НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
комп'ютерних наук
(назва кафедри)

_____ / Голуб Б.Л. /
(підпис) (ПІБ)

"___" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Програмне забезпечення для проведення настільної рольової гри Dungeons & Dragons у вигляді веб-застосунку»

Спеціальність 121 – «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н, доцент _____ Вайганг Г.О.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
асистент _____ _____ Баранова Т. А.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Консультант бакалаврської кваліфікаційної роботи
к.т.н, доцент _____ _____ Даков С.Ю.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Малишко Олександр Юрійович
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерних наук
Голуб Б.Л.

" " 2025 р.

ЗАВДАННЯ
на виконання бакалаврської кваліфікаційної роботи студенту
Малишку Олександру Юрійовичу
(прізвище, ім'я, по батькові)

Спеціальність 121 – «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи Програмне забезпечення для проведення настільної рольової гри Dungeons & Dragons у вигляді веб-застосунку
затверджена наказом ректора НУБіП України від "16"грудня 2024 р. №2249 "с"
Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи
_____
Перелік питань, які потрібно розробити:
- системний аналіз предметної області;
- проєктування інформаційного та програмного забезпечення;
- розробка інформаційного та програмного забезпечення;
- рекомендації щодо впровадження та експлуатації системи

Дата видачі завдання "16" грудня 2024 р.
Керівник бакалаврської кваліфікаційної роботи
_____ _____ Баранова Т.А.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Консультант бакалаврської кваліфікаційної роботи
к.т.н, доцент _____ _____ Даков С.Ю.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____ Малишко О.Ю.
(підпис) (ПІБ студента)

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

1. HPI — настільні рольові ігри
2. D&D — Dungeons & Dragons, найпопулярніша система настільних рольових ігор у світі
3. VTT (Virtual Table Top) — віртуальний ігровий стіл, інтерактивний простір для моделювання тактичних сцен
4. NPC (Non-Player Character) — неігровий персонаж
5. HP (Hit Points) — очки здоров'я персонажа
6. Flask — легкий і гнучкий фреймворк на базі мови Python для створення веб-додатків
7. HTML (HyperText Markup Language) — мова розмітки гіпертексту, стандартна мова розмітки для створення веб-сторінок
8. CSS (Cascading Style Sheets) — каскадні таблиці стилів, мова опису зовнішнього вигляду документа
9. JavaScript — мова програмування, що використовується для створення інтерактивних веб-сторінок
10. AJAX (Asynchronous JavaScript and XML) — технологія асинхронного оновлення даних, що підвищує інтерактивність та швидкодію системи
11. JSON (JavaScript Object Notation) — текстовий формат обміну даними, що використовується для зберігання неструктурованих даних
12. API (Application Programming Interface) — інтерфейс програмування додатків, набір визначень взаємодії різнотипного програмного забезпечення
13. SQL (Structured Query Language) — мова структурованих запитів, що використовується для роботи з реляційними базами даних
14. SQLite — компактна вбудована реляційна база даних
15. PostgreSQL — об'єктно-реляційна система керування базами даних

16. NoSQL — підхід до проектування баз даних, що не використовує табличну схему відношень
17. ORM (Object-Relational Mapping) — технологія програмування, що зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування
18. ACID (Atomicity, Consistency, Isolation, Durability) — набір властивостей, що гарантують надійну роботу транзакцій бази даних
19. UML (Unified Modeling Language) — уніфікована мова моделювання, використовується для візуалізації, специфікації, конструювання і документування програмних систем
20. Дайс (dice) — гральні кубики, які використовуються для випадкового визначення результатів у настільних іграх, зокрема в Dungeons & Dragons.

ВСТУП

У сучасному цифровому середовищі настільні рольові ігри (HPI), зокрема Dungeons & Dragons (D&D), зазнають значної трансформації. Завдяки розвитку інформаційних технологій та зростанню попиту на інтерактивні форми дозвілля, з'являється потреба у спеціалізованих інструментах, що забезпечують зручну та гнучку організацію ігрового процесу у дистанційному форматі. Особливо це стало актуальним в умовах розширення форматів дистанційної взаємодії, які охоплюють не лише роботу чи навчання, але й соціальну комунікацію та дозвілля. Враховуючи популярність Dungeons & Dragons у всьому світі, включаючи українське ігрове середовище, існує нагальна потреба у створенні доступного, локалізованого та зручного у використанні програмного рішення.

Метою цього дипломного проєкту є розробка веб-застосунку, що забезпечує повний цикл підготовки та проведення настільної рольової гри Dungeons & Dragons. Такий додаток має включати функціонал для створення і управління ігровими кампаніями, додавання персонажів (гравців та неігрових), відстеження подій, зберігання інформації про локації, а також реалізацію віртуального ігрового столу з можливістю маніпулювання об'єктами на полі. Важливим аспектом є доступність застосунку у веб-форматі без потреби встановлення додаткового програмного забезпечення, що значно розширює коло потенційних користувачів.

Розробка системи здійснювалася із застосуванням сучасного стеку технологій, що забезпечує як надійність, так і масштабованість програмного продукту. Для серверної частини було обрано фреймворк Flask — легкий і гнучкий інструмент на базі мови Python, який дозволяє швидко створювати веб-додатки. В якості інтерфейсної частини використано HTML(HyperText Markup Language), CSS(Cascade Style Sheet) та JavaScript, зокрема з використанням технології AJAX(Asynchronous JavaScript and XML) для асинхронного оновлення даних, що підвищує інтерактивність та швидкодію системи. Для ефективного обслуговування

клієнтських запитів та розгортання проєкту у продакшн-середовищі використано Flask Web-Server. Зберігання структурованих даних реалізовано у вигляді бази даних, що дозволяє гнучко управляти інформацією про кампанії, персонажів та події. Окремо передбачено збереження неструктурованих даних у форматі JSON для простоти оновлення та обробки даних у рамках знань, доступних гравцям.

Таким чином, проєкт не лише задовольняє нагальні потреби користувачів у галузі цифрового дозвілля, але й демонструє можливості використання інформаційних технологій у створенні доступних інструментів для організації складної, багатокористувацької взаємодії в контексті рольових ігор.

Основні положення даної роботи були представлені у вигляді тез доповіді на VII Всеукраїнській науково-практичній конференції студентів і аспірантів «Теоретичні та прикладні аспекти розробки комп’ютерних систем» (НУБіП України, Київ, 2025) [14].

1. СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметна область дипломного проєкту зосереджена на створенні інтегрованого веб-середовища для проведення настільних рольових ігор, зокрема Dungeons & Dragons (D&D) – найпопулярнішої системи HPI у світі. Цей жанр ігор характеризується колективним створенням наративу, керованого правилами, де гравці приймають рішення від імені вигаданих персонажів у фентезійному світі, а майстер гри (Dungeon Master) описує наслідки цих рішень та керує загальним сюжетом.

Традиційно D&D вимагає фізичної присутності учасників, використання паперових персонажів, фізичних кубиків і друкованих довідників. Однак сучасні тенденції демонструють зростаючий попит на цифрові інструменти, які дозволяють перенести гру в онлайн-простір без втрати ключових елементів взаємодії. Цей перехід посилився під час пандемії COVID-19, коли фізичні зустрічі стали проблематичними, і водночас розкрив потенціал віддаленої гри для об'єднання людей незалежно від їхнього географічного розташування.

Аналіз показує, що український ринок відчуває нестачу локалізованих інструментів для віртуального проведення настільних рольових ігор, які б одночасно відповідали трьом ключовим критеріям:

1. Забезпечували повний цикл від створення персонажа до завершення кампанії
2. Мали україномовний інтерфейс та термінологію
3. Залишалися доступними для новачків, які лише знайомляться з D&D

1.2 Аналіз вимог до програмної системи

На основі аналізу технічного завдання можна виділити наступні основні абстракції предметної області, які формують концептуальний каркас системи:

1. Користувацький профіль

Централізована сутність, що представляє учасника системи з унікальними обліковими даними. Користувач може виступати в ролі майстра (організатора ігрових сесій) або гравця (активного учасника). Система аутентифікації забезпечує розмежування прав доступу до функціоналу та контенту.

2. Ігрова кампанія

Логічний контейнер, що об'єднує серію пов'язаних ігрових сесій у єдиний наратив. Кампанія включає:

- Список учасників
- Колекцію локацій
- Базу неігрових персонажів (NPCs) і антагоністів
- Хронологію ключових подій і рішень гравців

Кампанія представляє собою замкнену екосистему з власними правилами та обмеженнями, встановленими майстром гри, і може існувати паралельно з іншими кампаніями, не перетинаючись з ними на рівні ігрової механіки.

3. Ігровий персонаж

Мультиатрибутна сутність, що представляє аватар гравця у віртуальному світі. Персонаж характеризується комплексом взаємопов'язаних показників:

- Базові характеристики (сила, спритність, витривалість, інтелект, мудрість, харизма)
- Похідні параметри (клас броні, ініціатива, швидкість переміщення)

- Рівень досвіду та професіональні навички
- Інвентар з обладнанням та магічними предметами
- Репертуар способів атаки
- Біографія, світогляд та зовнішній вигляд

Життєвий цикл персонажа тісно пов'язаний з прогресом у кампанії: він набуває досвіду, розвиває здібності, отримує нове спорядження та модифікує свої характеристики відповідно до правил системи D&D.

4. Віртуальний ігровий стіл (VTT)

Інтерактивний простір для моделювання тактичних сцен, який синхронізує візуальну та механічну складові гри. Стіл містить:

- Масштабовану карту з координатною сіткою
- Маніпульовані токени персонажів і істот
- Інструменти вимірювання відстані та зон ефекту
- Панель ініціативи для відстеження черговості ходів
- Інтерфейс швидкого доступу до характеристик

VTT виступає основним середовищем для розв'язання бойових та дослідницьких задач, забезпечуючи просторову репрезентацію взаємодії між сутностями гри.

5. Система симуляції випадковості (кубики)

Механізм моделювання стохастичних процесів, що лежить в основі правил D&D. Включає:

- Генератор псевдовипадкових чисел для імітації кидків фізичних кубиків
- Інтерфейс формування складних комбінацій (напр., 2d6+4)
- Модифікатори кидків (перевага, завада, критичні успіхи/провали)

Система кубиків забезпечує елемент непередбачуваності, необхідний для підтримання балансу між детермінованою структурою правил та наративною свободою гравців.

6. Посібник

Структурований каталог довідкових матеріалів, що охоплює всі аспекти правил та ігрового світу:

- Бібліотека класів, рас, здібностей та особливостей
- Каталог предметів, зброї та магічних артефактів
- Бестіарій з параметрами істот та монстрів
- Правила різних ігрових механік (бій, дослідження, відпочинок)

База знань функціонує як централізоване сховище інформації, що забезпечує швидкий доступ до потрібних правил під час гри без необхідності консультуватися з фізичними книгами.

Нефункціональні вимоги

1. Продуктивність

- Система повинна підтримувати одночасну гру для **щонайменше 10 користувачів** без помітних затримок.
- Відгук інтерфейсу на дії користувача (переміщення токенів, зміна HP) не повинен перевищувати **200 мс**.
- Час завантаження сторінки не повинен перевищувати **3 секунд** при середньому навантаженні.

2. Масштабованість

- Платформа повинна бути розроблена з можливістю **збільшення кількості користувачів** без значного зниження продуктивності.

- Дані кампаній та персонажів повинні зберігатися у **базі даних**, що дозволяє горизонтальне масштабування.

3. Надійність

- У разі помилки або збою система повинна **автоматично зберігати поточний стан гри**.
- Переривання сеансу не повинно призводити до втрати даних.

