

Результати

Номер	Назва	Виробник	Рік	Рейтинг
102	theHunter: Call of the Wild	Expansive Worlds AB	2017	8.5
93	Tom Clancy's Ghost Recon Wil...	Ubisoft Entertainment ...	2017	8.9

ІНДИВІДУАЛІЗАЦІЯ **ВІДКРИТИ**

Рисунок 5.5 – Персоналізований пошук

Приклад визначення відповідей на запитання користувачем для забезпечення персоналізації наведено на рис. 5.6 (відкривається за натиснення на кнопку “Індивідуалізація”). Потрібно обрати одну з відповідей

на кожне запитання, після чого натиснути на кнопку “Вперед” для переходу до наступного запитання, або “Назад” для повернення до попереднього запитання, якщо необхідно змінити варіант відповіді.

Дослідження та програмна реалізація методів пошуку комп'ютерних ігор

Власна
Спільнота
Геймерів

ПЕРСОНАЛІЗАЦІЯ ПОШУКУ

3. Чи любите ви полювання?

- ☐ В будь-якому вигляді і якомога більше
- ☐ Люблю полювання, але тільки не на людей
- ☐ Іноді
- ☐ Можливо, але це не мій інтерес
- ☐ Виключно не сприймаю

НАЗАД ВПЕРЕД

Рисунок 5.6 – Персоналізація пошуку

Після останньої відповіді активується кнопка “Завершити”.

На рис. 5.7 приведено приклад форми, що відкривається за кнопкою “Відкрити”. Дана форма відображається і для користувача, і для адміністратора. Але для адміністратора не доступне поле “Коментар”, тобто він не може звідси ввести власний коментар (опубліковані коментарі доступні звідси тільки для перегляду).

Також адміністратор не може виставляти оцінки іграм, тому випадючий перелік з оцінкою за 10-бальною шкалою та кнопка “Оцінити” для нього також буде неактивною.

Дослідження та програмна реалізація методів пошуку комп'ютерних ігор

Власна
Спільнота
Геймерів

ІНФОРМАЦІЯ ПРО ГРУ FIFA 20

Назва: FIFA 20
Рейтинг: 9.1

Жанр: спортивний симулятор
Адреса: <https://www.ea.com/games/fifa/fifa-20>
Виробник: EA Sports
Платформи: Windows, PlayStation 4, Xbox One 4

Максимальна кількість гравців: 2
Максимальна кількість гравців онлайн: 2

Середня вартість: 700 грн.
Рік створення: 2019

Мови: англійська, французька, іспанська, португальська, арабська, японська, китайська, голандська, норвезька, польська, шведська, російська, данська та чеська.

Опис:
EA SPORTS FIFA 20 на базі технології Frostbite демонструє дві сторони найпопулярнішої гри в світі - велич професійних футбольних змагань і новий режим автентичного вуличного футболу в EA Sports Volta. FIFA 20 демонструє безліч нововведень: система «футбольного інтелекту» забезпечує безпрецедентний реалізм ігрового процесу, режим FIFA UltimateTeam дарує ще більше можливостей для створення команди своєї мрії, а EA Sports Volta повертає гру на вулиці, дозволяючи насолодитися самим автентичним виглядом неформального футболу!



Коментар

ДОДАТИ

Оцінка

ОЦІНИТИ

Опубліковані коментарі

gamertoplevel 25.11.2020
Нема нічого краще за PES!!!

bigdragon 20.11.2020
З задоволенням граю вже більше 10 років. Улюблена страва ввечері!

azar 18.11.2020
Граю краще ніж зав'язую шнурки

Рисунок 5.7 – Перегляд інформації про комп'ютерну гру

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

Програмне забезпечення призначено для виконання пошуку комп'ютерних ігор. Замість впровадження переліків рекомендацій ігор буде використовуватися персоналізований пошук. До впровадження даного програмного забезпечення дані функції виконувались експертом 500 годин на рік. Використання даної програми дозволить скоротити експерта.

У розробці беруть участь програміст із заробітною платою 8000 грн. в місяць і консультант з заробітною платою 8500 грн. у місяць. На розробку проєкту було витрачено 2 місяці. Розробка розпочалася першого жовтня і повинна бути виконана до 30 листопада 2020 року.

6.1 Планування розробки програмного продукту

Весь комплекс розробки програмного продукту можна розділити на етапи: для кожного етапу вказуються трудомісткість, кількість виконавців і тривалість робіт [16]. Інформація з характеристики робіт по розробці програми зведена у табл. 6.1.

Таблиця 6.1 – Характеристика робіт по розробці програми

Найменування робіт	Трудомісткість		Виконавці	Тривалість, днів
	люд.-дні	%		
1	2	3	4	5
Аналіз програмного забезпечення	10	16,4	Програміст Консультант	5
Визначення вимог до програмного продукту	10	16,4	Програміст Консультант	5

Продовження таблиці 6.1

1	2	3	4	5
Проектування структури програми	20	32,8	Програміст Консультант	10
Розробка схеми функціонування програми	5	8,2	Програміст	5
Створення програмного коду	10	16,4	Програміст	10
Тестування і налагодження програмного продукту	3	4,9	Програміст	3
Складання програмної документації	3	4,9	Програміст	3
Разом	61	100		41

За даними табл. 6.1 складається графік планування розробки програмного продукту.

Графік планування по розробці програмного забезпечення представлений в табл. 6.2.

Чорним показано роботу програміста, а сірим – відповідно роботу консультанта.

Таблиця 6.2 – Характеристика робіт та графік планування по розробці програми

Найменування робіт	Календарний період, дні								
	05.10-09.10	12.10-16.10	19.10-23.10	26.10-30.10	02.11-06.11	09.11-13.11	16.11-20.11	23.11-27.11	30.11-30.11
Аналіз програмного забезпечення	■								
Визначення вимог до програмного продукту		■							
Проектування структури програми			■	■					
Розробка схеми функціонування програми					■				
Створення програмного коду						■	■		
Тестування і налагодження програмного продукту								■	
Складання програмної документації									■

6.2 Визначення витрат на розробку програми

Для визначення витрат на розробку програми складають калькуляцію кошторисної вартості робіт, дана калькуляція включає наступні статті, що розглядаються в наступних пунктах:

- основна заробітна плата;
- додаткова заробітна плата;
- відрахування на єдиний соціальний внесок;

– матеріали та комплектуючі вироби за вирахуванням вартості зворотних відходів;

– витрати на спеціальне обладнання;

– інші прямі витрати;

– накладні витрати [16].

6.2.1 Основна заробітна плата

Витрати за цією статтею складаються з планового фонду зарплати всіх категорій працівників, зайнятих в розробці програми [16]. Розрахунок зарплати на підставі даних про трудомісткості, представлений в табл. 6.3.

Таблиця 6.3 – Основна заробітна плата

Посада виконавця	Чисельність, чол.	Місячний оклад	Кількість місяців роботи	Сума зарплатні, грн
Програміст	1	8000	2	16000
Консультант	1	8500	1	8500
Разом	2			24500

6.2.2 Розрахунок додаткової заробітної плати

Додаткову заробітну плату $ЗП_{\text{дод}}$ приймають рівною 10% від основної зарплати $ЗП_{\text{осн}}$. Вона розраховується за формулою (6.1):

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 0,1. \quad [16]. \quad (6.1)$$

Підставив величину основної заробітної плати в дану формулу отримаємо:

$$ЗП_{\text{дод}} = 24500 \cdot 0,1 = 2450 \text{ грн.}$$

6.2.3 Відрахування на єдиний соціальний внесок

Ці відрахування визначають у відсотковому відношенні від сум основної та додаткової зарплати з урахуванням премій і доплат. Відрахування вираховуються за формулою (6.2):

$$\begin{aligned} \text{ВТ}_{\text{соц}} &= (\text{ЗП}_{\text{осн}} + \text{ЗП}_{\text{доп}}) \cdot 0,22, [16], \\ \text{ВТ}_{\text{соц}} &= (24500 + 2450) \cdot 0,22 = 5929 \text{ грн.} \end{aligned} \quad (6.2)$$

6.2.4 Визначення витрат на матеріали

Використовується 2 найменування матеріалів:, картридж – 100 грн, папір – 120 грн. Витрати на матеріали розраховують за формулою (6.3):

$$M = \sum_{i=1}^n (\text{Ц}_i \cdot N_i \cdot (1 + K_{\text{м.з.}}) - \text{Ц}_{\text{іо}} \cdot N_{\text{іо}}), \quad (6.3)$$

де M – витрати на матеріали, покупні напівфабрикати і комплектуючі вироби, грн.;

$K_{\text{м.з.}}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

Ц_i – ціна i -го найменування матеріалу, напівфабрикату і комплектуючого, грн.;

N_i – потреба в i -му матеріалі, напівфабрикаті і комплектуючому;

$\text{Ц}_{\text{іо}}$ – ціна зворотних відходів i -го найменування матеріалу, грн.;

$N_{\text{іо}}$ – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих [16].

Розрахуємо:

$$C_{io} = 0, N_{io} = 0, K_{m.з} = 0,05,$$

$$M = (1 + 0,05) \cdot (100 + 120) = 231 \text{ грн.}$$

Разом витрати на матеріали становлять 231 гривні.

6.2.5 Витрати на спеціальне обладнання

У цю статтю входять витрати на придбання, транспортування, монтаж і налагодження нестандартного обладнання: практично, в даному випадку, в цій статті враховуються витрати на оплату машинного часу електронно-обчислювальної машини (ЕОМ) для написання і налагодження програмних засобів, для чого необхідно скласти кошторис «витрат на утримання і експлуатацію устаткування», виходячи з якої визначиться вартість однієї машино-години роботи персонального комп'ютера (ПК), після множення якої на машинний час, який пішов на написання і налагодження програми, отримаємо витрати на оплату машинного часу [16].

Амортизаційні відрахування визначають за формулою (6.4):

$$A = \frac{\Phi_6 \cdot H_a}{100}, \quad (6.4)$$

де Φ_6 – балансова вартість обчислювальної техніки, грн.;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, % [16].

Балансова вартість обчислювальної техніки становить 40000 грн.
Отримуємо:

$$A = 40000 \cdot 0,25 = 10000 \text{ грн.}$$

Статтю "Експлуатація обладнання" розраховують підсумовуванням витрат на силову електроенергію і допоміжні матеріали, витрати на електроенергію визначають за формулою (6.5):

$$C_e = N_n \cdot \Phi_{ef} \cdot K_{зв} \cdot K_{зм} \cdot C_e, \quad (6.5)$$

де N_n – номінальна потужність ЕОМ, кВт;

Φ_{ef} – річний ефективний фонд часу роботи ЕОМ, машино-год;

$K_{зв}$ – середній коефіцієнт завантаження за часом;

$K_{зм}$ – коефіцієнт завантаження по потужності;

C_e – ціна одного кВт · год електроенергії, грн. / (кВт · год) [16].

Номінальна Потужність ЕОМ – 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ ставити 1800 годин. Середні коефіцієнти завантаження за часом та за потужністю рівні відповідно 0,9 и 0,6. Ціна однієї кіловат-години електроенергії ставить 2,11 грн. Отримуємо:

$$C_e = 0,2 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 410,184 \text{ грн.}$$

Витрати за статтею "Заробітна плата обслуговуючих робітників та відрахування на соціальне страхування в інші фонди" визначають за формулою (6.6):

$$ЗП_{обсл} = \Phi ЗП_z \cdot (1 + K_{відрах.}) \cdot \frac{t_{обсл}}{\Phi_{ef.обсл}}, \quad (6.6)$$

де $\Phi ЗП_z$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн.;

$K_{відрах.}$ – коефіцієнт, що враховує відрахування на соціальне страхування і в інші фонди;

$t_{обсл}$ – час протягом року, необхідне на технічне обслуговування ЕОМ, ч / рік;

$\Phi_{эф.обсл}$ – річний ефективний фонд часу обслуговуючого персоналу, год./рік [16].

Заробітна плата обслуговуючого персоналу = 3723 грн./місяць:

$$\Phi ЗП_z = 3723 \cdot 12 = 44676 \text{ грн.},$$

$$ЗП_{обсл} = 44676 \cdot (1+0,22) \cdot 12/1750 = 373,75 \text{ грн.}$$

Суму витрат за статтею "Поточний ремонт обладнання" приймають укрупнено рівною 3% від балансової вартості обладнання. Вона становить 1200 грн.

Інші витрати приймають рівними 3-5% від суми всіх попередніх статей витрат на утримання і експлуатацію обладнання. Сума всіх попередніх статей дорівнює 10273, 5% від суми становлять 513,65 грн.

Кошторис витрат на утримання і експлуатацію устаткування наведений в табл. 6.4.

Таблиця 6.4 – Кошторис витрат на утримання і експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація обладнання	10000
Заробітна плата основна і додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	373,75
Експлуатація обладнання / крім витрат на поточний ремонт	410,184
Поточний ремонт обладнання	1200
Інші витрати	513,65
Разом	12496,93

Витрати на оплату машинного часу ЕОМ для написання і налагодження програмних засобів визначаються за формулою (6.7):

$$C_{\text{мо}} = P_{\text{екс}} \cdot t_{\text{мо}}, \quad (6.7)$$

де, $C_{\text{мо}}$ – витрати на оплату машинного часу, грн.;

$P_{\text{екс}}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн. / машино-год.;

$t_{\text{мо}}$ – машинний час цифрової ЕОМ для написання і налагодження даного програмного продукту, машино-год. [16].

Розрахуємо:

$$P_{\text{екс}} = 12496,93/1800=6.94 \text{ грн. машино-год.}$$

ЕОМ експлуатується протягом 45 днів в одну зміну, що складає в сумі 360 годин. Отримуємо:

$$C_{\text{мо}}=6.94 \cdot 360=2498,4 \text{ грн.}$$

6.2.6 Розрахунок накладних витрат

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т.д.), утримання та експлуатацію будівель: накладні витрати включаються до вартості розробки програми непрямым шляхом – у відсотках до основної заробітної плати розробників [16]. У даному випадку накладні витрати становлять 50% до основної заробітної плати розробників, що складає 12250 грн.

Калькуляція кошторисної вартості робіт з розробки програми наведена в табл. 6.5.

Таблиця 6.5 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн.	Питома вага до підсумку, %
Основна заробітна плата	24500	51,19
Додаткова заробітна плата	2450	5,12
Єдині соціальні виплати	5929	12,39
Матеріали і комплектуючі	231	0,48
Витрати на спеціальне обладнання	2498,4	5,22
Накладні витрати	12250	25,60
Разом	47858,4	100

6.3 Визначення економічної ефективності програмного продукту

Річну економію від впровадження програми, що дозволяє знизити витрати машинного часу ЕОМ для рішення певного завдання в порівнянні із програмою, що застосовувалася раніше, визначають за формулою (6.8):

$$E = (T_{M_1} - T_{M_2}) \cdot B_{\text{екс.кгз}} + E_{\text{н}}^a \cdot \frac{\Phi_{\text{б.кгз}} \cdot (T_{M_1} - T_{M_2})}{\Phi_{\text{еф.кгз}}} - \frac{S_{\text{рп}}}{T_{\text{с}}} + E_{\text{зп}}, \quad (6.8)$$

де T_{M_1}, T_{M_2} – машинний час, необхідний для рішення поставлених завдань, відповідно в старому та у новому варіантах, машино-год./рік;

$B_{\text{екс}}$ – експлуатаційні витрати, що доводяться на 1 год машинного часу ЕОМ, грн/ машино-год.;

$E_{\text{н}}^a$ – нормативний коефіцієнт ефективності капітальних вкладень у засоби автоматизації;

$\Phi_{\text{б.кгз}}$ – балансова вартість ЕОМ, грн.;

$\Phi_{\text{еф.кгз}}$ – річний ефективний фонд часу роботи ЕОМ, машино-год.;

$S_{\text{рп}}$ – сумарні витрати на розробку програми, грн.;

$T_{\text{с}}$ – термін служби впроваджуваної програми до її морального зношування, років;

$E_{\text{зп}}$ – економія на заробітній платні [16].

До впровадження даного програмного забезпечення дані функції виконувались аналітиком 500 годин на рік (оплата аналітика за годину 60 гривень). Отримані дані використовувалися далі маркетингологом для планування маркетингової політики. Використання даного програмного забезпечення дозволить скоротити аналітика, а маркетинголог буде користуватися отриманими даними від програмного забезпечення.

$$E = (500 - 0) \cdot 6,94 + 0,33 \cdot (40000 \cdot (500 - 0)) / 1800 - 47858,4/3 + \\ + 500 \cdot (1 + 0,22) \cdot 1,1 \cdot 60 = 31443,87 \text{ грн.}$$

Коефіцієнт ефективності капітальних вкладень E и строк їхньої окупності $T_{\text{ок}}$ розраховуються за формулами (6.9)-(6.10):

$$E\phi = \frac{E}{S_{\text{рп}}}, \quad (6.9)$$

$$T_{\text{ок}} = \frac{S_{\text{рп}}}{E}, \quad (6.10)$$

$$E\phi = 31443,87 / 47858,4 = 0.66,$$

$$T_{\text{ок}} = 47858,4 / 31443,87 = 1,5 \text{ року.}$$

Зведемо отримані техніко-економічні показники в табл. 6.6.

Таблиця 6.6 – Техніко-економічні показники

Показники	Сума	Одиниця виміру
Витрати на розробку програми	47858,4	грн.
Річна економія	31443,87	грн.
Строк окупності програми	1,5	рік

Строк окупності програми 1,5 року, що значно менше часу її нормативного строку окупності (3 роки). Річна економія при використанні програми становить 31443,87 гривні. Виходячи з виконаних розрахунків, можна зробити висновок, що розробка даного програмного продукту є економічно доцільною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

Статистичні дані показують, що основними і найбільш часто виникаючими захворюваннями при роботі з комп'ютером є запалення очей, глаукома, катаракта, головний біль, хвороби серцево-судинної системи, біль у хребті, різь в очах, біль при русі рук, свербіж шкіри, дерматити, лущення, нервово-психічні розлади, болі в області м'язів обличчя і шиї [17].

Для створення сприятливих умов роботи з комп'ютером необхідна відповідність нормативно-правовим актам з охорони праці, які регламентують показники стану навколишнього середовища:

- достатню площу і об'єм приміщення, де встановлено комп'ютер;
- показники мікроклімату робочого приміщення (температура повітря, відносна вологість, та ін.);
- параметри світлового середовища робочого місця і робочої поверхні (наявність природного і штучного освітлення, освітленість екрана монітора, освітленість горизонтальної поверхні столу, відсутність прямого і відбитого сліпучого відблиску джерел світла, відсутність пульсації освітленості);
- рівень шуму і вібрації робочих місць в межах норм;
- відсутність в повітрі робочої зони шкідливих речовин [17].

Нижче розглядаються заходи з охорони праці, виробничої санітарії, гігієни праці та пожежної безпеки при дослідженні та програмної реалізації методів автоматичного визначення заголовків новин.

7.2 Заходи щодо забезпечення безпеки

Забезпечення безпеки праці, організації оптимальних умов праці, а також захисту навколишнього середовища в умовах роботи з ПК досягається шляхом дотримання норм і проведення заходів, передбачених законодавчими актами України [17].

ПК підключаються до мережі з напругою 220 В і частотою 50 Гц. Згідно Правил улаштування електроустановок 2014 приміщення відноситься до 1-ї категорії «Приміщення без підвищеної небезпеки»: у приміщенні відсутні відкриті струмопровідні частини з напругою понад 12В, при виконанні правил Правил улаштування електроустановок 2014 ураження електричним струмом можливо тільки в разі несправності апаратури і живлячих кабелівс [17].

Електричний поділ мережі передбачає поділ мережі на окремі електрично не зв'язані між собою ділянки за допомогою розділових трансформаторів, підвищення електробезпеки відбувається за рахунок зменшення ємності мережі та підвищення опору ізоляції фаз мережі щодо землі [18].

Зменшення напруги електроустановки істотно підвищує умови електробезпеки: так при напрузі 10-12 В і електричному опорі тіла людини 5-6 кОм сила струму, що протікає через людину, не перевищить 2 мА, згідно ГОСТ 12.1.030-81 «Електробезпека. Захисне заземлення, занулення» [19].

Подвійна ізоляція забезпечує застосування крім основної, робочої ізоляції струмоведучих частин ще одного шару додаткової ізоляції, для зниження величини виникаючих зарядів статичної електрики в ОЦ покриття виконані з одношарового полівінілхлоридного антистатичного лінолеуму, іншим методом захисту є нейтралізація заряду статичної електрики іонізованим газом, до загальних заходів захисту від статичної електрики в ОЦ можна віднести загальне та місцеве зволоження повітря [20].

7.3 Заходи щодо забезпечення виробничої санітарії та гігієни праці

Об'ємно-планувальні рішення будівель та приміщень для роботи з візуальними дисплейними терміналами (ВДТ) ПК повинні відповідати вимогам Державних санітарних норм і правил 3.3.2.007-98 «Державних санітарних правил і норм роботи з візуальними дисплейними терміналами

електронно-обчислювальних машин», розміщення робочих місць з ВДТ ПК в підвальних приміщеннях, на цокольних поверхах заборонено [20].

В обчислювальному центрі при дослідженні та програмній реалізації методів автоматичного визначення заголовків новин забезпечено виконання норми Державних будівельних норм В.2.5-28-2018 «Природне і штучне освітлення», відносно штучного освітлення: освітленість робочих місць в горизонтальній площині на рівні 0,8 м від підлоги не менше 400 лк, вертикальна освітленість в площині екранів дисплея – не менше 300 лк, коефіцієнт пульсації освітленості – не більше 10% [21].