4. Юзабіліті

- Інтерфейс повинен бути **інтуїтивно зрозумілим**, адаптованим для як досвідчених, так і нових гравців.
- Основні функції (створення персонажа, переміщення токенів, кидки кубів) повинні виконуватися **не більше ніж у 3 кліки**.
- Система повинна підтримувати **мобільні пристрої**.

5. Сумісність

- Підтримка основних сучасних браузерів (Chrome, Firefox, Edge, Safari).
- Підтримка різних операційних систем (Windows, macOS, Linux).

Розглянута предметна область являє собою складну, багаторівневу систему взаємодій між користувачами, персонажами та ігровими механіками в контексті колективного творення наративу. Запропонована концептуальна модель враховує як традиційні аспекти настільних рольових ігор, так і сучасні вимоги до онлайн-платформ, забезпечуючи комплексне рішення для переведення D&D у цифровий формат без втрати ключових елементів соціальної взаємодії та творчої свободи.

1.3 Моделювання предметної області

Моделювання предметної області виконано з використанням різних типів UML(Unified Modeling Language) -діаграм для всебічного аналізу системи:

Діаграми прецедентів (Use Case)

Діаграма прецедентів (рис. 1), демонструє взаємодію користувачів із системою на високому рівні абстракції. Вона відображає основні сценарії використання веб-застосунку для настільної рольової гри Dungeons & Dragons і показує два основних актори системи: звичайного користувача та майстра підземель (Dungeon Master).

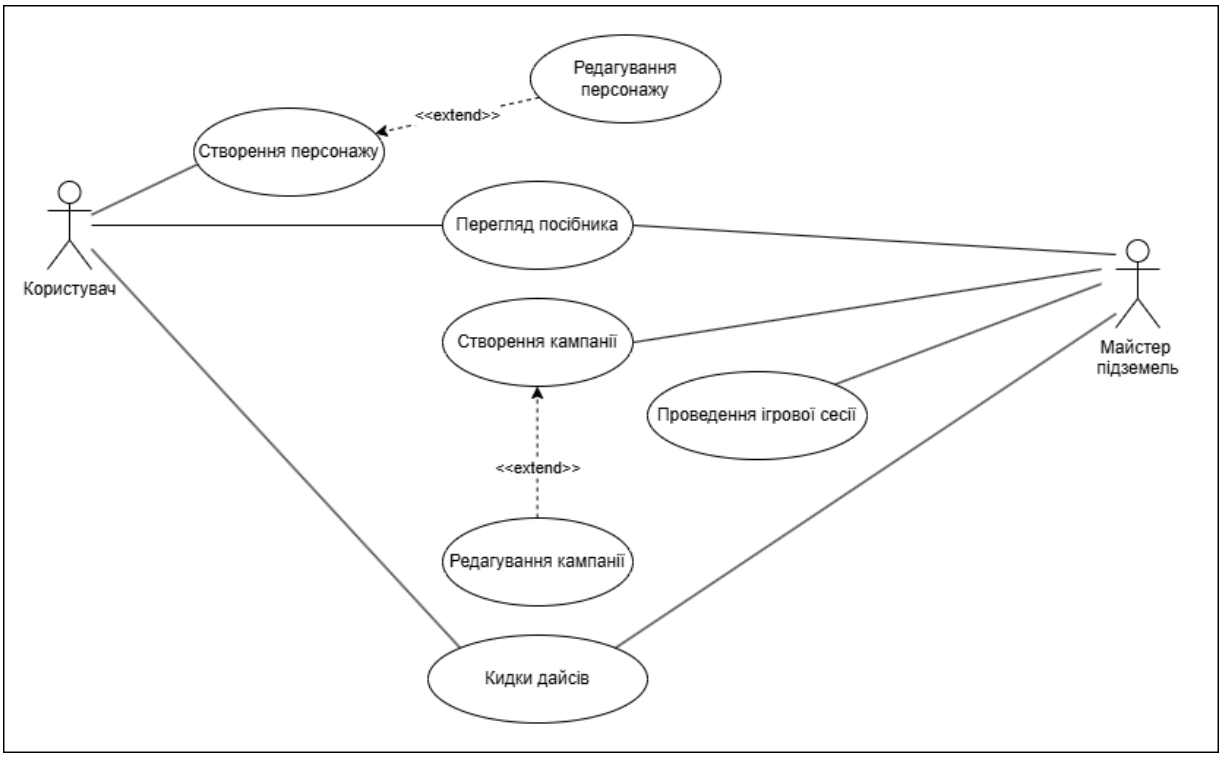

Рис. 1. Діаграма прецедентів

Користувач має доступ до таких функцій:

- Створення персонажа з подальшим розширенням до можливості його редагування

- Перегляд посібника з правилами та довідковими матеріалами
- Кидки дайсів для визначення результатів ігрових дій

Майстер підземель має розширені повноваження:

- Створення та редагування кампаній, що є контейнерами для ігрових сесій
- Проведення ігрових сесій із використанням віртуального столу
- Перегляд посібника для звернення до правил
- Кидки дайсів для визначення результатів дій неігрових персонажів

Діаграма чітко відображає ієрархію прецедентів через зв'язки «extend», що показують опціональне розширення базової функціональності. Такий підхід дозволяє відобразити модульну природу системи, де користувачі можуть починати з простих операцій (створення персонажа) і поступово переходити до складніших взаємодій (редагування характеристик).

Діаграми послідовності (Sequence)

Діаграма послідовності (рис. 2), яка деталізує часовий аспект взаємодії між компонентами системи. Вона відображає комунікацію між гравцем, майстром підземель та основними модулями застосунку

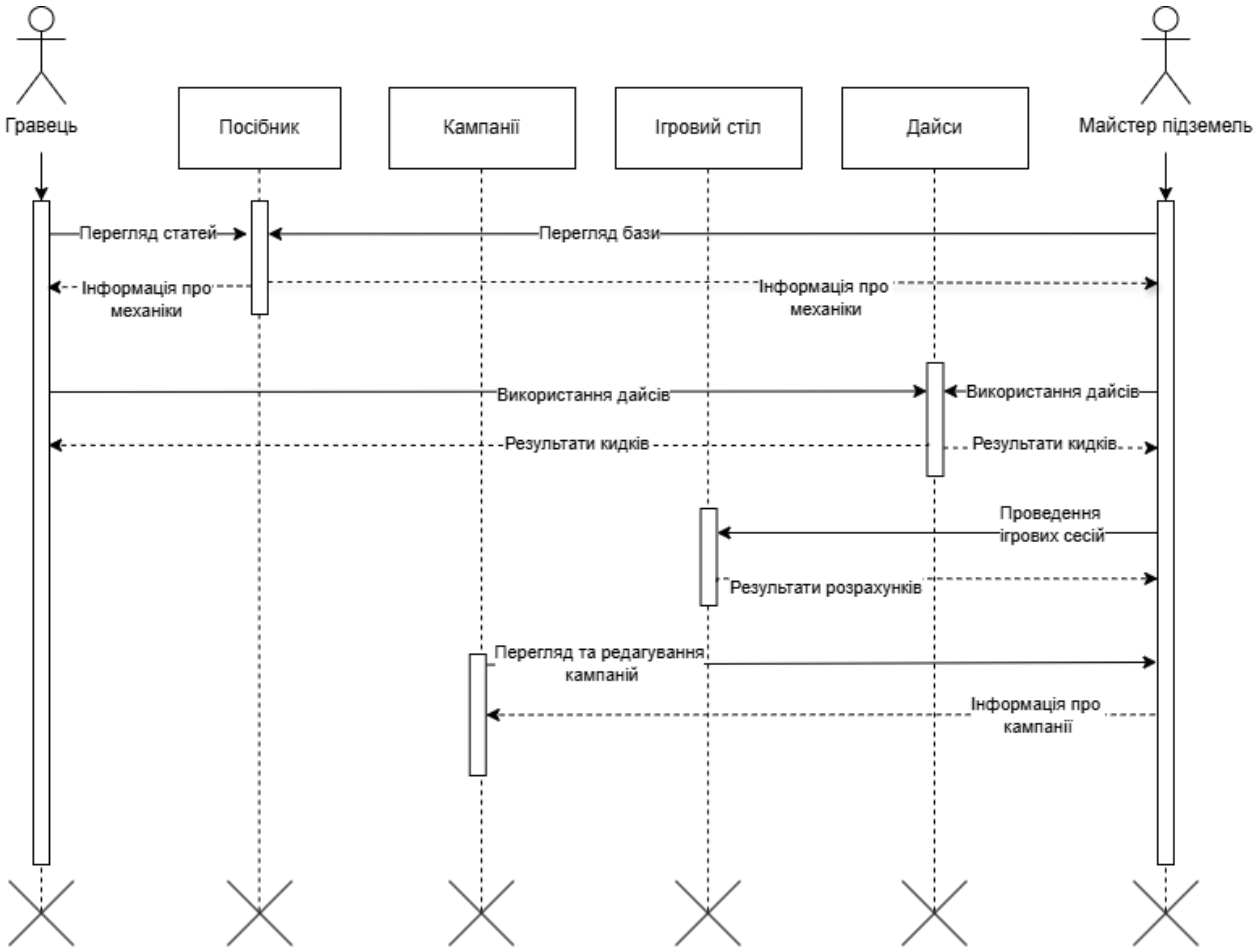

Рис. 2. Діаграма послідовності

- Модуль персонажів
- Посібник
- Кампанії
- Віртуальний стіл
- Система дайсів

Діаграма послідовності ілюструє комплексний процес інформаційного обміну, де:

1. Гравець переглядає та редагує персонажів, отримуючи результати розрахунків характеристик

2. Обидва типи користувачів можуть звертатись до посібника для отримання інформації про механіки гри

3. Майстер проводить ігрові сесії, використовуючи систему дайсів

4. Результати кидків та розрахунків передаються між усіма учасниками процесу

Особливу увагу варто звернути на двонаправлені потоки даних, що демонструють інтерактивну природу системи - кожна дія отримує відповідну реакцію та оновлення стану, що забезпечує динамічність ігрового процесу.

Діаграми активності (Activity)

Рисунки 3-8 представляють діаграми активності для ключових процесів системи:

Робота з дайсами (рис. 3)

Діаграма відображає послідовність дій користувача при проведенні випадкових розрахунків:

1. Перегляд головної сторінки
2. Перехід до дайсів
3. Виконання кидків
4. Отримання випадкових значень

Такий процес забезпечує користувачам швидкий доступ до основного механізму невизначеності у грі без необхідності здійснення додаткових налаштувань.

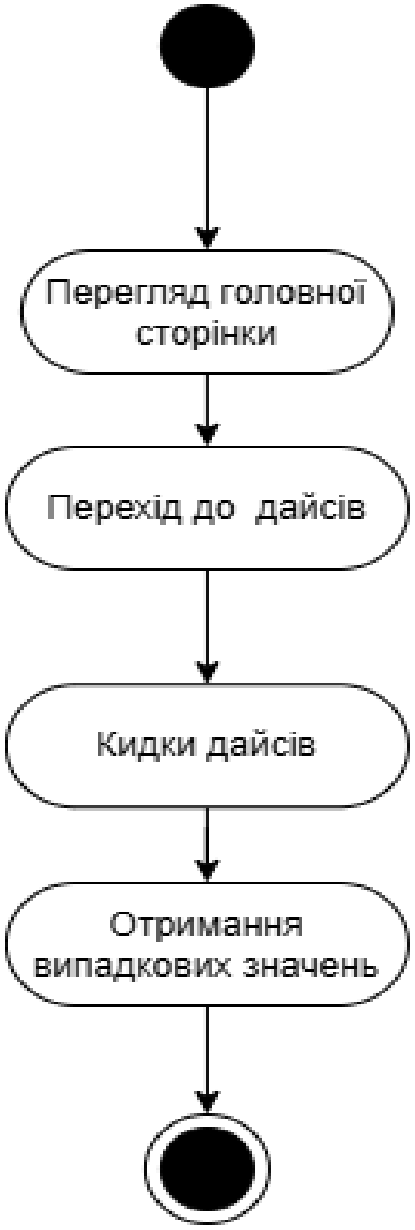

Рис. 3. Діаграма активності – використання дайсів

Перегляд бази знань (Рис. 4)

Послідовність дій включає:

1. Перехід до бази знань
2. Пошук необхідної статті
3. Перегляд конкретного матеріалу

Структура діаграми підкреслює лінійність процесу пошуку інформації, що сприяє швидкому доступу до потрібних правил під час гри.