Розрахуємо систему загального рівномірного штучного освітлення в приміщенні масажного кабінету.

Приймаємо розміри приміщення:

– довжина $A = 6\text{м}$;

– ширина $B = 4\text{м}$;

– висота $H = 4\text{ м}$.

Висота робочої поверхні складає $h = 0,7\text{м}$.

Згідно з Державними будівельними нормами В.2.5-28-2018 «Природне і штучне освітлення. Норми проектування» освітлення в кабінеті масажу нормуються як $E_n = 200\text{ Лк}$.

Розряд зорової роботи – III г.

Коефіцієнти відбиття поверхонь приміщення (від: стелі – ρ_c ; стін – $\rho_{ст}$; підлоги – $\rho_{п}$) у світлому приміщенні $\rho_c=70\%$; $\rho_{ст}=50\%$; $\rho_n=30\%$;

Обираємо світильник типу ЛВП (IP=54 – ступінь захисту світильника, $(L/h) = 1,3$ – числове значення коефіцієнта світильника).

Коефіцієнт запасу k_z враховує зниження рівня освітленості з часом в результаті забруднення та старіння ламп, світильників і поверхонь приміщення, приймається в залежності від виробничих умов. Приймаємо $k_z = 1,4$.

Коефіцієнт нерівномірності (мінімального) освітлення z (відношення середньої освітленості до мінімальної освітленості), для люмінесцентних

ламп низького тиску, як правило дорівнює $z = 1,1$.

Розрахуємо кількість рядів світильників у приміщенні за формулою (7.1):

$$N_p = \frac{B}{(H - h_p) \cdot (L / h)} = 0,93 \approx 1. \quad (7.1)$$

Визначимо максимально припустиму відстань між рядами світильників за формулою (7.2):

$$L_{\max} = \frac{B}{N_p}, \quad (7.2)$$

де B – ширина приміщення;

N_p – кількість рядів.

Розрахуємо:

$$L_{\max} = \frac{4}{0,93 \approx 1} = 4 \text{ м.}$$

Розраховуємо висоту підвісу світильника над робочою поверхнею за формулою (7.3):

$$h = \frac{L_{\max}}{(L / h)}, \quad (7.3)$$

$$h = \frac{4}{1,3} = 3,07 \text{ м.}$$

Знайдемо висоту звисання світильника від стелі за формулою (7.4):

$$h_3 = H - h_p - h, \quad (7.4)$$

де H – висота приміщення;

h_p – висота робочої поверхні;

h – висота підвісу світильника над робочою поверхнею.

Розрахуємо:

$$h_3 = 4 - 0,7 - 3,07 = 0,23 \text{ м.}$$

Чисельне значення індексу приміщення визначаємо за рівнянням (7.5):

$$i = \frac{A \cdot B}{h \cdot (A + B)}, \quad (7.5)$$

де A – довжина приміщення, м;

B – ширина приміщення, м;

h – висота розміщення світильників над робочою поверхнею, м.

Розрахуємо:

$$i = \frac{6 \cdot 4}{3,07 \cdot (6 + 4)} = \frac{24}{30,7} = 0,78.$$

Значення коефіцієнта використання світлового потоку η вибирається в залежності від виду джерела світла, типу обраного світильника, коефіцієнтів відбиття поверхонь приміщення (ρ_c , $\rho_{ст}$, ρ_n) та індексу приміщення. В нашому випадку $\eta = 0,38$.

Визначаємо сумарний світловий потік освітлювальної установки у даному виробничому приміщенні за формулою (7.6):

$$\Phi_n = (E_n \cdot S \cdot k_z \cdot z) \div \eta, \quad (7.6)$$

де Φ_{Σ} – розрахункове значення сумарного світлового потоку у приміщенні, лм;

E_n – нормоване значення освітленості, лк;

S – площа освітлюваної поверхні, м²;

k_3 – коефіцієнт запасу;

z – коефіцієнт нерівномірності (мінімальної) освітленості;

η – коефіцієнт використання світлового потоку.

Розрахуємо:

$$\Phi_{\Sigma} = \frac{200 \cdot 6 \cdot 4 \cdot 1,4 \cdot 1,1}{0,38} = \frac{7392}{0,38} = 19452 \text{ лм.}$$

Визначаємо умовну загальну кількість світильників у приміщенні за формулою (7.7):

$$N^* = AB/L_{max}^2, \quad (7.7)$$

$$N^* = \frac{6 \cdot 4}{16} = 1,5 \approx 2 \text{ шт.}$$

Розрахуємо світловий потік умовного джерела світла за формулою (7.8):

$$\Phi_{\lambda} = \frac{\Phi_{\Sigma}}{N_{\lambda}} \text{ лм,} \quad (7.8)$$

де Φ_{Σ} – розрахункове значення сумарного світлового потоку у приміщенні, лм;

N_{λ} – загальна кількість ламп у приміщенні, шт.

Загальна кількість ламп у приміщенні визначається за формулою (7.9):

$$N_{\text{л}} = N^* \cdot n, \quad (7.9)$$

де n – кількість ламп у світильнику (у ЛВП світильнику знаходиться 2 лампи).

Розрахуємо:

$$N_{\text{л}} = 2 \cdot 2 = 4 \text{ шт.}$$

Тоді,

$$\Phi_{\text{л}} = \frac{19452}{4} = 4863 \text{ лм.}$$

Обираємо тип стандартної лампи з найближчим значенням фактичного світлового потоку лампи $\Phi_{\text{л}}$ і знаходимо коефіцієнт m (співвідношення між розрахунковим світловим потоком лампи $\Phi_{\text{л}}$ та фактичним світловим потоком вибраної стандартної лампи $\Phi_{\text{л}}$).

Обираємо лампу ЛБ65, де $\Phi_{\text{л}} = 4800 \text{ лм.}$

Розрахуємо:

$$m = \frac{4863}{4800} = 1,01.$$

Визначаємо оптимальну (фактичну) кількість світильників у приміщенні за формулою (7.10):

$$N = N \cdot m, \quad (7.10)$$

$$N = 2 \cdot 1,01 = 2 \text{ шт.}$$

Фактична кількість ламп у приміщенні розраховується за формулою (7.11):

$$N_{\phi l} = N_{\phi} \cdot n, \quad (7.11)$$

$$N_{\phi l} = 2 \cdot 2 = 4 \text{ шт.}$$

Визначаємо загальну розрахункову освітленість E_p у приміщенні, що створюється при застосуванні стандартних ламп, за формулою (7.12):

$$E_p = \frac{\Phi_l N_{\phi l} \eta}{S k_z} \text{ лк}, \quad (7.12)$$

$$E_p = \frac{4800 \cdot 4 \cdot 0,38}{6 \cdot 4 \cdot 1,4 \cdot 1,1} = \frac{7296}{36,96} = 197,4 \text{ лк.}$$

Нормовані параметри мікроклімату, іонного складу повітря, вмісту шкідливих речовин відповідають вимогам Державних санітарних норм 3.36.042-99 «Санітарні норми мікроклімату виробничих приміщень» [22]-[23].

Велике значення має задоволення обстановки ергономічним і естетичним вимогам, цього домоглися за допомогою колірного рішення приміщення і обладнання обчислювального центру з урахуванням вимог ГОСТ 17.2.032-78 – в приміщеннях стіни пофарбовані фарбами пастельних тонів [24].

При організації роботи для збереження здоров'я, запобігання професійних захворювань і підтримки працездатності передбачені перерви для відпочинку, тривалість обідньої перерви визначається чинним законодавством про працю і Правилами внутрішнього трудового розпорядку підприємства [24]-[25].

7.4 Заходи щодо забезпечення пожежної безпеки

Заходи з протипожежної безпеки розроблені для виробничого приміщення обладнаного ПК з ВДТ розроблені згідно вимог

Нормативно-правових актів з пожежної безпеки А.01.001-2014 «Правила пожежної безпеки в Україні» [26].

Виходячи з аналізу речовин та матеріалів, які використовуються при роботі у приміщенні, обладнаному ПК з ВДТ:

- згідно ГОСТ 27331-87 [26] у приміщенні, обладнаному ПК з ВДТ можлива пожежа класу – А та Е;

- відповідно до вимог Державних будівельних норм В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [27], воно належить до категорії «Д» з пожежної небезпеки – простір у приміщенні, у якому перебувають тверді горючі речовини та матеріали;

- у разі виникнення пожежі у приміщенні обладнаному ПК з ВДТ для евакуації персоналу відповідно до вимог Державних будівельних норм В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» передбачені виходи, по обидві сторони приміщення, з одного боку вікно (на пожежні сходи), а з іншого – вхідні двері, згідно Санітарних норм і правил 2.09.02-85, відстань від найбільш віддаленого робочого місця до найближчого евакуаційного виходу не обмежується [26]-[27].

Обладнання, силові та освітленні мережі приміщення, обладнаного ПК з ВДТ відповідають вимогам пожежної безпеки, оскільки виконані відповідно до вимог Нормативно-правових актів з пожежної безпеки 40.1-1.32-01 «Правила будови електроустановок. Електрообладнання спеціальних установок», та мають ступінь захисту ізоляції обладнання IP44 яка відповідає класу пожежонебезпечної зони П-Па до якої належить приміщення [28].

Оскільки приміщення що обладнане ПК з ВДТ має площу 24 м², тому відповідно до вимог п. 3.8 розділу «Типові норми належності вогнегасників» ДСТУ 4297:2004 «Пожежна техніка, технічне обслуговування вогнегасників, загальні технічні вимоги» для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 2 штуки (з розрахунку один вогнегасник с величиною заряду вогнегасної

речовини 3 кг. і більше, на 20 м² площі приміщення), відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м [29].

7.5 Заходи щодо забезпечення безпеки в надзвичайних ситуаціях

Забезпечення стійкості роботи систем газопостачання промислового об'єкта досягається:

- виконанням норм інженерно-технічних заходів при будівництві;
- проведенням додаткових заходів щодо підвищення фізичної стійкості будівель газорозподільних станцій і пунктів;
- прокладкою під землею газопроводів високого та середнього тиску;
- надійною роботою систем аварійного відключення ділянок газопроводу при аварійних ситуаціях;
- підвищенням надійності газопостачання об'єктів за рахунок кільцювання газопроводів в межах міст;
- подачею газу споживачеві не менше ніж через дві газорозподільні станції;
- впровадженням в диспетчерське управління і обслуговування газового господарства телемеханічних пристроїв і автоматики;
- високим рівнем професійної підготовки обслуговуючого персоналу, здатного запобігати або локалізувати аварійні ситуації [30]-[31].

ВИСНОВКИ

У дипломній кваліфікаційній роботі магістра розглянуто проблему пошуку комп'ютерних ігор. Розглянуто поняття комп'ютерних ігор, особливості жанрового різноманіття, масштаби та тенденції розвитку ринку комп'ютерних ігор. Величезна кількість ігор, жанрова фрагментація та фрагментація за платформами призводять до необхідності створення платформ для доступу до даних різних ігор, тобто агрегації комп'ютерних ігор. В якості таких платформ було розглянуто і проаналізовано функціональні можливості платформ metacritic та xbox.

Виконано огляд методів інформаційного пошуку, серед яких розглянуто теоретико-множинні моделі, імовірнісні моделі, алгебраїчні моделі та гібридні моделі. Розглянуто метод кластерного аналізу k-середніх.

Виконано вибір мови програмування C# для реалізації програмного забезпечення, оскільки вона є мовою високого рівня, дозволяє виконувати автоматичне управління пам'яттю, може запускатися на різних платформах за рахунок .NET Core, є чистою об'єктно-орієнтованою мовою, дозволяє виконувати перевірку границь масивів, а також створювати широкі сучасні застосунки. Виконано моделювання функціональних вимог до програмного забезпечення у вигляді діаграм прецедентів UML для адміністратора та користувача. Виконано проектування структури бази даних з вибором Microsoft SQL Server у якості системи керування базою даних та з приведенням структурної схеми бази даних. Виконано проектування структури застосунку зі створенням діаграми прецедентів UML.

Розроблено програмне забезпечення агрегації комп'ютерних ігор для забезпечення можливостей персоналізованого пошуку комп'ютерних ігор за запитом користувача. Описано прийняті рішення під час розробки програми.

Таким чином, можна вважати, що завдання на дипломну кваліфікаційну роботу магістра виконані повністю, а мету роботи досягнуто.

Наукова новизна роботи полягає в тому, що в ній розроблено метод пошуку комп'ютерних ігор, який використовує метод k-середніх для розділення комп'ютерних ігор на кластери, визначає прихильність користувача до ігор відповідного кластеру за відповідями на поставлені запитання або за уподобаними комп'ютерними іграми, що дозволяє виконувати персоналізований пошук у відповідності з тими значеннями характеристик комп'ютерних ігор, яким віддає перевагу користувач.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Как создать игровой сайт [Электрон. ресурс] : три составляющих вашего успеха. – Режим доступа : <https://www.motocms.com/blog/ru/kak-sozdat-igrovoi-site/>
2. Жанры компьютерных игр [Электрон. ресурс]. – Режим доступа : <http://www.4stud.info/oss/computer-game-genres.html>
3. Жанры компьютерных игр – гайд по жанрам игр на ПК [Электрон. ресурс]. – Режим доступа : <https://cubiq.ru/zhanry-kompyuternyh-igr/>
4. Newzoo Global Mobile Market Report 2019 [Electronic resource]. – Access mode : <https://newzoo.com/insights/trend-reports/newzoo-global-mobile-market-report-2019-light-version/>
5. Анализ и тенденции индустрии мобильных игр в 2020 году [Электрон. ресурс]. – Режим доступа : <https://vc.ru/life/114169-analiz-i-tendencii-industrii-mobilnyh-igr-v-2020-godu>
6. Глобальный обзор игровой индустрии: тренды, инсайты и прогнозы на 2021 год [Электрон. ресурс]. – Режим доступа : <https://adindex.ru/news/researches/2019/01/25/230750.phtml>
7. Типы информационного поиска. Методы и алгоритмы полнотекстового поиска [Электрон. ресурс]. – Режим доступа : <https://sites.google.com/site/dssearchtext/events/tipyinformacionnogopoiska>
8. Урвачева, В.А. Обзор методов информационного поиска [Текст] / В.А. Урвачева // Вестник Таганрогского института имени А.П. Чехова. – 2016. – № 1. – С. 457-463.
9. Кластерный анализ [Электрон. ресурс]. – Режим доступа : <https://www.statmethods.ru/statistics-metody/klasternyj-analiz/>
10. Методы кластерного анализа. Итеративные методы [Электрон. ресурс]. – Режим доступа : <https://intuit.ru/studies/courses/6/6/lecture/184>

11. Шапошникова, Е.А. Исследование и анализ алгоритма кластеризации данных методом k-средних [Текст] / Е.А. Шапошникова // Международный научный журнал «Интернаука». – 2017. – № 18 (40), том 1. – С. 94-96.
12. C++ Vs. C# – What’s the Difference? [Electronic resource]. – Access mode : <https://www.guru99.com/cpp-vs-c-sharp.html>
13. Скит, Дж. С# для профессионалов [Текст] : тонкости программирования, 3-е издание, новый перевод = C# in Depth, 3rd ed. / Дж. Скит. – М. : Вильямс, 2014. – 608 с.
14. Страуструп, Б. Язык программирования C++. Краткий курс [Текст] / Б. Страуструп. – М. : Вильямс, 2019. – 320 с.
15. Диаграммы прецедентов [Электрон. ресурс]. – Режим доступа : http://it.samgtu.ru/sites/it.samgtu.ru/files/diagrammy_precedentov.pdf
16. Остапенко, В.В. Методичні вказівки по економічному обґрунтуванню дипломних проєктів для студентів спеціальностей 7.080403 «Програмне забезпечення автоматизованих систем», 7.080402 «Інформаційні технології проєктування» і 7.091501 «Комп’ютерні та інтелектуальні системи і мережі» всіх форм навчання [Текст] / В.В. Остапенко, Е.М. Касьян, Є.М. Сердюк. – Запоріжжя : ЗНТУ, 2009. – 22 с.
17. Правила улаштування електроустановок [Текст] : Нормативне виробничо-практичне видання : вид. 5-те, перероблене й доповнене : затв. М-вом енергетики та вугільної промисловості України 20.06.14 : введення в дію з 20.11.14. – Х. : Міненерговугілля України, 2014. – 793 с.
18. Бедрій, Я.І. Основи охорони праці [Текст] : Навчальний посібник для студентів вищих навчальних закладів / Я.І. Бедрій. – Тернопіль : Навчальна книга – Богдан, 2014. – 240 с.
19. Голінько, В.І. Основи охорони праці [Текст] / В.І. Голінько. – Дніпропетровськ : Національний гірничий університет, 2010. – 271 с.
20. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. НПАОП 0.00-7.15-18. Державні санітарні

правила і норми [Текст]. – Харків : Форт, 2018. – 48 с.

21. ДБН В.2.5-28:2018. Природне та штучне освітлення. Посібник [Текст] : проблеми природного і штучного освітлення. – К. : Етін, 2019. – 180 с.

22. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку [Текст]. – К. : МОЗ України, 1999. – 24 с.

23. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень [Текст]. – Харків : Форт, 2000. – 24 с.

24. ГОСТ 17.2.032-78 Рабочее место при выполнении работ сидя Система стандартов безопасности труда [Текст] : Сб. ГОСТов. – М. : ИПК Издательство стандартов, 2001. – 7 с.

25. Основи охорони праці. Підручник [Текст] / О. І. Запорожець, О. С. Протоєрейський, Г. М. Франчук, І. М. Боровик. – К. : Центр учбової літератури, 2009. – 264 с.

26. Правила пожежної безпеки в Україні [Текст]. – Харків : Форт, 2015. – 124 с.

27. ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [Текст]. – Київ : Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України, 2017. – 39 с.

28. НПАОП 40.1-1.32-01 Правила будови електроустановок. Електрообладнання спеціальних установок [Текст]. – Харків : Індустрія, 2010. – 118 с.

29. ДСТУ 4297:2004 Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги [Текст]. – Харків : Форт, 2004. – 48 с.

30. Основи цивільного захисту [Текст] : Навч. посібник / В.О. Васійчук, В.Є. Гончарук, С.І. Качан, С.М. Мохняк. – Львів : ЛНУ, 2010. – 384 с.

31. Стеблюк, М.І. Цивільна оборона та цивільний захист [Текст] : Підручник / М.І. Стеблюк. – К. : Знання-Прес, 2007. – 487 с.

ДОДАТОК А
Технічне завдання

Вступ

Основне призначення програмного забезпечення – створення платформи для агрегації даних про різні комп'ютерні ігри.

Суть програмного забезпечення полягає в тому, що з його допомогою має відбуватися орієнтування у величезному світі комп'ютерних ігор: з одного боку воно має надавати інформацію про ігри, які призначені для різних платформ, є безкоштовними або платними, а з іншого боку – воно має дозволяти виконувати пошук ігор, які можуть зацікавити користувача, адже без такого якісного інструменту орієнтуватися у всьому масиві даних буде майже неможливо.

A.1 Підстави для розробки

Підставою для розробки є завдання на магістерську роботу за темою “Дослідження та програмна реалізація методів пошуку комп'ютерних ігор”, затверджене наказом № 301 від 3 листопада 2020 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Призначення розробки – підбір комп'ютерних ігор за перевагами користувача у світі комп'ютерних ігор та надання йому інформації про відібрані ігри з можливістю наступного переходу на зовнішні ресурси.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно виконувати наступні функції:

- додавання інформації про комп'ютерну гру;

- пошук комп'ютерних ігор за назвою;
- персоналізований пошук комп'ютерних ігор;
- виставлення власної оцінки за комп'ютерну гру;
- внесення коментаря про комп'ютерну гру;
- перегляд коментарів користувачів про комп'ютерну гру;
- перегляд комп'ютерних ігор за жанром;
- перегляд найкращих комп'ютерних ігор за оцінками користувачів;
- редагування інформації про раніше додану комп'ютерну гру;
- авторизація;
- реєстрація.

A.3.2 Вимоги до надійності

Програмне забезпечення повинно коректно працювати як за наявності доступу до бази даних, що розташована на сервері, так і за його відсутності. У випадку введення некоректних даних мають виводитися повідомлення про помилку.

A.3.3 Умови експлуатації

Для використання програмного забезпечення необхідно мати персональний комп'ютер чи ноутбук з інстальованою операційною системою лінійки Windows, версії не нижче від Windows 7.

Для розробки необхідно використовувати середовище Visual Studio версії не нижче ніж 2017.

A.3.4 Вимоги до складу та параметрів технічних засобів

Для роботи програмного забезпечення на серверній стороні має забезпечуватися наступна мінімальна конфігурація апаратно-програмного

забезпечення комп'ютера:

- двоядерний процесор з частотою не менше 2,4 ГГц;
- оперативна пам'ять – не менше 4 Гб;
- наявність не менше 2 Гб вільного простору на жорсткому диску.

А.3.5 Вимоги до транспортування та збереження

Вимоги до транспортування й зберігання аналогічні вимогам, пропонованим до носія, на якому записана програма.

А.3.6 Вимоги до програмної документації

Розроблювана програмна документація:

- документ «Текст програми»;
- документ «Технічне завдання».

А.3.7 Вимоги до маркування та упакування

Програма може бути записана як на компакт-диску чи на зйомному носії. На пакуванні повинна бути вказана назва програми – «Дослідження та програмна реалізація методів пошуку комп'ютерних ігор».