Рис. 4. Діаграма активності – використання бази знань

Робота з персонажами (Рис. 5)

Діаграма демонструє процес з точками прийняття рішень:

1. Перехід до розділу персонажів
2. Перевірка наявності персонажа через умовний блок
3. Розгалуження на створення нового або перехід до існуючого персонажа
4. Робота з персонажем (редагування характеристик)
5. Збереження параметрів

Такий підхід забезпечує гнучкість в управлінні ігровими аватарами та підтримує життєвий цикл персонажа.

Рис. 5. Діаграма активності – управління персонажем

Робота з кампаніями (Рис. 6)

Процес взаємодії з кампаніями включає:

1. Перехід до розділу кампаній
2. Перевірку наявності кампаній
3. Розгалуження на створення нової або перехід до існуючої
4. Редагування інформації
5. Збереження змін

Діаграма відображає модульний підхід до управління сюжетними контейнерами з чітким розмежуванням етапів створення та редагування.

Рис. 6. Діаграма активності – управління кампаніями

Робота з ігровим столом (Рис. 7)

Ця діаграма відображає складний процес організації ігрової сесії:

1. Перехід до віртуального столу
2. Перевірка наявності активної сесії
3. Створення нової або приєднання до існуючої

4. Проведення сесії
5. Умовний блок для вибору збереження стану
6. Завершення з або без збереження

Процес враховує циклічну природу ігрових сесій з можливістю збереження прогресу для подальшого продовження.

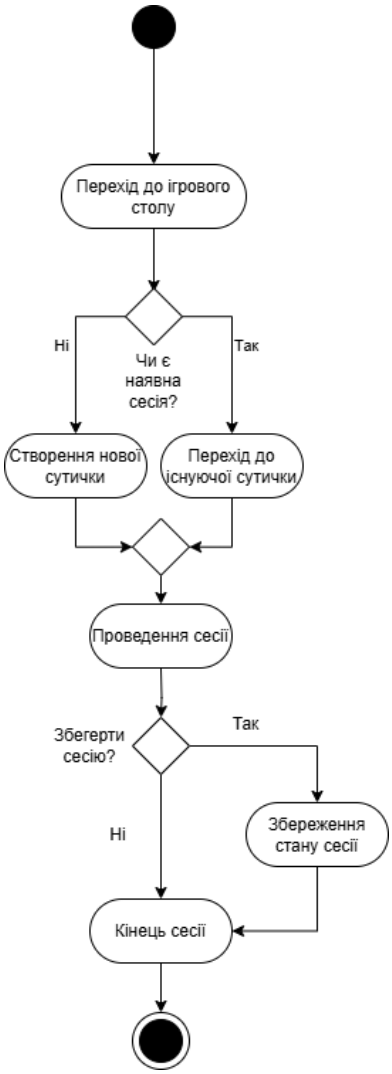

Рис. 7. Діаграма активності – робота з сесією

Проведення ігрової сесії (Рис. 8)

Найбільш детальна діаграма показує послідовність активностей під час активної гри:

1. Перехід до сесії
2. Додавання ворогів та загроз
3. Додавання об'єктів оточення
4. Проведення кидків ініціативи для встановлення черговості ходів
5. Циклічний процес ходів з переміщенням і діями
6. Завершення при вирішенні ситуації

Рис. 8. Діаграма активності – проведення сесії

На зображенні(рис. 9) представлена діаграма абстракцій, що визначає ключові сутності системи та їх властивості:

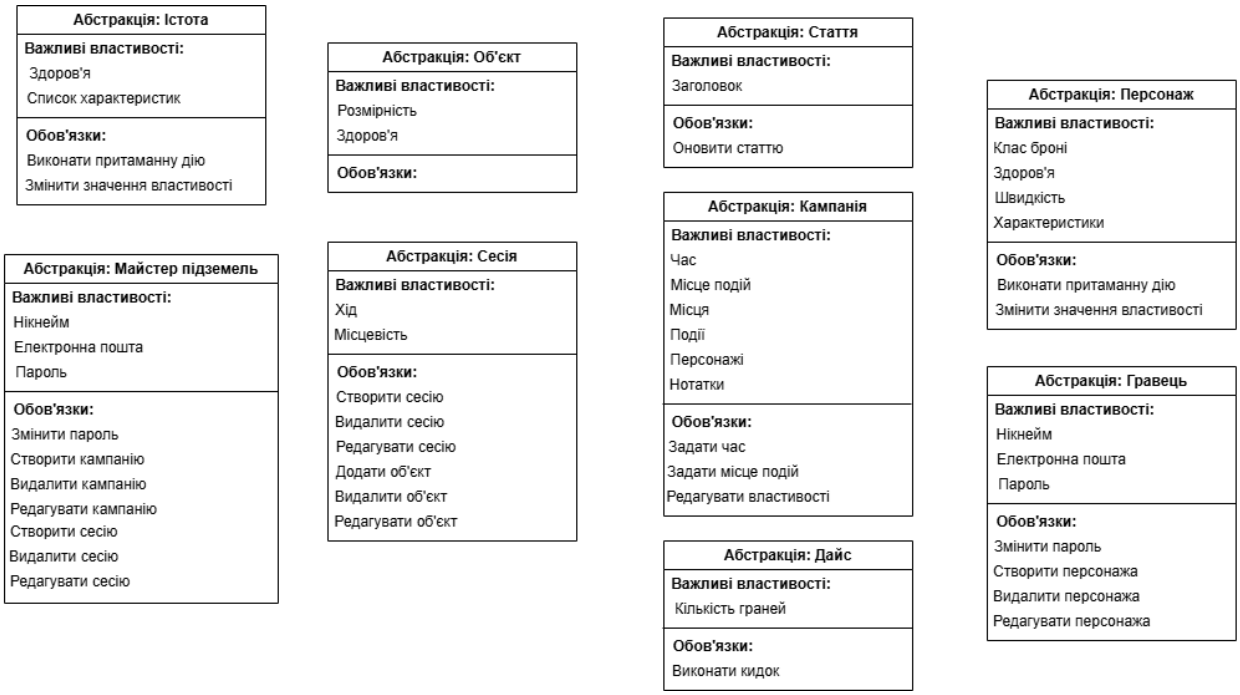

Рис. 9. Діаграма абстракцій

- Істота - базова абстракція з властивостями здоров'я(HP) та характеристик.
- Об'єкт - абстракція для неживих елементів ігрового світу.
- Майстер підземель - користувач з розширеними правами для управління кампаніями.
- Гравець - звичайний користувач, що керує персонажами.
- Персонаж - сутність з характеристиками, що представляє аватар гравця.
- Сесія - абстракція для конкретної ігрової зустрічі з учасниками.
- Кампанія - контейнер для пов'язаних сесій з сюжетною лінією.
- Стаття - елемент бази знань з описом правил або світу.
- Дайс - абстракція для генератора випадкових чисел певного діапазону.

Кожна абстракція має визначені властивості та обов'язки, що формують цілісну об'єктну модель системи з чітким розподілом відповідальності.

Представлені діаграми створюють вичерпну модель предметної області, що охоплює статичні аспекти (абстракції та їх зв'язки), динамічні взаємодії (послідовності обміну повідомленнями) та процеси (потоки активностей). Така багатогранна візуалізація дозволяє розглянути систему з різних перспектив і сформувати структуровану основу для подальшого проєктування і розробки.

1.4. Огляд інформаційних джерел та існуючих рішень

У процесі розробки веб-застосунку для проведення *Dungeons & Dragons*, було проаналізовано низку сучасних цифрових інструментів, які вже існують на ринку. Серед найбільш популярних та функціональних варто відзначити платформи **Owlbear Rodeo**, **Roll20** та **5e Companion**. Кожна з них має свої переваги та недоліки, які прямо впливають на зручність користування, доступність і можливості адаптації до локальних потреб, зокрема української аудиторії.

Owlbear Rodeo позиціонує себе як легкий та інтуїтивний віртуальний ігровий стіл. Його головна перевага полягає у простоті — платформа дозволяє швидко розгортати ігрові сесії без потреби в складних налаштуваннях або завантаженні додаткових даних. Завдяки мінімалістичному інтерфейсу, Owlbear Rodeo підходить для проведення швидких імпровізаційних ігор або коротких кампаній. Проте платформа має суттєві обмеження: відсутність вбудованих інструментів для управління персонажами, обмежена взаємодія з базами знань, а також повна відсутність україномовної локалізації.

Roll20 — це одна з найбільш масштабних і популярних платформ у світі для гри в D&D та інші TTRPG (Tabletop Role-Playing Games). Вона надає користувачам доступ до великого набору інструментів, включаючи інтерактивні карти,

автоматизовані кидки кубиків, підтримку макросів та скриптів, інтеграцію з офіційним контентом і розширеннями. Разом з тим, функціональна насиченість платформи створює складнощі для новачків. Частина функціоналу доступна лише у платній версії, що створює бар'єр для широкого використання. Крім того, Roll20 не має україномовного інтерфейсу та вимагає стабільного високошвидкісного інтернет-з’єднання.

5e Companion орієнтований переважно на підтримку п’ятої редакції Dungeons & Dragons, забезпечуючи зручні інструменти для створення й управління персонажами. Його перевага полягає в мобільній доступності та спрощеному інтерфейсі. Однак система майже повністю зосереджена на персональному менеджменті персонажів, надаючи обмежені можливості в контексті віртуального столу та спільного проведення повноцінних кампаній. Відсутність українізації знижує її придатність для місцевої спільноти.

Загальний аналіз показує, що попри наявність різноманітних платформ, жодне з існуючих рішень не задовольняє одночасно трьох ключових вимог, визначених для українського користувача:

- **Повний цикл гри**: більшість платформ або зосереджені лише на візуальній частині сесії (ігровому полі), або лише на механіці персонажів, тоді як комплексних рішень із взаємодією між всіма компонентами (НПС, події, карти, ініціатива тощо) бракує.
- **Україномовна локалізація**: жодна з платформ не підтримує повну або часткову локалізацію інтерфейсу українською мовою, що обмежує доступність інструменту для широкої аудиторії, зокрема молоді та новачків.
- **Доступність для новачків**: складність інтерфейсів, перевантаженість функціями та потреба в технічних знаннях ускладнюють процес

залучення нових користувачів, які не мають досвіду у використанні TTRPG-платформ.

До того ж, важливо підкреслити, що більшість наявних платформ не адаптовані до специфіки українського ігрового середовища. Відсутність термінологічної адаптації, підтримки місцевих ігрових спільнот та зручних вбудованих інструментів робить актуальною розробку нового продукту, орієнтованого саме на локального користувача.

Таким чином, наша система має потенціал заповнити наявну нішу, запропонувавши повністю україномовний веб-застосунок, який поєднує основні функції платформ для гри в D&D, при цьому залишаючись доступним, гнучким та простим у використанні.