ДОДАТОК Б
Текст програми

Б.1 Текст файлу DataProcessorModule.cs

```
//Модуль роботи з даними

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gamer
{

    //інтерфейс доступу для авторизації до модуля роботи з даними
    interface IAuthorization
    {
        int Auth(string login, string pass);
        int GetAccountType(int num);
        int Registrer(string login, string pass);
        List<Account> getAllAccounts();
    }

    //інтерфейс доступу адміністратора до модуля роботи з даними
    interface IAdmin
    {
        int FillAllGames ();
        bool AddGame(string title, string info[100], string url, string
developer, int gnum, int ognum, float price, string rating, int dyear, string photo,
List<String> genres, List<String> platforms, List<String> languages);
        List<Games> getAllGames();
        bool detectGameClusters();
        bool EditGame(int num, string title, string info[100], string url, float
rate, string developer, int gnum, int ognum, float price, string rating, int dyear,
string photo, List<String> genres, List<String> platforms, List<String>
languages);
        List<Game> SearchGame(string namegame);
        List<Comment> getGameComments(int gamenum);
        bool EditComment (int commnum, int accnum, string text[100],
DateTime when);
        bool calcAllGamesRate();
        bool calcAllGamesRate(int gamenum);
    }
}
```

```

//інтерфейс доступу користувача до модуля роботи з даними
interface IUser
{
    int GetAccountCluster(int num);
    int FillAllGames ();
    List<Game> SearchGame(string namegame);
    List<Game> PersonalSearchGame(int accnum, string namegame)
    bool AddComment (int gamenum, int accnum, string text[100],
DateTime when);
    bool detectUserCluster(int accnum, List<string> answers);
    bool AddGrade (int gamenum, int accnum, float valuee, DateTime
when);
    List<Comment> getGameComments(int gamenum)
    bool calcAllGamesRate();
    bool calcAllGamesRate(int gamenum);
    List<Game> GameGenreBest (string genre);
    List<Game> BestGames();
}

public class DataProcessorModule : IAdmin, IUser, IAuthorization
{
    List <Game> allgames = new List<Game>();
    List <Account> allaccs = new List<Account>();

    public int FillAllGames ()
    {
        List<Game> games;
        allgames.Clear();
        games = new List<Game>();
        Database GamerDBconn = new Database();
        GamerDBconn.open();
        games = GamerDBconn.getAllGames();
        int i;
        for (i=0; i<games.Count; i++)
        {
            allgames.Add(games[i]);
            List<Grade> grades;
            grades = new List<Grade>();
            grades = GamerDBconn.getGameGrades(allgames[i].Id);
            int j;
            for (j=0; j<grades.Count; j++)
                allgames[i].Grades.Add(grades[j]);
            List<Comment> comms;
            comms = new List<Comment>();
            comms = GamerDBconn.getGameComments(allgames[i].Id);

```

```

        for (j=0; j<comms.Count; j++)
            allgames[i].Comments.Add(comms[j]);
    }
    GamerDBconn.close();
    return allgames.Count;
}

public int FillAllAccounts ()
{
    List<Account> lacc;
    allaccs.Clear();
    lacc = new List<Account>();
    Database GamerDBconn = new Database();
    GamerDBconn.open();
    lacc = GamerDBconn.getAllAccounts();
    int i;
    for (i=0; i<lacc.Count; i++)
    {
        allaccs.Add(lacc[i]);
    }
    GamerDBconn.close();
    return allaccs.Count;
}

public bool AddGame(string title, string info[100], string url, string
developer, int gnum, int ognum, float price, string rating, int dyear, string photo,
List<String> genres, List<String> platforms, List<String> languages)
{
    Game g = new Game();
    Database GamerDBconn = new Database();
    GamerDBconn.open();
    int num = GamerDBconn.getMaxGameId();
    g.Id = num + 1;
    g.Title = title;
    g.Info = info;
    g.Url = url;
    g.Developer = developer;
    g.Gnum = gnum;
    g.Ognum = ognum;
    g.Price = price;
    g.Rating = rating;
    g.Dyear = dyear;
    g.Photo = photo;
    g.Genres = new List<String>(genres);
    g.Platforms = new List<String>(platforms);
}

```

```

        g.Languages = new List<String>(languages)
        allgames.Add(g);
        bool res = GamerDBconn.addGame(g);
        GamerDBconn.close();
        return res;
    }

    public List<Games> getAllGames()
    {
        return allgames;
    }

    public List<Account> getAllAccounts()
    {
        return allaccs;
    }

    public bool EditGame(int num, string title, string info[100], string url,
float rate, string developer, int gnum, int ognum, float price, string rating, int
dyear, string photo, List<String> genres, List<String> platforms, List<String>
languages)
    {
        if (!title.IsNullOrEmpty())
            allgames[num].Title = title;
        if (!info[0]..IsNullOrEmpty())
            allgames[num].Info = info;
        if (!url.IsNullOrEmpty())
            allgames[num].Url = url;
        if (rate < 0.0000000001)
            allgames[num].Rate = rate;
        if (!developer.IsNullOrEmpty())
            allgames[num].Developer = developer;
        if (gnum != 0)
            allgames[num].Gnum = gnum;
        if (ognum != 0)
            allgames[num].Ognum = ognum;
        if (price < 0.0000000001)
            allgames[num].Price = price;
        if (!rating.IsNullOrEmpty())
            allgames[num].Rating = rating;
        if (dyear != 0)
            allgames[num].Dyear = dyear;
        if (!photo.IsNullOrEmpty())
            allgames[num].Photo = photo;
        if (genres.Count != 0)

```

```

        allgames[num].Genres = new List<String>(genres);
    if (platforms.Count != 0)
        allgames[num].Platforms = new List<String>(platforms);
    if (languages.Count != 0)
        allgames[num].Languages = new List<String>(languages)
    Game g = new Game();
    Database GamerDBconn = new Database();
    GamerDBconn.open();
    bool    res    =    GamerDBconn.editGame(allgames[num].Id,
allgames[num]);
    GamerDBconn.close();
    return res;
}

public List<Game> SearchGame(string namegame)
{
    List<Game> games;
    games = new List<Game>();
    int i;
    for (i=0; i<allgames.Count(); i++)
    {
        string str = allgames[i].Title.ToUpper();
        if (str.Contains(namegame.ToUpper()))
            games.Add(allgames[i]);
    }
    return games;
}

public List<Game> PersonalSearchGame(int accnum, string
namegame)
{
    List<Game> games;
    games = new List<Game>();
    int i;
    for (i=0; i<allgames.Count(); i++)
    {
        if ((allgames[i].Cluster>allaccs[accnum].Cluster-0.1) &
(allgames[i].Cluster<allaccs[accnum].Cluster+0.1))
        {
            string str = allgames[i].Title.ToUpper();
            if (str.Contains(namegame.ToUpper()))
                games.Add(allgames[i]);
        }
    }
    List<Game> SortedList = gamesgenre.OrderBy(o=>o.Rate).ToList();
    return games;
}

```

```

    }

    public bool AddComment (int gamenum, int accnum, string text[100],
DateTime when)
    {
        Comment c = new Comment();
        Database GamerDBconn = new Database();
        GamerDBconn.open();
        int num = GamerDBconn.getMaxCommentId();
        c.Id = num + 1;
        c.Text = text;
        c.When = when;
        c.User = allaccs[accnum];
        allgames[gamenum].Comments.Add(c);
        GamerDBconn.AddComment(c);
        GamerDBconn.close();
        return true;
    }

    public bool AddGrade (int gamenum, int accnum, float valuee,
DateTime when)
    {
        int i;
        for (i=0; i<allgames[gamenum].Grades.Count; i++)
            if (allgames[gamenum].Grades[i].User.Id == accnum)
                return false;
        Grade g = new Grade();
        Database GamerDBconn = new Database();
        GamerDBconn.open();
        int num = GamerDBconn.getMaxGradeId();
        g.Id = num + 1;
        g.Valuee = valuee;
        g.When = when;
        g.User = allaccs[accnum];
        allgames[gamenum].Grades.Add(g);
        GamerDBconn.AddGrade(g);
        GamerDBconn.close();
        calcAllGamesRate(gamenum);
        return true;
    }

    public bool detectGameClusters()
    {
        GonMethod gm = new GonMethod();
        List<float> gclusters = gm.Clusterize(allgames);
    }

```

```

        if (gclusters.Count==0)
            return false;
        int i;
        for (i=0; i<glusters.Count; i++)
            allgames[i].Cluster = glusters[i];
        return true;
    }

    public bool detectUserCluster(int accnum, List<string> answers)
    {
        GonMethod gm = new GonMethod();
        float cl = gm.GetCluster(accnum, answers);
        if (cl == -1)
            return false;
        allaccs[i].Cluster = cl;
        Database GamerDBconn = new Database();
        GamerDBconn.open();
        GamerDBconn.SaveUserAnswers(int accnum, List<string>
answers);
        GamerDBconn.close();
        return true;
    }

    public List<Comment> getGameComments(int gamenum)
    {
        return allgames[gamenum].Comments;
    }

    public bool EditComment (int commnum, int accnum, string
text[100], DateTime when)
    {
        int i;
        int c=-1;
        for (i=0; i<allcomms.Count; i++)
            if (allcomms[i].Id == commnum)
                c = i;
        Database GamerDBconn = new Database();
        GamerDBconn.open();
        allcomms[c].Text = text;
        allcomms[c].When = when;
        allcomms[c].User = allaccs[accnum];
        int j;
        for (i=0; i<allgames.Count; i++)
            for (j=0; j<allgames[i].Comments.Count; j++)
                if (allgames[i].Comments[j].Id == commnum)

```

```

        {
            allgames[i].Comments[j].Text = text;
        }
    bool res = GamerDBconn.EditComment(allcomms[c]);
    GamerDBconn.close();
    return res;
}

public bool calcAllGamesRate()
{
    int i;
    for (i=0; i<allgames.Count(); i++)
    {
        double r = 0;
        int j;
        for (j=0; j<allgames[i].Grades.Count; j++)
            r += allgames[i].Grades[j];
        r = r/allgames[i].Grades.Count;
        allgames[i].Rate = r;
        Database GamerDBconn = new Database();
        GamerDBconn.open();
        bool res = GamerDBconn.editGame(allgames[i].Id,
allgames[i]);
        GamerDBconn.close();
    }
}

public bool calcAllGamesRate(int gamenum)
{
    double r = 0;
    int j;
    for (j=0; j<allgames[gamenum].Grades.Count; j++)
        r += allgames[gamenum].Grades[j];
    r = r/allgames[gamenum].Grades.Count;
    allgames[gamenum].Rate = r;
    Database GamerDBconn = new Database();
    GamerDBconn.open();
    bool res = GamerDBconn.editGame(allgames[gamenum].Id,
allgames[gamenum]);
    GamerDBconn.close();
}

public List<Game> GameGenreBest (string genre)
{
    calcAllGamesRate();
}