1.5 Постановка завдання

На основі проведеного аналізу предметної області, вимог та існуючих рішень, завдання проєкту формулюється наступним чином:

Мета: Розробити веб-застосунок, що забезпечує повний цикл підготовки та проведення настільної рольової гри Dungeons & Dragons з україномовним інтерфейсом та орієнтацією на доступність для новачків.

Основні завдання:

1. Реалізувати систему управління користувачами з розмежуванням ролей (гравець, майстер підземель).
2. Створити функціонал для роботи з персонажами:
    - Створення персонажів із вибором класу, раси та характеристик
    - Редагування параметрів персонажа
    - Автоматичний розрахунок похідних характеристик
- Розробити систему управління кампаніями:

- Створення та ведення кампаній
- Збереження хронології подій
- Управління неігровими персонажами

- Реалізувати віртуальний ігровий стіл з:
  - Підтримкою карт з масштабуванням
  - Маніпуляцією токенами персонажів
  - Системою відстеження ініціативи

3. Впровадити систему генерації випадкових значень (кидки кубиків):
   - Підтримка різних типів кубиків
   - Формування складних формул кидків
- Створити посібник із правилами та довідковою інформацією:
  - Інформація про класи, раси, здібності
  - Опис ігрових механік
  - Довідник по істотам
- Забезпечити технічні аспекти реалізації:
  - Відмовостійкість системи
  - Адаптивний інтерфейс
  - Автоматичне збереження стану сесій

Розробка веб-застосунку здійснюватиметься з використанням технологічного стеку:

- Серверна частина: Python з фреймворком Flask
- Клієнтська частина: HTML5, CSS3, JavaScript
- Зберігання даних: PostgreSQL та JSON-формат

2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

ER-діаграма(рис. 10) відображає реляційну структуру бази даних для основних сутностей системи. Ця частина бази даних реалізована з використанням SQL(Structured Query Language) підходу, що забезпечує структуровану організацію даних з чіткими зв'язками між таблицями.

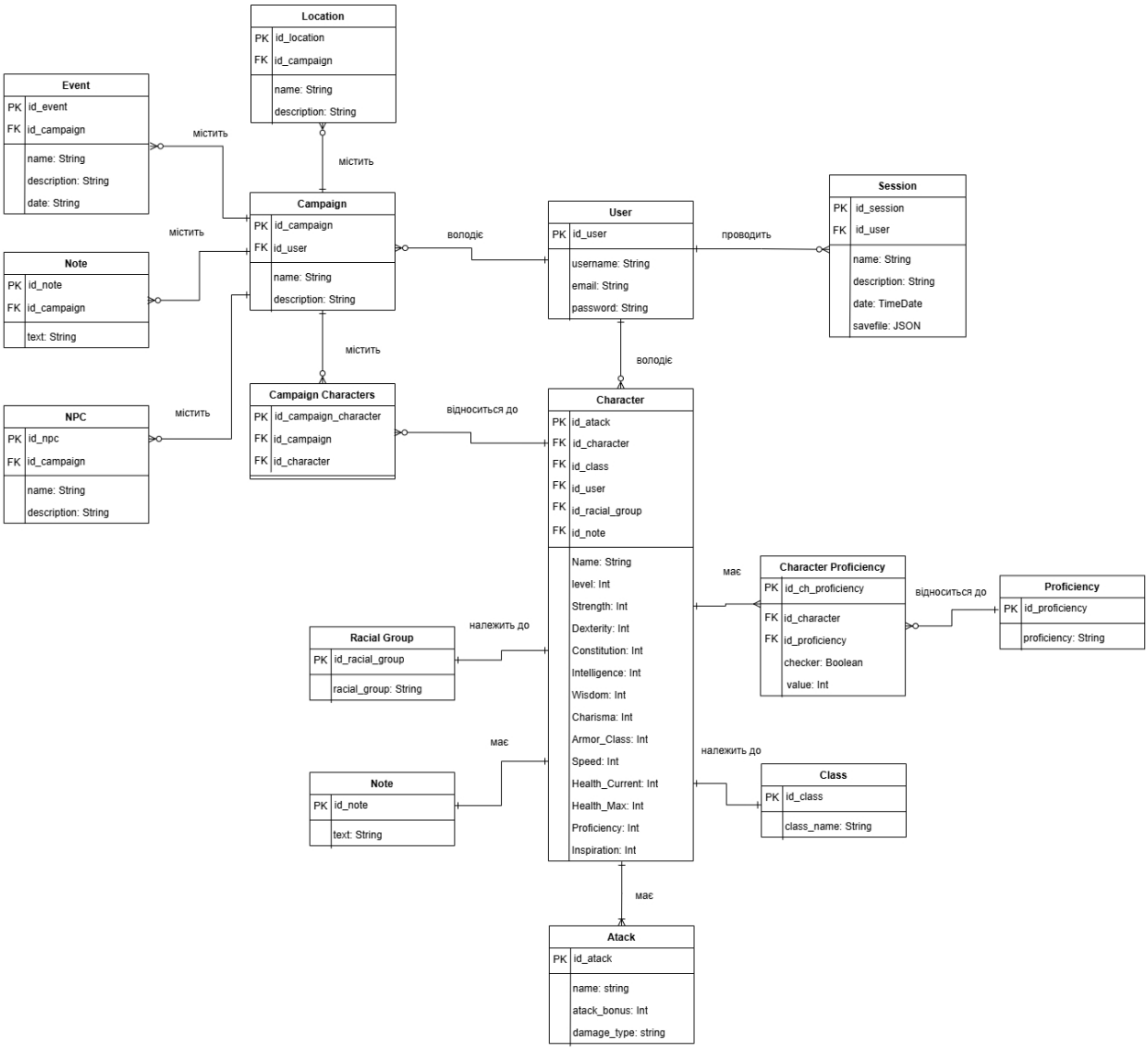

Рис. 10. ER-діаграма SQL структури бази даних

Основні сутності та їх атрибути:

1. User (Користувач)

- id_user (PK): унікальний ідентифікатор користувача
- username: ім'я користувача для входу в систему
- email: електронна пошта для комунікації та відновлення доступу
- password: зашифрований пароль для автентифікації

2. Campaign (Кампанія)

- id_campaign (PK): унікальний ідентифікатор кампанії
- id_user (FK): зовнішній ключ, що вказує на користувача-власника кампанії
- name: назва кампанії
- description: опис сюжету та особливостей кампанії

3. Character (Персонаж)

- id_character (PK): унікальний ідентифікатор персонажа
- id_user (FK): зовнішній ключ, що вказує на користувача-власника персонажа
- id_class (FK): зовнішній ключ, що вказує на клас персонажа
- id_racial_group (FK): зовнішній ключ, що вказує на расову групу персонажа
- id_note (FK): зовнішній ключ, що вказує на нотатки про персонажа
- id_attack (FK): зовнішній ключ, що вказує на атаки персонажа
- Name: ім'я персонажа
- level: рівень персонажа
- Strength: показник сили
- Dexterity: показник спритності
- Constitution: показник статури

- Intelligence: показник інтелекту
- Wisdom: показник мудрості
- Charisma: показник харизми
- Armor_Class: клас броні
- Speed: швидкість переміщення
- Health_Current: поточне здоров'я
- Health_Max: максимальне здоров'я
- Proficiency: показник майстерності
- Inspiration: натхнення

4. Session (Сесія)

- id_session (PK): унікальний ідентифікатор сесії
- id_user (FK): зовнішній ключ, що вказує на користувача-учасника сесії
- name: назва сесії
- description: опис подій сесії
- date: дата проведення
- savefile: JSON-файл зі збереженням стану сесії

5. Location (Локація)

- id_location (PK): унікальний ідентифікатор локації
- id_campaign (FK): зовнішній ключ, що вказує на кампанію, до якої належить локація
- name: назва локації
- description: опис локації

6. Event (Подія)

- id_event (PK): унікальний ідентифікатор події
- id_campaign (FK): зовнішній ключ, що вказує на кампанію, до якої належить подія

- name: назва події
- description: опис події
- date: дата події в ігровому світі

7. Note (Нотатка)

- id_note (PK): унікальний ідентифікатор нотатки
- id_campaign (FK): зовнішній ключ, що вказує на кампанію, до якої належить нотатка
- text: текст нотатки

8. NPC (Неігровий персонаж)

- id_npc (PK): унікальний ідентифікатор неігрового персонажа
- id_campaign (FK): зовнішній ключ, що вказує на кампанію, до якої належить NPC
- name: ім'я NPC
- description: опис NPC

9. Class (Клас персонажа)

- id_class (PK): унікальний ідентифікатор класу
- class_name: назва класу

10. Racial_Group (Расова група)

- id_racial_group (PK): унікальний ідентифікатор расової групи
- racial_group: назва расової групи

11. Attack (Атака)

- id_attack (PK): унікальний ідентифікатор атаки
- name: назва атаки
- attack_bonus: бонус до атаки
- damage_type: тип пошкоджень

12. Campaign_Characters (Зв'язок кампаній з персонажами)

- id_campaign_character (PK): унікальний ідентифікатор зв'язку
- id_campaign (FK): зовнішній ключ, що вказує на кампанію
- id_character (FK): зовнішній ключ, що вказує на персонажа

13. Character_Proficiency (Майстерність персонажа)

- id_character_proficiency (PK): унікальний ідентифікатор запису про майстерність
- id_character (FK): зовнішній ключ, що вказує на персонажа
- id_proficiency (FK): зовнішній ключ, що вказує на тип майстерності
- checker: прапорець наявності майстерності
- value: значення показника майстерності

14. Proficiency (Тип майстерності)

- id_proficiency (PK): унікальний ідентифікатор типу майстерності
- proficiency: назва типу майстерності

Зв'язки між сутностями:

1. **User** (Користувач) і **Campaign** (Кампанія): зв'язок "володіє" - один користувач може володіти багатьма кампаніями, але кожна кампанія належить лише одному користувачу.
2. **User** (Користувач) і **Session** (Сесія): зв'язок "проводить" - один користувач може проводити багато сесій, але кожна сесія проводиться одним користувачем.
3. **Campaign** (Кампанія) і **Location** (Локація): зв'язок "містить" - одна кампанія може містити багато локацій, але кожна локація належить лише одній кампанії.
4. **Campaign** (Кампанія) і **Event** (Подія): зв'язок "містить" - одна кампанія може містити багато подій, але кожна подія належить лише одній кампанії.

5. **Campaign** (Кампанія) і **Note** (Нотатка): зв'язок "містить" - одна кампанія може містити багато нотаток, але кожна нотатка належить лише одній кампанії.
6. **Campaign** (Кампанія) і **NPC** (Неігровий персонаж): зв'язок "містить" - одна кампанія може містити багато NPC, але кожен NPC належить лише одній кампанії.
7. **Campaign** (Кампанія) і **Character** (Персонаж): зв'язок "містить" через проміжну таблицю Campaign_Characters - одна кампанія може містити багато персонажів, і один персонаж може брати участь у багатьох кампаніях (зв'язок "багато-до-багатьох").
8. **Character** (Персонаж) і **Character_Proficiency** (Майстерність персонажа): зв'язок "має" - один персонаж може мати багато різних майстерностей.
9. **Character_Proficiency** (Майстерність персонажа) і **Proficiency** (Тип майстерності): зв'язок "відноситься до" - запис про майстерність персонажа відноситься до конкретного типу майстерності.
10. **Character** (Персонаж) і **Class** (Клас персонажа): зв'язок "належить до" - один персонаж належить до одного класу.
11. **Character** (Персонаж) і **Racial_Group** (Расова група): зв'язок "належить до" - один персонаж належить до однієї расової групи.

Така структура бази даних забезпечує ефективне зберігання та управління статичними даними системи, такими як інформація про користувачів, персонажів, кампанії та їх компоненти. Реляційна модель дозволяє встановлювати чіткі зв'язки між сутностями та забезпечує цілісність даних через систему зовнішніх ключів.

На рис. 11 представлена ER-діаграма, що відображає нереляційну структуру даних для збереження стану ігрових сесій. Ця частина бази даних реалізована з використанням NoSQL підходу (зберігання у форматі JSON), що забезпечує гнучкість при роботі з динамічними даними ігрового процесу.

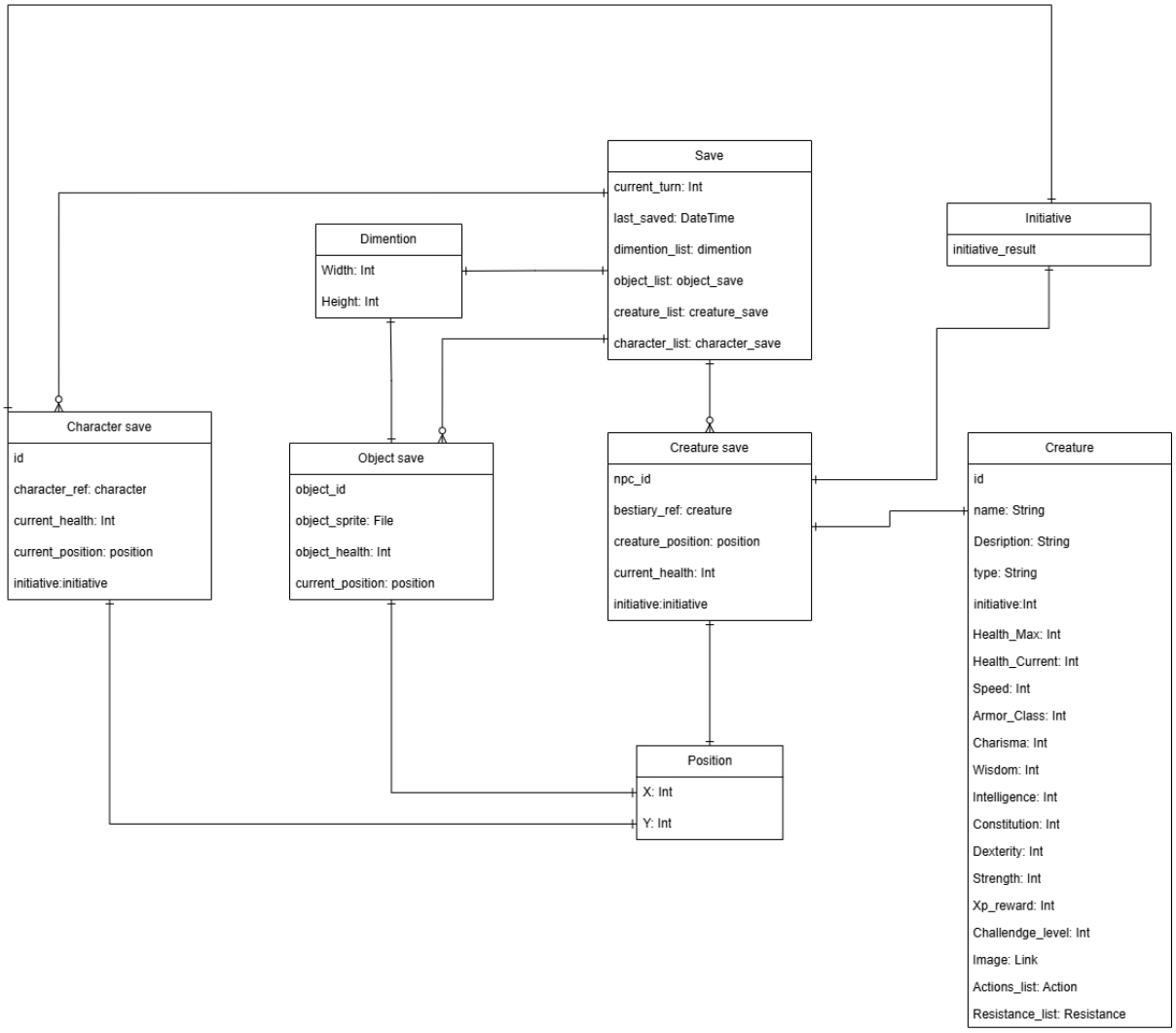

Рис. 11. ER-діаграма NoSQL структури для збереження ігрових сесій

Основні сутності та їх атрибути:

1**. Save** (Збереження)

- current_turn: поточний хід у грі
- last_saved: дата і час останнього збереження

- dimension_list: список вимірів у збереженні
- object_list: список об'єктів у збереженні
- creature_list: список істот у збереженні
- character_list: список персонажів у збереженні

2**. Dimension** (Вимір)

- Width: ширина ігрового поля
- Height: висота ігрового поля

3**. Position** (Позиція)

- X: координата X на ігровому полі
- Y: координата Y на ігровому полі

4. **Character_save** (Збереження персонажа)

- id: унікальний ідентифікатор збереження персонажа
- character_ref: посилання на персонажа в SQL базі даних
- current_health: поточне здоров'я персонажа
- current_position: поточна позиція на ігровому полі
- initiative: значення ініціативи персонажа

5**. Object_save** (Збереження об'єкта)

- object_id: унікальний ідентифікатор об'єкта
- object_sprite: файл зображення об'єкта
- object_health: здоров'я об'єкта (якщо застосовно)
- current_position: поточна позиція на ігровому полі

6. **Creature_save** (Збереження істоти)

- npc_id: ідентифікатор неігрового персонажа
- bestiary_ref: посилання на істоту в бестіарії
- creature_position: позиція істоти на ігровому полі
- current_health: поточне здоров'я істоти

- initiative: значення ініціативи істоти

7. **Creature** (Істота)
    - id: унікальний ідентифікатор істоти
    - name: назва істоти
    - Description: опис істоти
    - type: тип істоти
    - initiative: базове значення ініціативи
    - Health_Max: максимальне здоров'я
    - Health_Current: поточне здоров'я
    - Speed: швидкість переміщення
    - Armor_Class: клас броні
    - Charisma: показник харизми
    - Wisdom: показник мудрості
    - Intelligence: показник інтелекту
    - Constitution: показник статури
    - Dexterity: показник спритності
    - Strength: показник сили
    - Xp_reward: нагорода досвідом за перемогу
    - Challenge_level: рівень складності
    - Image: посилання на зображення
    - Actions_list: список доступних дій
    - Resistance_list: список стійкостей до типів пошкоджень
8. **Initiative** (Ініціатива)
    - initiative_result: результат кидка на ініціативу

Зв'язки між сутностями:

1. **Save** (Збереження) і **Dimension** (Вимір): зв'язок "містить" - одне збереження містить інформацію про один або кілька вимірів ігрового поля.
2. **Save** (Збереження) і **Object_save** (Збереження об'єкта): зв'язок "містить" - одне збереження містить інформацію про багато об'єктів на ігровому полі.
3. **Save** (Збереження) і **Creature_save** (Збереження істоти): зв'язок "містить" - одне збереження містить інформацію про багато істот на ігровому полі.
4. **Save** (Збереження) і **Character_save** (Збереження персонажа): зв'язок "містить" - одне збереження містить інформацію про багато персонажів на ігровому полі.
5. **Character_save** (Збереження персонажа) і **Position** (Позиція): зв'язок "має" - збереження персонажа включає інформацію про його позицію на ігровому полі.
6. **Object_save** (Збереження об'єкта) і **Position** (Позиція): зв'язок "має" - збереження об'єкта включає інформацію про його позицію на ігровому полі.
7. **Creature_save** (Збереження істоти) і **Position** (Позиція): зв'язок "має" - збереження істоти включає інформацію про її позицію на ігровому полі.
8. **Creature_save** (Збереження істоти) і **Creature** (Істота): зв'язок "посилається на" - збереження істоти містить посилання на конкретну істоту з бестіарію.
9. **Character_save** (Збереження персонажа) і **Initiative** (Ініціатива): зв'язок "має" - збереження персонажа включає значення його ініціативи.
10. **Creature_save** (Збереження істоти) і **Initiative** (Ініціатива): зв'язок "має" - збереження істоти включає значення її ініціативи.

Вибір комбінованого підходу з використанням SQL та NoSQL баз даних для веб-застосунку Dungeons & Dragons є стратегічним рішенням, що базується на комплексному аналізі вимог до системи та особливостей предметної області. Розглянемо детальніше технічні та архітектурні переваги такого рішення:

Відповідність типам даних

- Статичні дані (користувачі, персонажі, класи тощо) мають чітку структуру, тому найкраще зберігати їх у реляційній SQL базі з цілісністю зв'язків.
- Динамічні дані (стан сесії, характеристики персонажів) постійно змінюються, тож зберігання у NoSQL забезпечує гнучкість і простоту оновлення.

Продуктивність

- NoSQL забезпечує швидке читання/запис, що важливо для активної гри.
- Зберігання стану гри у JSON спрощує кешування і знижує навантаження на сервер.
- Стан можна отримати одним запитом, що зменшує кількість з'єднань.

Переваги розробки

- NoSQL легко адаптується до змін правил гри без складних міграцій.
- Розділення SQL і NoSQL дає змогу ізолювати модулі, спрощуючи розробку й тестування.
- Модульність сприяє незалежному масштабуванню компонентів.

Нефункціональні вимоги

- NoSQL добре масштабується горизонтально, підтримуючи зростання користувачів.
- Стан гри зберігається атомарно як один документ, що підвищує надійність.
- Розділення баз підвищує відмовостійкість системи.

Перспективи розвитку

- SQL полегшує інтеграцію з іншими системами (форуми, оплата). Структуровані дані SQL зручні для аналітики.
- Гібридний підхід дає гнучку основу для майбутніх розширень.

Поєднання SQL і NoSQL — не компроміс, а оптимальна архітектура для веб-застосунку настільних рольових ігор. Воно забезпечує баланс між структурованістю, гнучкістю, продуктивністю й надійністю, необхідними для успішної реалізації проєкту.

2.2. Діаграма класів та кооперацій

Діаграма класів (рис. 12) для веб-застосунку Dungeons & Dragons представляє структурну модель системи, що відображає основні класи, їх атрибути, методи та взаємозв'язки між ними. Ця діаграма є фундаментальною для розуміння архітектури програмного рішення та демонструє статичний вигляд системи.