```

```

        List<Game> gamesgenre = new List<Game>();
        int i;
        for (i=0; i<allgames.Count; i++)
        {
            int j;
            for (j=0; j<allgames[i].Genres
                if (allgames[i].Genres[j] == genre)
                    gamesgenre.Add(allgames[i]);
            }
        List<Game> SortedList =
gamesgenre.OrderBy(o=>o.Rate).ToList();
        return SortedList;
    }

    public List<Game> BestGames()
    {
        calcAllGamesRate();
        List<Game> SortedList =
allgames.OrderBy(o=>o.Rate).ToList();
        return SortedList;
    }

    public int Auth(string login, string pass)
    {
        int i;
        for (i=0; i<allaccs.Count(); i++)
        {
            if (allaccs[i].Login == login & allaccs[i].Pass == pass)
                return allaccs[i].Id;
        }
        return -1;
    }

    public int GetAccountCluster(int num)
    {
        return allaccs[num].Cluster;
    }

    public int GetAccountType(int num)
    {
        return allaccs[num].Type;
    }

    public int Registrer(string login, string pass)
    {

```

```

        int i;
        for (i=0; i<allaccs.Count(); i++)
        {
            if (allaccs[i].Login == login)
                return -1;
        }
        Account ac = new Account();
        Database GamerDBconn = new Database();
        GamerDBconn.open();
        int num = GamerDBconn.getMaxAccountId();
        ac.Id = num + 1;
        ac.Login = login;
        ac.Pass = pass;
        ac.Type = 0;
        allaccs.Add(ac);
        GamerDBconn.AddAccount(ac);
        GamerDBconn.close();
        return num+1;
    }
}

```

Б.2 Текст файла KmeansCluster.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Linq;
namespace Gamer
{
    class Cluster
    {
        private static readonly Random rndm = new Random();
        public Cluster cdat;
        public double val[4];
        public ArrayList daal = new ArrayList();

        public Cluster(double[] x, double[] y)
        {
            int i;
            for (i=0; i<4; i++)
            {

```

```

        double t = y[i] - x[i];
        this.val[i] = rndm.NextDouble() * t + x[i];
    }
}

public Cluster(Cluster clstr)
{
    int i;
    for (i=0; i<4; i++)
    {
        this.val[i] = clstr.val[i];
    }
}

public double[] mean()
{
    int dasize = daal.Count;
    double[] dval = new double[4];
    foreach (Observation l in daal)
    {
        int i;
        for (i=0; i<4; i++)
        {
            dval[i] += l.cvals[i];
        }
    }
    dval.Select(j => j / dasize);
    return dval;
}

public double[] dispersion()
{
    double[] mean=this.mean();
    double[] dsp = new double[4];
    int i;
    for (int i=0; i<4; i++)
    {
        foreach(Observation l in daal)
        {
            dsp [i]+=Math.Pow((mean[i] - l.deger[i]), 2);
        }
    }
    dsp.Select(x => x /(daal.Count*1000));
    return dsp;
}

```

```

public bool detect(int num)
{
    double[] disp=this.dispersion();
    bool flag;
    flag = false;
    if (num > 500)
    {
        flag = true;
    } else if (disp [0]==0 && disp [1] == 0 && disp [2] == 0 && disp
[3] == 0)
    {
        flag = true;
    }
    return flag;
}

public void CreateCluster()
{
    double[] m=this.mean();
    this.cdat = new Cluster(this);
    int i;
    for (i=0; i<4; i++)
    {
        this.val[i] = m[i];
    }
}

class Observation
{
    public double[] cvals = new double[4];
    public string parv;
    public Observation (double v1, double v2, double v3, double v4,
string parv)
    {
        this.cvals[0] = v1;
        this.cvals[1] = v2;
        this.cvals[2] = v3;
        this.cvals[3] = v4;
        this.parv = parv;
    }
}

```

```
}
```

Б.3 Текст файла Game.cs

```
using System.IO;

namespace Gamer
{
    public class Game
    {
        private int id;
        private string title;
        private string info[100];
        private string url;
        private float rate;
        private string developer;
        private int gnum;
        private int ognum;
        private float price;
        private string rating;
        private int dyear;
        private string photo;
        private float cluster;
        private List<String> genres = new List<String>();
        private List<String> platforms = new List<String>();
        private List<String> languages = new List<String>();
        private List<Grade> grades = new List<Grade>();
        private List<Comment> comments = new List<Comment>();
        public Game ()
        {
            this.Id = 0;
            this.Title = "";
            this.Info= new string[100];
            this.Url = "";
            this.Rate = 0.0;
            this.Developer = "";
            this.Gnum = 0;
            this.Ognum = 0;
            this.Price = 0;
            this.Rating = "";
            this.Dyear = 1900;
            this.Photo = "";
            this.Cluster = -1;
        }
    }
}
```

```
}

public int Id
{
    get => id;
    set => id = value;
}

public string Title
{
    get => title;
    set => title = value;
}

public string Info
{
    get => info;
    set => info = value;
}

public string Url
{
    get => url;
    set => url = value;
}

public float Rate
{
    get => rate;
    set => rate = value;
}

public string Developer
{
    get => developer;
    set => developer = value;
}

public int Gnum
{
    get => gnum;
    set => gnum = value;
}

public int Ognum
```

```
{
    get => ognum;
    set => ognum = value;
}

public float Price
{
    get => price;
    set => price = value;
}

public string Rating
{
    get => rating;
    set => rating = value;
}

public int Dyear
{
    get => dyear;
    set => dyear = value;
}

public string Photo
{
    get => photo;
    set => photo = value;
}

public float Cluster
{
    get => cluster;
    set => cluster = value;
}

public List<String> Genres
{
    get => genres;
    set => genres = value;
}

public List<String> Platforms
{
    get => platforms;
```

```

        set => platforms = value;
    }

    public List<String> Languages
    {
        get => languages;
        set => languages = value;
    }

    public List<Grade> Grades
    {
        get => grades;
        set => grades = value;
    }
}

public class Grade
{
    private int id;
    private Account user;
    private float valuee;
    private DateTime when;

    public Grade ()
    {
        this.Id = 0;
        this.User = new Account();
        this.Value = 0.0;
        this.When = DateTime.Now;
    }

    public int Id
    {
        get => id;
        set => id = value;
    }

    public Account User
    {
        get => user;
        set => user = value;
    }

    public float Valuee
    {

```

```

        get => valuee;
        set => valuee = value;
    }

    public DateTime When
    {
        get => when;
        set => when = value;
    }
}

public class Comment
{
    private int id;
    private string text[100];
    private Account user;
    private DateTime when;

    public Comment ()
    {
        this.Id = 0;
        this.User = new Account();
        this.Text = new string [100];
        this.When = DateTime.Now;
    }

    public int Id
    {
        get => id;
        set => id = value;
    }

    public Account User
    {
        get => user;
        set => user = value;
    }

    public string Text
    {
        get => text;
        set => text = value;
    }
}

```

```

    public DateTime When
    {
        get => when;
        set => when = value;
    }

}

public class Account
{
    private int id;
    private string login;
    private string pass;
    private float cluster;
    private int type;

    public Account ()
    {
        this.Id = 0;
        this.Login = "";
        this.Pass = "";
        this.Cluster = -1;
    }

    public int Id
    {
        get => id;
        set => id = value;
    }

    public string Login
    {
        get => login;
        set => login = value;
    }

    public string Pass
    {
        get => pass;
        set => pass = value;
    }

    public float Cluster
    {
        get => cluster;
        set => cluster = value;
    }
}

```