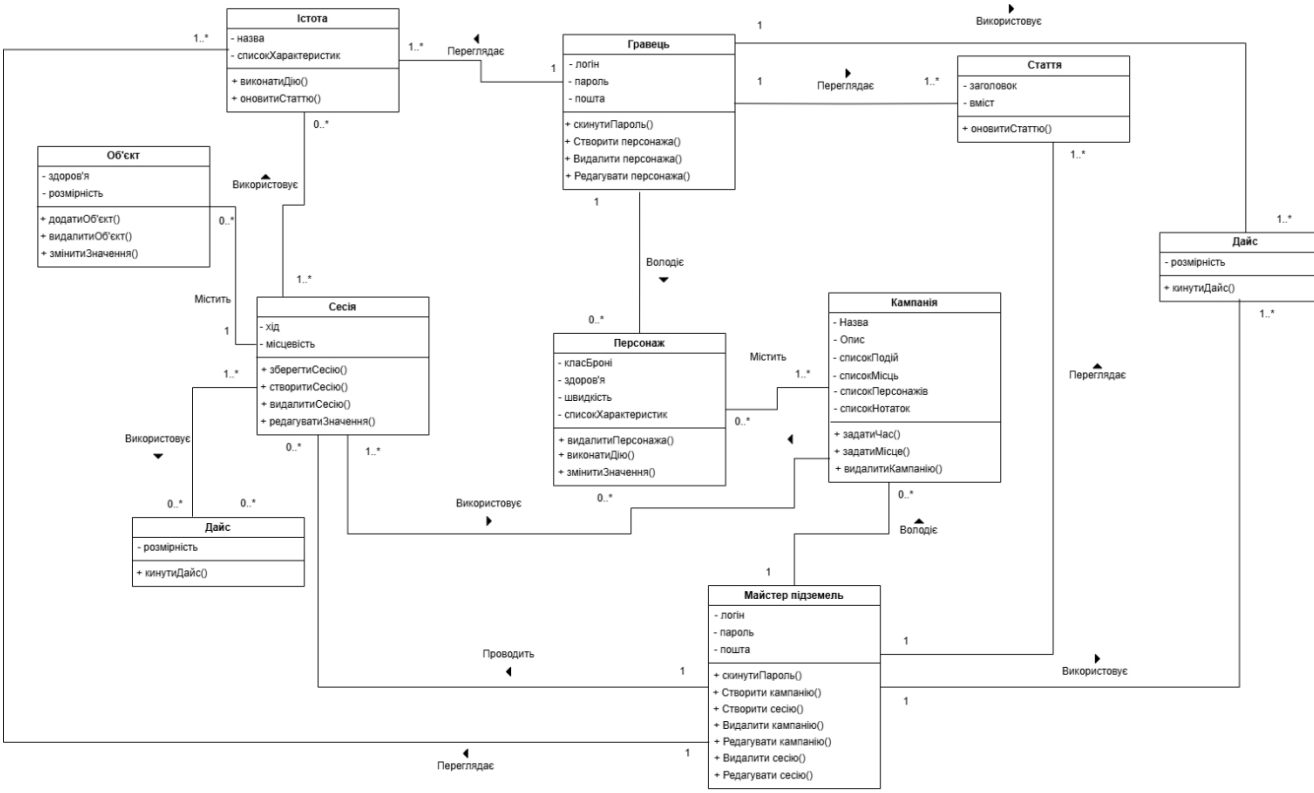

Рис. 12 Діаграма класів

Клас **"Гравець"** є ключовим для системи, оскільки представляє користувача з роллю гравця. Він містить атрибути для зберігання логіну, паролю, пошти та методи для сигнатури, створення, видалення та редагування персонажів. Цей клас має

зв'язок з класом "Майстер підземель", який є спеціалізованим типом користувача з розширеними можливостями для керування ігровим процесом.

Клас "**Майстер підземель**" успадковує атрибути та методи класу "Гравець", але додатково має функціональність для створення, видалення та редагування кампаній та сесій. Цей клас є центральним для організації ігрового процесу та має зв'язки з класами "Кампанія" та "Віртуальний стіл".

Клас "**Об'єкт**" представляє базовий елемент віртуального простору з атрибутами здоров'я, видимості та методами для додавання, видалення та зміни об'єктів. Він пов'язаний з класом "Віртуальний стіл", що забезпечує середовище для взаємодії об'єктів.

Клас **"Сесія**" відображає ігрову сесію з атрибутами ідентифікатора та місцевості, а також методами для керування характеристиками персонажів. Цей клас має зв'язок з класом "Персонаж", що представляє ігрового персонажа з атрибутами здоров'я, видимості та списком характеристик.

Клас "**Кампанія**" об'єднує кілька сесій у логічну послідовність і містить атрибути для зберігання часу, місцевості та списку персонажів. Цей клас має зв'язки з класами "Персонаж" та "Майстер підземель".

Клас **"Статті**" представляє інформаційні матеріали з атрибутами заголовка, змісту та опису статті. Цей клас пов'язаний з класом "Історія" та забезпечує доступ до довідкової інформації.

Клас **"Дайс"** представляє віртуальний кубик для генерації випадкових чисел з атрибутом видимості та методами для кидка кубика. Цей клас має зв'язки з класами "Віртуальний стіл" та "Гравець", що відображає використання кубиків у ігровому процесі.

Діаграми кооперацій (рис. 13) для веб-застосунку Dungeons & Dragons демонструють взаємодію об'єктів системи під час виконання конкретних сценаріїв

використання. Ця діаграма відображає динамічний аспект системи та показує, як об'єкти співпрацюють для досягнення певної мети.

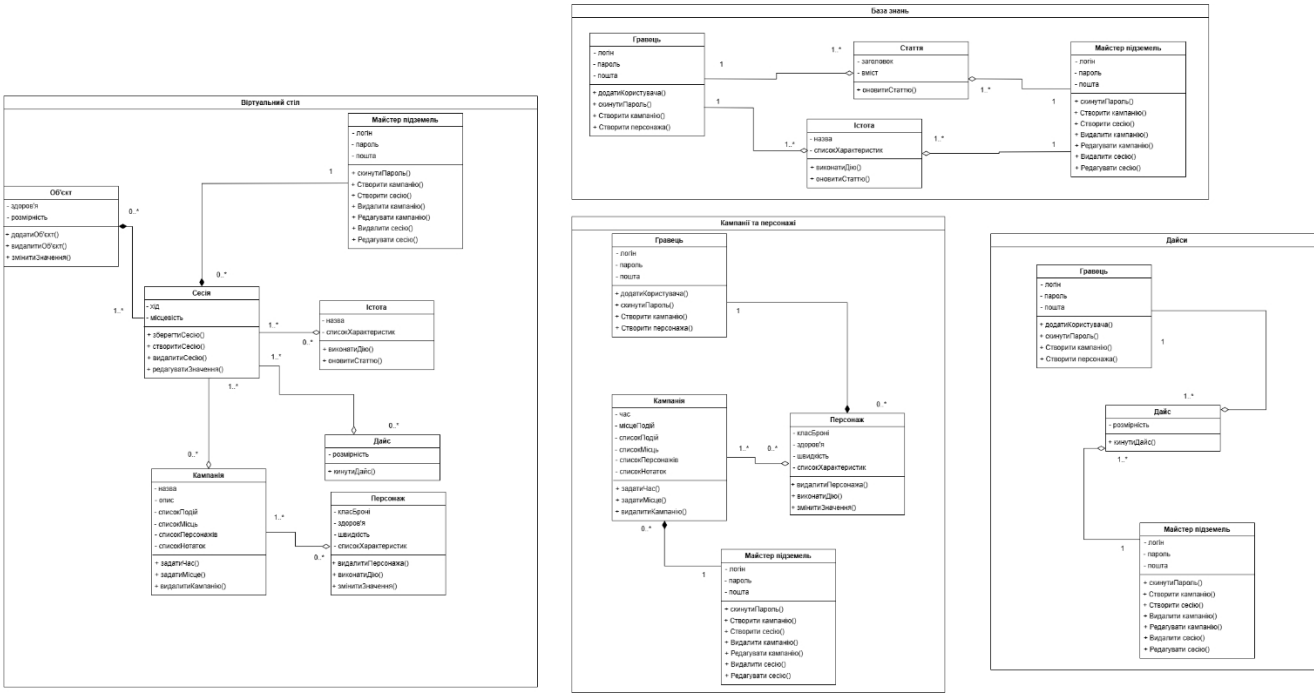

Рис. 13 Діаграми кооперації

2.3. Діаграма пакетів

Архітектура веб-застосунку *Dungeons & Dragons* побудована на основі п’яти основних пакетів (рис 14), кожен з яких відповідає за окремий аспект функціональності. Такий підхід забезпечує чітке розмежування відповідальності, модульність та ефективну взаємодію між компонентами системи.

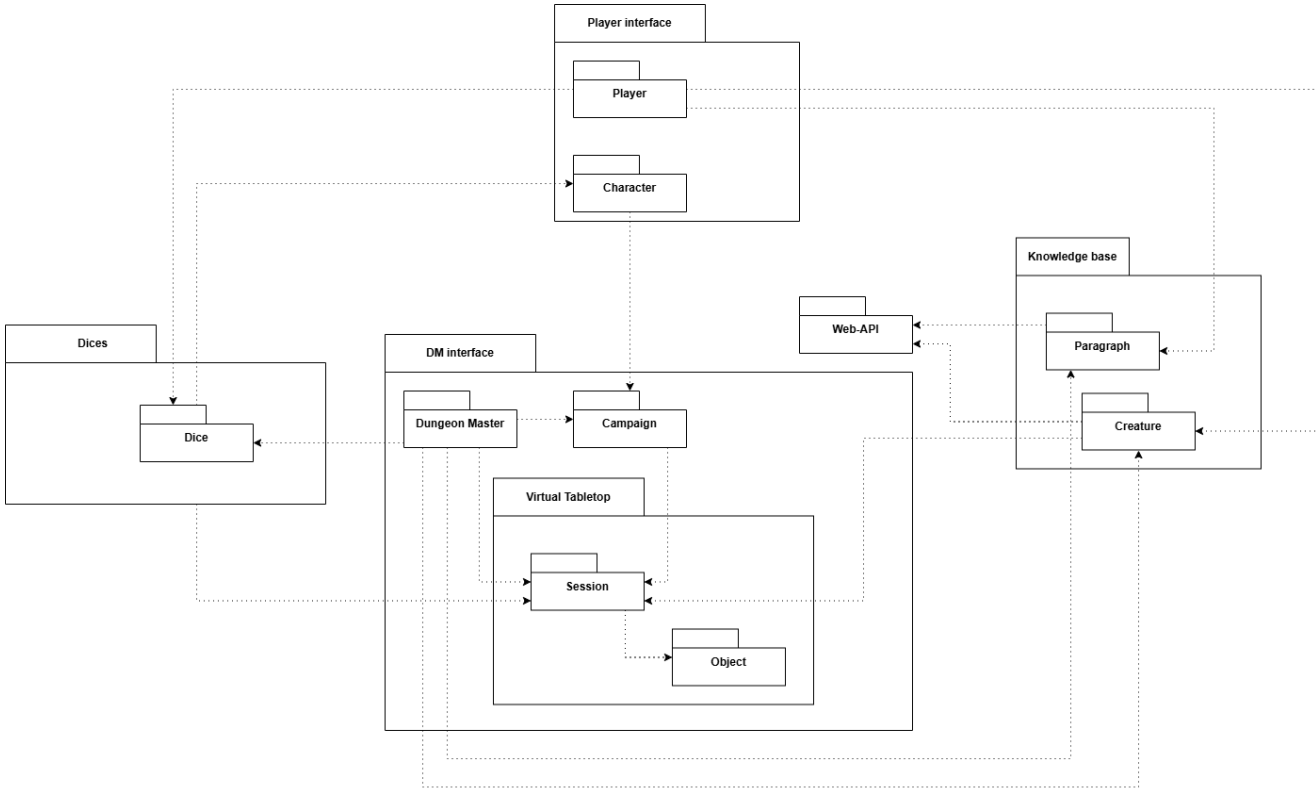

Рис. 14 Діаграма пакетів

Інтерфейс гравця охоплює компоненти для створення, редагування та видалення персонажів, забезпечуючи інтуїтивно зрозумілу взаємодію з ігровими сутностями та зручність керування ними.

Інтерфейс майстра підземель є розширенням функцій гравця і взаємодіє з модулем кампаній, що дозволяє створювати та адмініструвати ігрові сесії. Це дає змогу майстру ефективно керувати сценарієм та ігровим процесом.

Віртуальний стіл виступає ядром системи, створюючи інтерактивне середовище для проведення сесій. Він включає компоненти управління об'єктами та дозволяє візуалізувати перебіг гри.

База знань забезпечує централізоване зберігання довідкової інформації та надає доступ до неї через веб-API(Application Programming Interface). Вона містить відомості про правила гри, персонажів та істот.

Система кидків кубиків реалізована як незалежний модуль генерації випадкових чисел, що підтримує справедливість ігрових механік і використовується різними частинами застосунку.

Загалом обрана архітектура є оптимальною з погляду масштабованості, підтримки та розвитку. Кожен пакет можна вдосконалювати окремо, без потреби в суттєвих змінах решти системи, що сприяє зручності розробки та розширенню функціональності.

2.4. Діаграма компонентів

Діаграма компонентів (рис. 15) веб-застосунку *Dungeons & Dragons* відображає архітектуру системи, що складається з шести взаємопов'язаних підсистем, кожна з яких виконує специфічні функції та взаємодіє з іншими для забезпечення повноцінного функціонування.

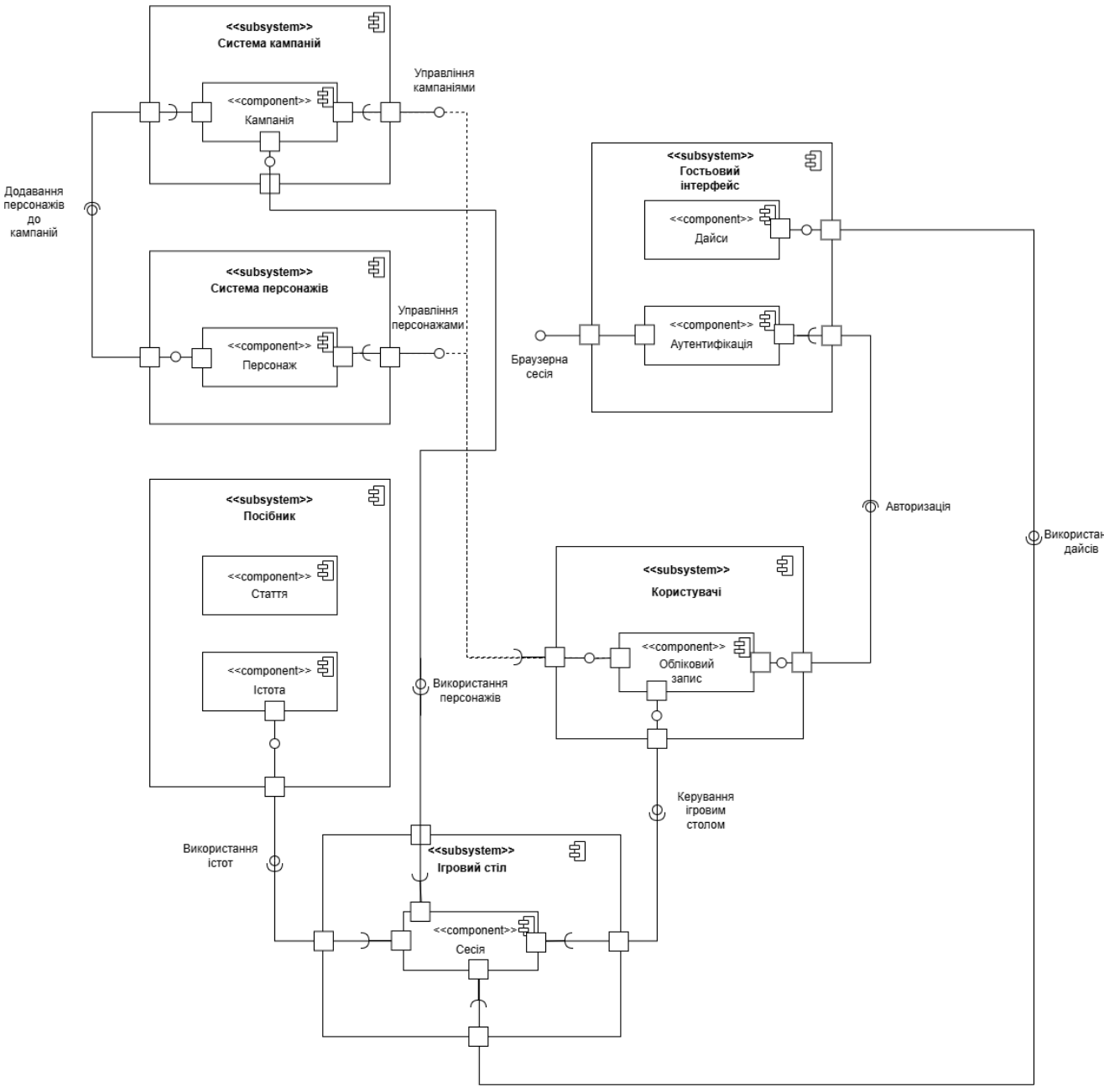

Рисунок 15 Діаграма компонентів

Система кампаній містить компонент "Кампанія" для створення, редагування та збереження ігрових кампаній. Вона інтегрується із **Системою персонажів**, що дозволяє пов’язувати персонажів із кампаніями, формуючи цілісний ігровий процес.

Система персонажів включає компонент "Персонаж" і відповідає за керування ігровими героями. Взаємодія з іншими підсистемами реалізується через інтерфейси, що забезпечує слабке зв’язування та високу масштабованість.

Підсистема "Посібник" складається з компонентів "Стаття" та "Історія", надаючи довідкову інформацію, правила гри та елементи наративу. Вона слугує джерелом знань і збагачує досвід гравців.

Підсистема "Ігровий стіл" містить компонент "Сесія", що забезпечує віртуальне середовище для проведення ігор. Вона тісно пов’язана з кампаніями, персонажами та обліковими записами, дозволяючи керувати токенами та ініціативою.

Підсистема "Користувачі" включає компонент "Обліковий запис" і відповідає за реєстрацію, автентифікацію, авторизацію та розмежування ролей користувачів (гравець, майстер підземель).

Гостьовий інтерфейс містить компоненти "Дайси" і "Автентифікація", що забезпечують доступ до базової функціональності (кидання кубиків, вхід у систему) для неавторизованих користувачів.

Взаємодія між підсистемами реалізована через чітко визначені інтерфейси, що підтримує модульність, масштабованість та спрощує інтеграцію нових функцій. Потоки даних між компонентами, зокрема "Управління кампаніями", "Керування персонажами", "Авторизація", "Використання дайсів", демонструють, як інформація циркулює системою.

3. РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління інформаційною базою

Для реалізації веб-застосунку Dungeons & Dragons необхідно обрати оптимальну систему управління інформаційною базою, яка забезпечить ефективне зберігання та обробку даних. Враховуючи специфіку проєкту, було прийнято рішення використовувати комбінований підхід, що поєднує реляційну базу даних та документо-орієнтовану NoSQL базу даних.

Аналіз діаграми класів веб-застосунку Dungeons & Dragons показує, що система містить як структуровані дані з чіткими зв'язками (користувачі, персонажі, кампанії), так і динамічні дані зі складною структурою (ігрові сесії, віртуальні столи). Тому було вирішено використовувати:

1. **Реляційну СУБД (MySQL)** для зберігання структурованих даних з чіткими зв'язками:

- Інформація про користувачів (гравці та майстри підземель)
- Дані персонажів
- Інформація про кампанії
- Довідкові статті

2. **Документо-орієнтовану СУБД (MongoDB)** для зберігання динамічних даних:

- Стан ігрових сесій
- Конфігурація віртуальних столів
- Історія ігрових подій

Такий комбінований підхід дозволяє використовувати переваги обох типів СУБД:

- Реляційна СУБД забезпечує цілісність даних, підтримку транзакцій та ефективне виконання складних запитів з об'єднанням таблиць
- NoSQL СУБД забезпечує гнучкість схеми даних, високу продуктивність при роботі з великими обсягами даних та зручне зберігання ієрархічних структур

На основі діаграми класів було розроблено структуру реляційної бази даних, яка включає основні таблиці, записані в додатку А.

Таблиця `user` зберігає основну інформацію про користувачів системи, включаючи їх облікові дані та зв'язок з персонажами. Модель успадковується від `UserMixin` для забезпечення функціональності автентифікації.

Таблиця `character` зберігає детальну інформацію про персонажів, включаючи їх базові характеристики, бойові показники та зв'язки з іншими таблицями (атаки, нотатки, клас, расова група та користувач-власник).

Таблиця `campaign` зберігає інформацію про кампанії, включаючи назву, опис та зв'язки з користувачем-власником та учасниками кампанії через таблицю `campaign_member`.

Для зберігання динамічних даних використовується json формат, що дозволяє зберігати дані у вигляді документів.

Колекція `session_states`

Приклад документу, що зберігає детальний стан ігрової сесії знаходиться в додатку Б.

Цей документ зберігає повний стан ігрової сесії, включаючи розташування об'єктів на віртуальному столі, результати кидків кубиків.

Для забезпечення ефективної взаємодії між реляційною та NoSQL базами даних використовується підхід, при якому:

1. Основні структуровані дані (користувачі, персонажі, кампанії) зберігаються в реляційній базі даних
2. Динамічні дані (стан ігрових сесій, детальна інформація про персонажів) зберігаються в NoSQL базі даних
3. Зв'язок між даними в різних базах забезпечується через спільні ідентифікатори (наприклад, `session_id`, `character_id`)

Такий підхід дозволяє:

- Забезпечити цілісність основних даних за допомогою реляційної СУБД
- Зберігати складні ієрархічні структури в NoSQL СУБД без необхідності їх нормалізації
- Ефективно виконувати запити як до структурованих, так і до динамічних даних

Використання комбінованого підходу дозволяє досягти оптимального балансу між структурованістю даних та гнучкістю їх зберігання, що є критично важливим для системи з такою складною предметною областю, як настільна рольова гра Dungeons & Dragons.

3.2 Розробка інформаційної бази

У цьому розділі описано процес розробки інформаційної бази даних для веб-застосунку Dungeons & Dragons, включаючи міграцію з SQLite до PostgreSQL та опис фізичної моделі бази даних.

Для розробки веб-застосунку початково використовувалася СУБД SQLite через її простоту та зручність. Для створення повноцінної діаграми фізичної моделі та детального аналізу структури бази даних було здійснено міграцію до PostgreSQL з використанням інструменту PGAdmin. Міграція бази даних включала наступні кроки:

1. Експорт схеми бази даних з SQLite

2. Модифікація SQL-скрипту для сумісності з PostgreSQL (заміна типів даних, адаптація синтаксису обмежень)
3. Створення бази даних в PostgreSQL
4. Імпорт схеми в PostgreSQL
5. Експорт даних з SQLite у форматі CSV
6. Імпорт даних в PostgreSQL
7. Перевірка цілісності даних після міграції

На основі мігрованої бази даних було створено діаграму фізичної моделі, яка відображає структуру таблиць, їх зв'язки, первинні та зовнішні ключі, а також типи даних.