```
public int Type
{
    get => type;
    set => type = value;
}

}

}
```

ДОДАТОК В
Слайди презентації

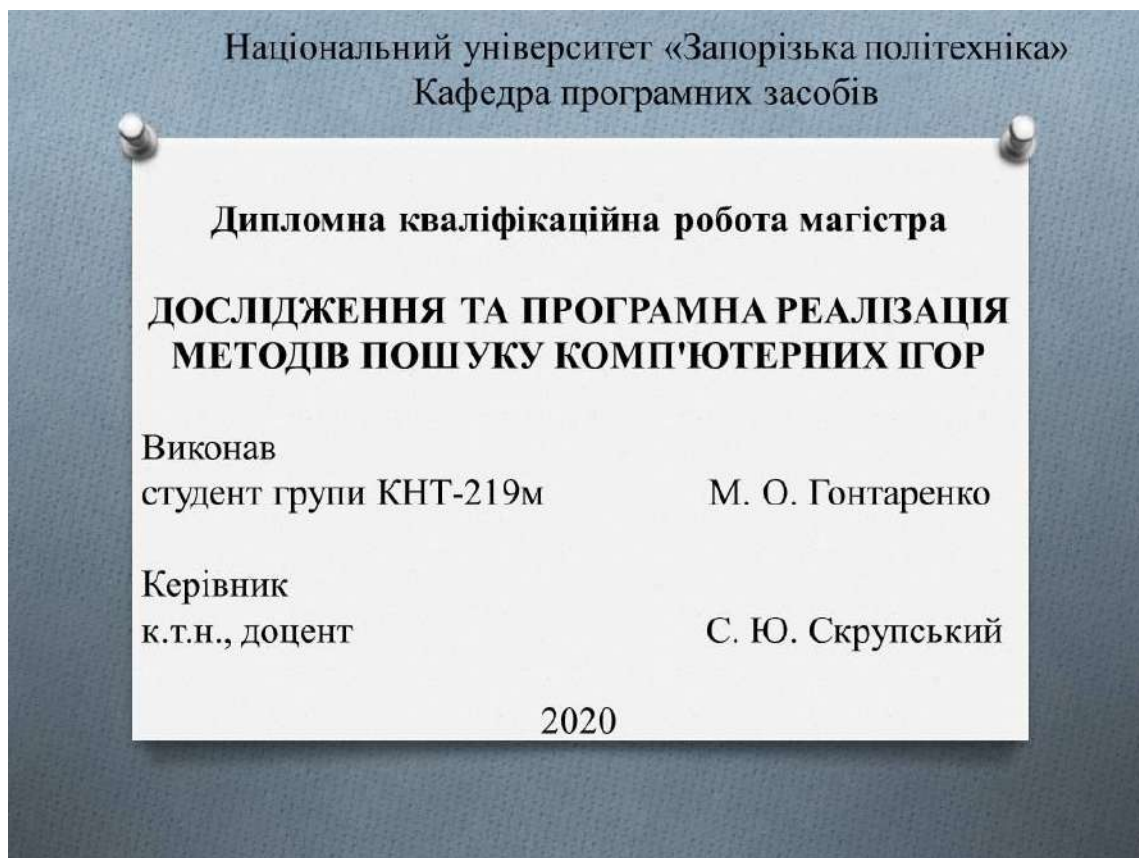


Рисунок В.1 – Слайд 1

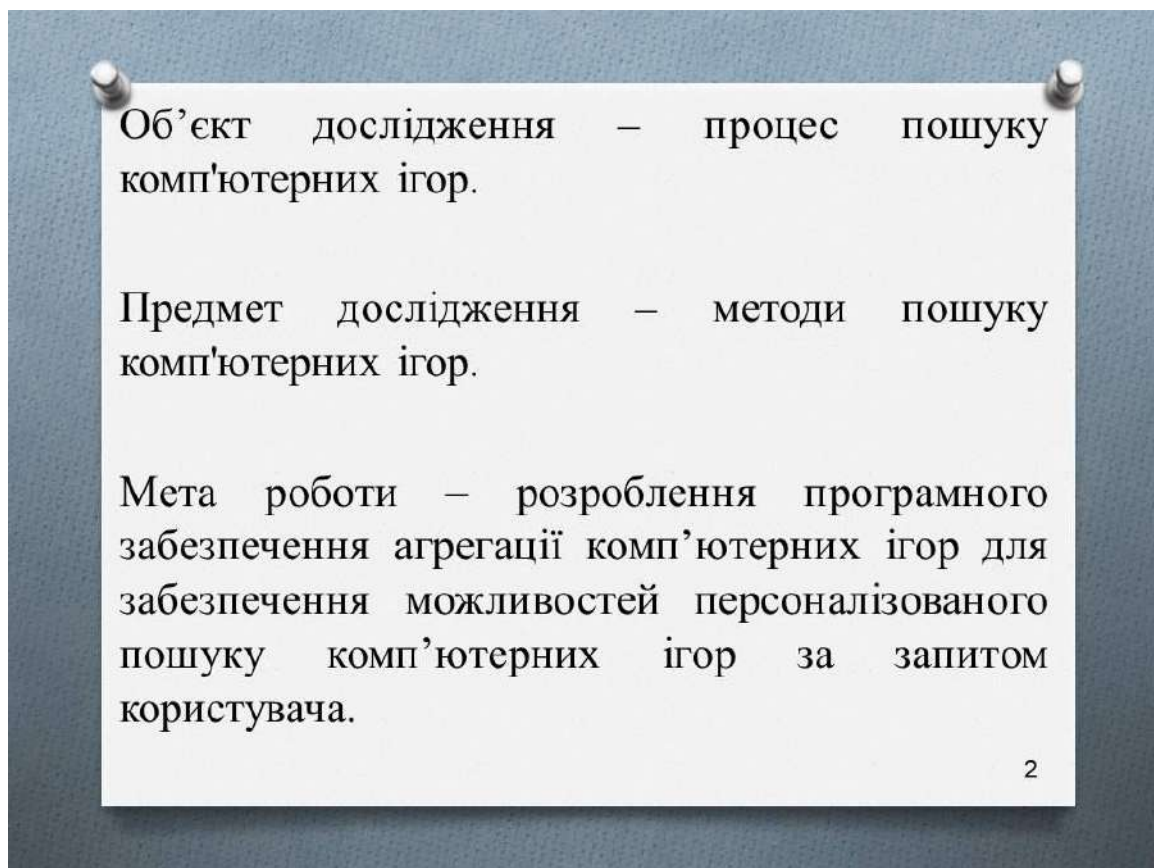


Рисунок В.2 – Слайд 2

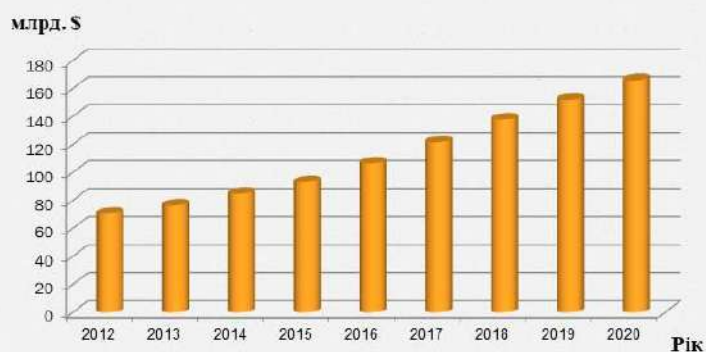
Завдання

- ✓ аналіз проблеми;
- ✓ дослідження методів пошуку комп'ютерних ігор;
- ✓ розробка технічного завдання;
- ✓ вибір засобів розробки;
- ✓ проектування структури програмного забезпечення;
- ✓ розробка програмного забезпечення;
- ✓ тестування та відлагодження програми;
- ✓ економічне обґрунтування доцільності розробки.

3

Рисунок В.3 – Слайд 3

Актуальність проблеми



4

Рисунок В.4 – Слайд 4

Метод пошуку комп'ютерних ігор

- ✓ кластерний аналіз всіх комп'ютерних ігор на основі використання методу k-середніх;
- ✓ визначення значень ознак, яким віддає перевагу користувач на основі набору запитань або за множиною уподобань;
- ✓ визначення віддаленості характеристик ігор від персональних характеристик користувача;
- ✓ формування запиту користувача на пошук ігор;
- ✓ отримання переліку ігор, які відповідають запиту;
- ✓ сортування переліку ігор за віддаленістю від персональних характеристик користувача.

5

Рисунок В.5 – Слайд 5

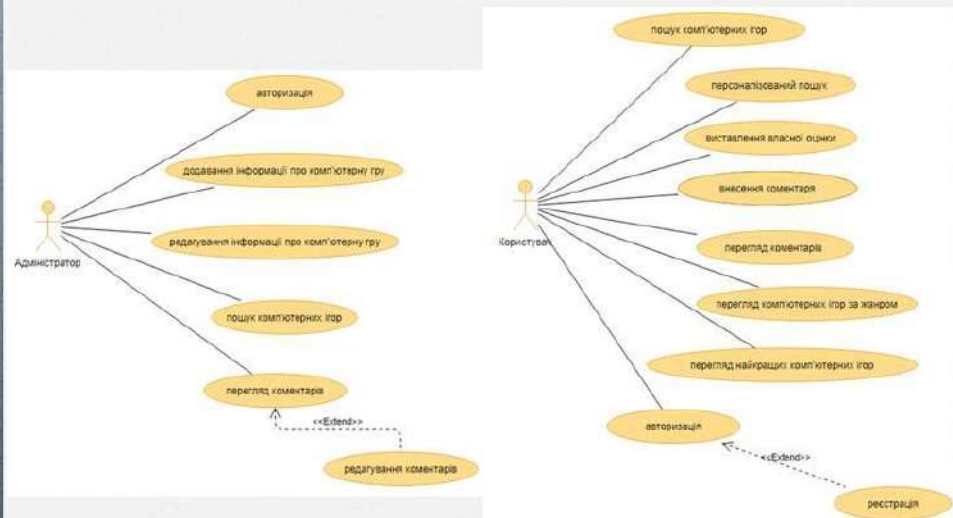
Вибір мови програмування

Показник	C#	C++
Тип мови	мова програмування високого рівня	мова програмування низького рівня
Управління пам'яттю	автоматичне управління пам'яттю	керування пам'яттю вручну
Платформа	може запускатися на різних платформах за рахунок .NET Core	працює на різноманітних платформах
Чиста об'єктно-орієнтованість	+	–
Перевірка границь масивів	+	–
Автоматичний збір сміття	+	–
Цикл foreach	+	–
Тип проєктів	розробки сучасних застосунків	Застосунки, які працюють безпосередньо з апаратним забезпеченням або потребують кращої продуктивності

6

Рисунок В.6 – Слайд 6

Моделювання функціональних вимог до програмного забезпечення



7

Рисунок В.7 – Слайд 7

Структурна схема бази даних



8

Рисунок В.8 – Слайд 8



Рисунок В.9 – Слайд 9

Функціональна схема програми



Рисунок В.10 – Слайд 10

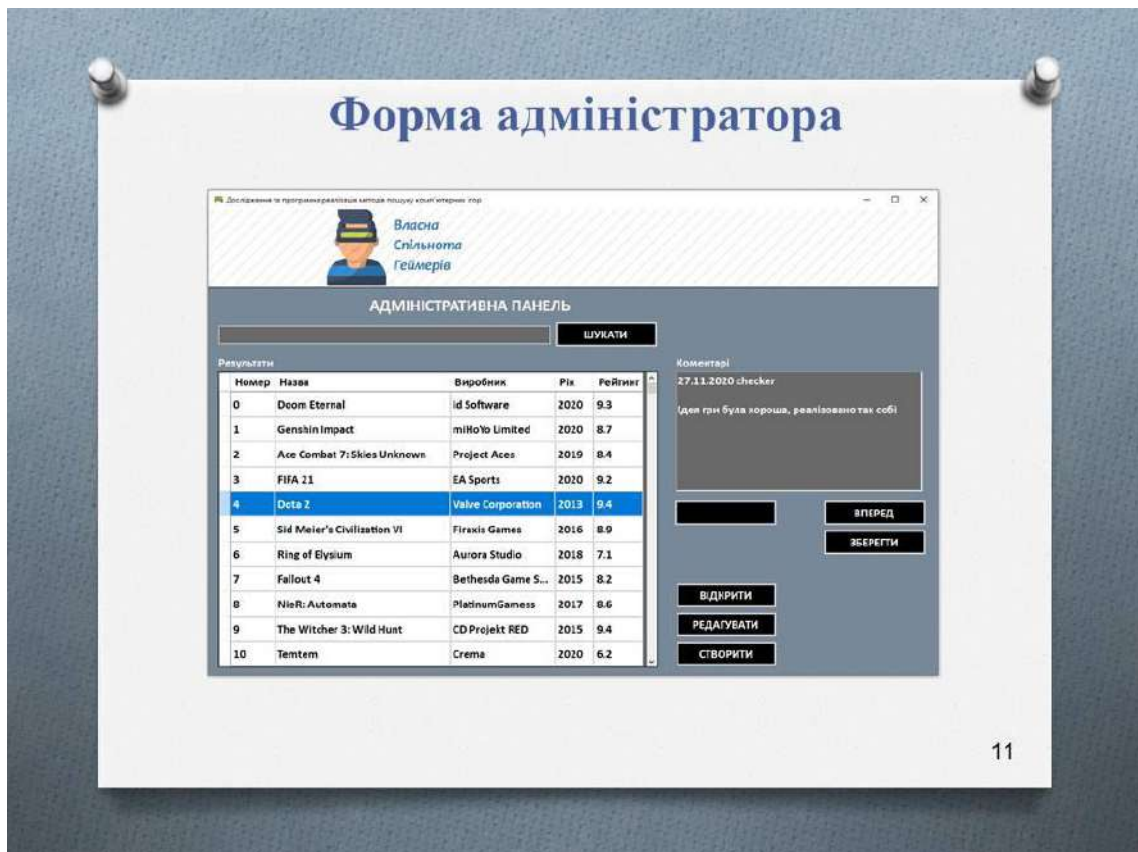


Рисунок В.11 – Слайд 11

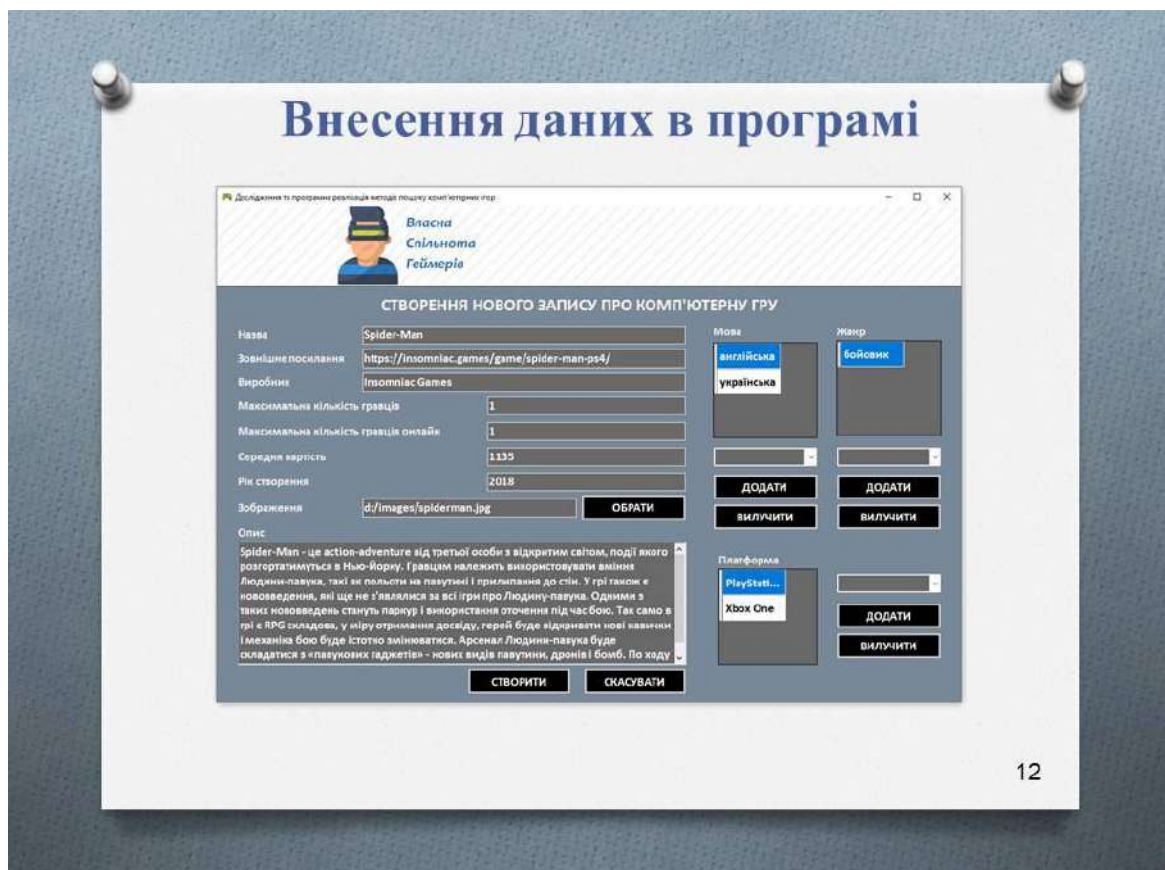


Рисунок В.12 – Слайд 12

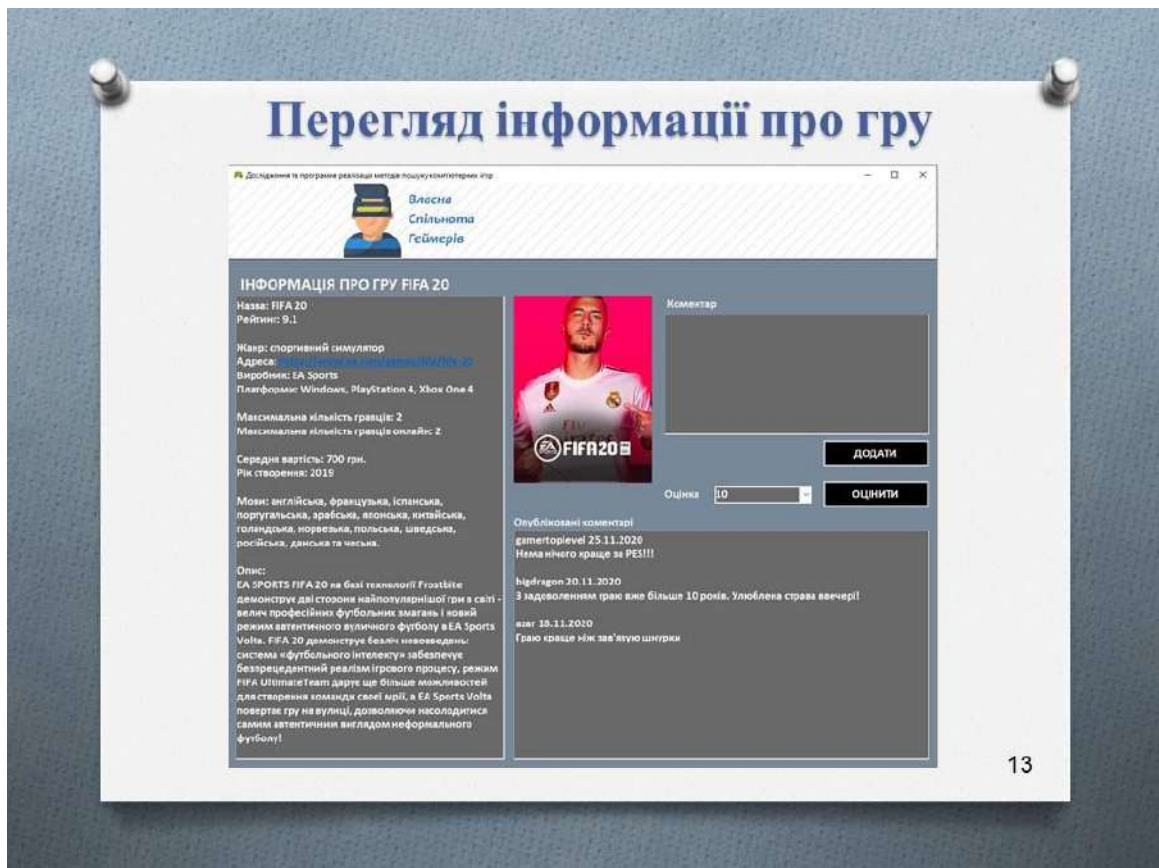


Рисунок В.13 – Слайд 13

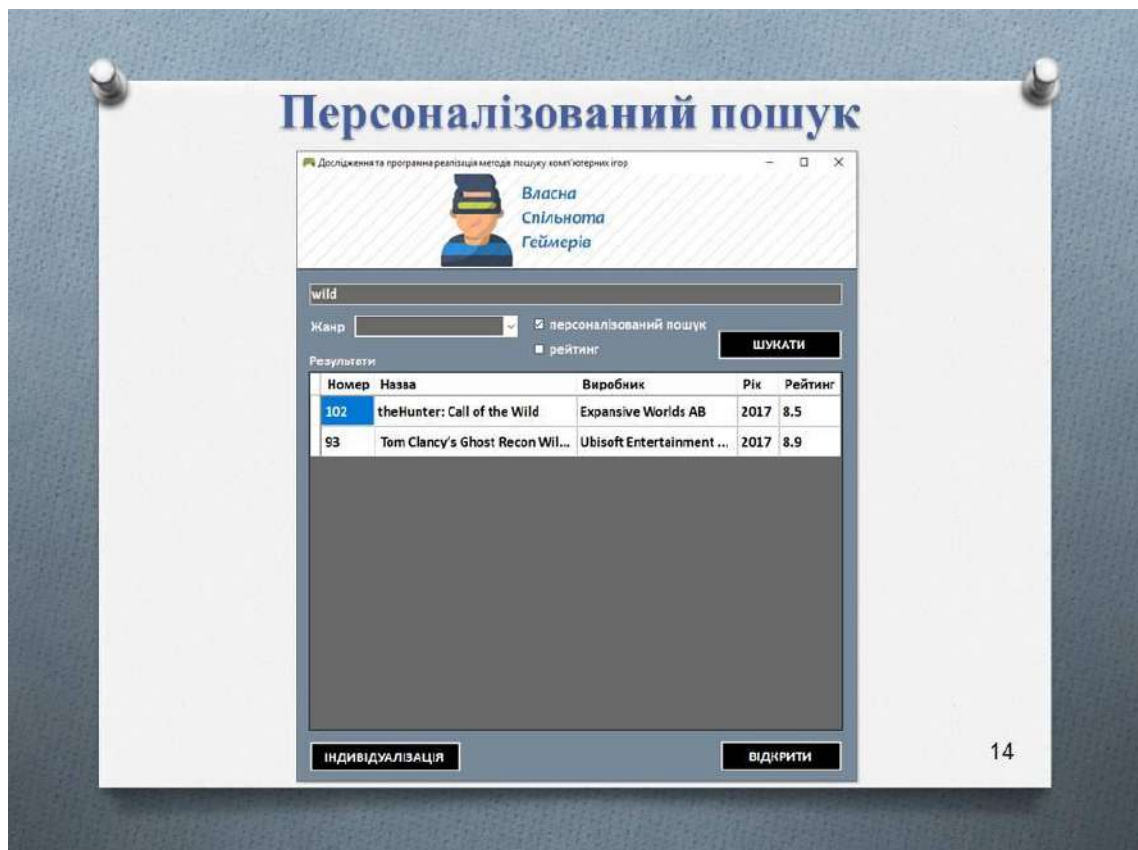


Рисунок В.14 – Слайд 14

Висновки

Виконано вибір мови програмування C# для реалізації програмного забезпечення. Виконано моделювання функціональних вимог до програмного забезпечення у вигляді діаграм прецедентів UML для адміністратора та користувача. Виконано проєктування структури бази даних з вибором Microsoft SQL Server у якості системи керування базою даних та з приведенням структурної схеми бази даних. Виконано проєктування структури застосунку зі створенням діаграми прецедентів UML. Розроблено програмне забезпечення агрегації комп'ютерних ігор для забезпечення можливостей персоналізованого пошуку комп'ютерних ігор за запитом користувача.

Завдання на дипломну кваліфікаційну роботу магістра виконані повністю.

Наукова новизна роботи полягає в тому, що в ній розроблено метод пошуку комп'ютерних ігор, який використовує метод k-середніх для розділення комп'ютерних ігор на кластери, визначає прихильність користувача до ігор відповідного кластеру за відповідями на поставлені запитання або за уподобаними комп'ютерними іграми, що дозволяє виконувати персоналізований пошук у відповідності з тими заченнями характеристик комп'ютерних ігор, яким віддає перевагу користувач.

15

Рисунок В.15 – Слайд 15