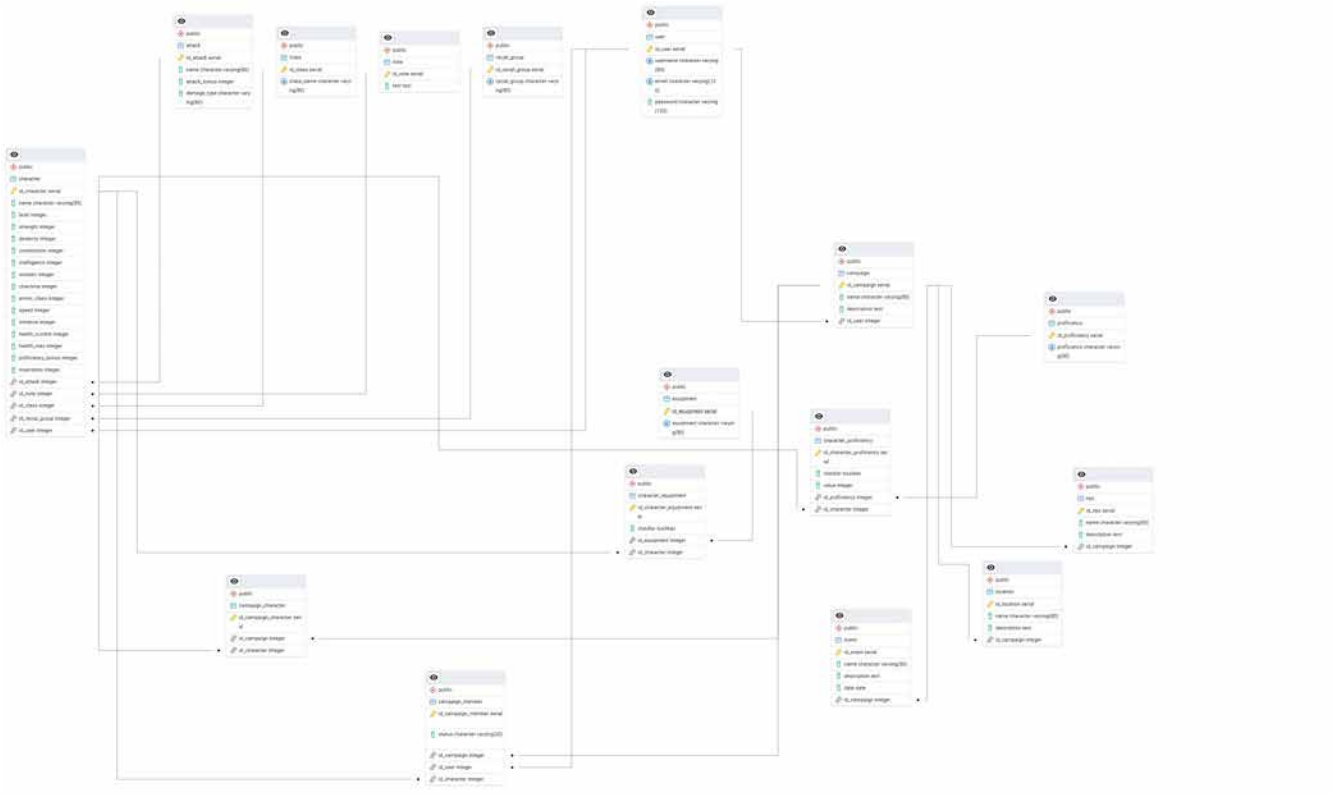

Рис. 16 Діаграма фізичної моделі даних реляційної БД

Фізична модель бази даних включає наступні основні таблиці та зв'язки між ними:

Таблиця `user` (Користувач)

Центральна таблиця системи, що зберігає інформацію про користувачів з унікальними ідентифікаторами, іменами, електронними поштами та паролями. Зв'язки: один користувач може мати багато персонажів, створювати багато кампаній та бути учасником багатьох кампаній.

Таблиця `character` (Персонаж)

Зберігає інформацію про ігрових персонажів з характеристиками, що визначають їх ігрові можливості. Зв'язки: кожен персонаж належить одному користувачу, має один клас, належить до однієї расової групи, може мати багато навичок, предметів спорядження та брати участь у багатьох кампаніях.

Таблиця `campaign` (Кампанія)

Містить інформацію про ігрові кампанії для проведення ігрових сесій. Зв'язки: кожна кампанія створюється одним користувачем, може мати багато учасників та включати багато персонажів.

Таблиці зв'язків між основними сутностями

- `campaign_member`: зв'язок між користувачами та кампаніями
- `campaign_character`: зв'язок між персонажами та кампаніями
- `character_proficiency`: зв'язок між персонажами та навичками
- `character_equipment`: зв'язок між персонажами та спорядженням

Довідникові таблиці

- `attack`: інформація про можливі атаки персонажів
- `note`: текстові нотатки, пов'язані з персонажами
- `class`: інформація про класи персонажів

- `racial_group`: інформація про расові групи персонажів
- `proficiency`: інформація про навички та вміння
- `equipment`: інформація про предмети спорядження
- `spell`: інформація про заклинання

Для зберігання динамічних даних ігрових сесій використовується JSON, що дозволяє гнучко зберігати дані, структура яких може змінюватися під час гри.

Комбінований підхід до зберігання даних

Веб-застосунок використовує комбінований підхід до зберігання даних:

- SQLite: для структурованих даних (користувачі, персонажі, кампанії)
- JSON: для динамічних даних ігрових сесій

Розроблена фізична модель бази даних повністю відповідає вимогам системи та забезпечує ефективне зберігання та обробку даних. Міграція з SQLite до PostgreSQL дозволила створити детальну діаграму фізичної моделі та провести більш детальний аналіз структури бази даних. Використання комбінованого підходу, що поєднує реляційну базу даних SQLite та документи JSON, дозволяє ефективно зберігати та обробляти різні типи даних веб-застосунку.

3.3 Вибір інструментарію для створення прикладного програмного забезпечення

Для серверної частини веб-застосунку Dungeons & Dragons обрано фреймворк Flask на Python через його легкість, гнучкість та швидкість розробки. Flask дозволяє повністю контролювати архітектуру, легко розширюється (через Flask-SQLAlchemy, Flask-Login), підтримує шаблонізатор Jinja2 та добре підходить для проєктів середнього масштабу.

Для зберігання структурованих даних використовується SQLite, що не потребує окремого сервера, є портативною, надійною (ACID (Atomicity, Consistency, Isolation, Durability) -транзакції) та має достатню продуктивність. Для аналізу структури БД використовувалась PostgreSQL, проте в робочій версії залишається SQLite. ORM(Object-Relational Mapping) SQLAlchemy (через Flask-SQLAlchemy) забезпечує абстракцію від SQL, підвищує безпеку та полегшує міграцію.

Динамічні дані ігрових сесій зберігаються у форматі JSON, що має гнучку структуру, компактність, нативну підтримку в Python і зручну інтеграцію з клієнтом.

Клієнтська частина реалізована стандартними технологіями:

- HTML5 – структура сторінок,
- CSS3 – адаптивність і стилі,
- JavaScript – інтерактивність.

Це забезпечує універсальність, продуктивність та просту підтримку.

Для генерації HTML використовується Jinja2, що підтримує умови, цикли, макроси, безпечне екранування HTML і легко інтегрується з Flask.

Flask-Login забезпечує автентифікацію користувачів, простий захист маршрутів і гнучке налаштування.

Werkzeug відповідає за хешування паролів та надає інструменти для WSGI-застосунків.

Markdown використовується для форматування тексту правил і описів. Вона має простий синтаксис, легко інтегрується та підтримує розширення. Візуалізація обраного інструментарію зображена на діаграмі на рис. 17.

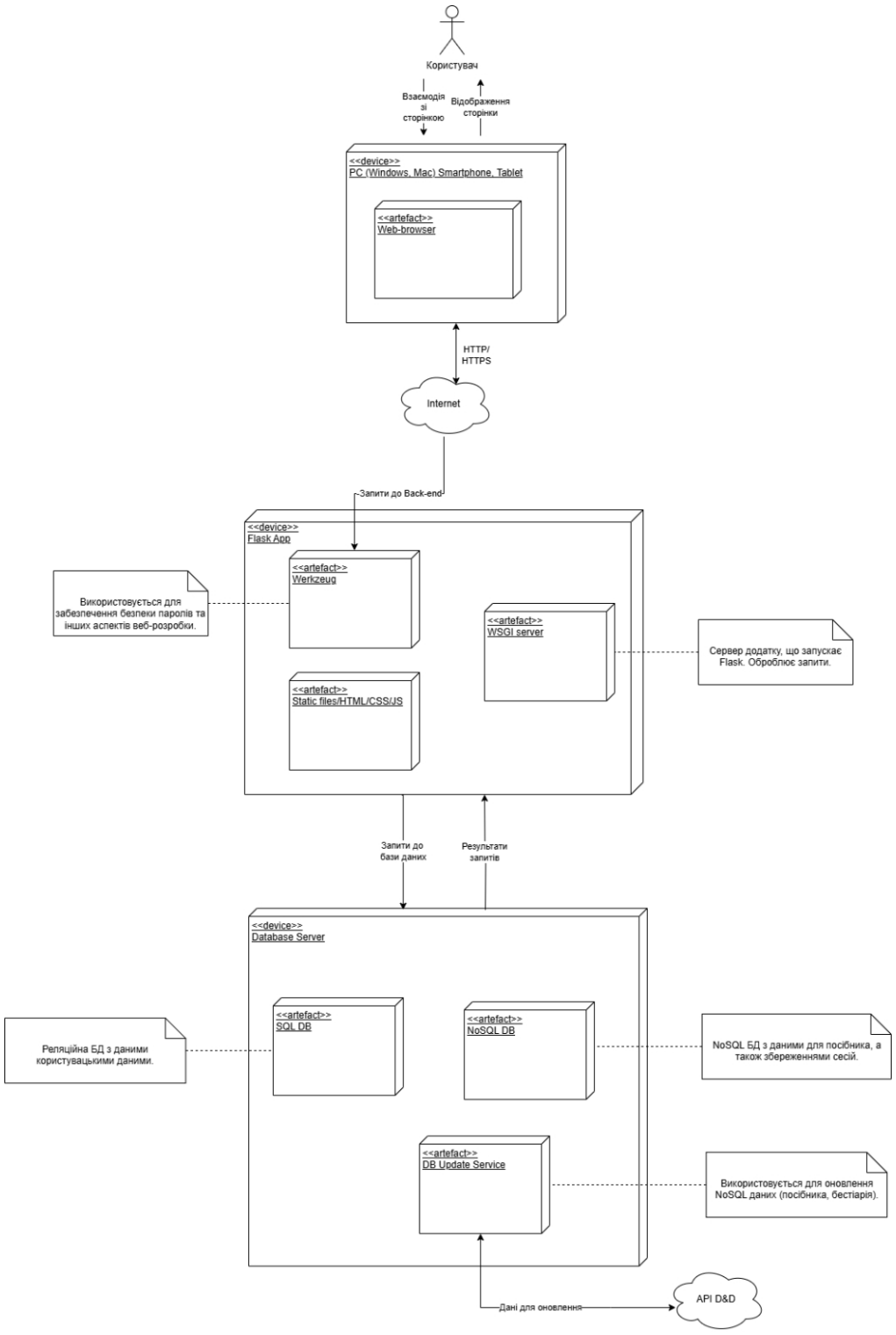

Рис. 17 Діаграма розрогтання

Обраний інструментарій для розробки веб-застосунку Dungeons & Dragons забезпечує оптимальний баланс між простотою розробки, продуктивністю та функціональністю. Використання Flask як основного фреймворку дозволяє швидко

розробляти та розгортати веб-застосунок, а SQLite забезпечує просте та надійне зберігання структурованих даних. Комбінація з JSON(JavaScript Object Notation) для динамічних даних надає необхідну гнучкість для складної предметної області настільної рольової гри. Стандартні веб-технології на клієнтській стороні забезпечують універсальність та продуктивність інтерфейсу, а додаткові інструменти, такі як Flask-Login та Werkzeug, вирішують специфічні завдання безпеки та автентифікації.

Таким чином, обраний інструментарій повністю відповідає вимогам проєкту та забезпечує ефективну **розробку та експлуатацію веб-застосунку Dungeons & Dragons.**

3.4 Алгоритмізація та програмування програмних модулів

Веб-застосунок побудовано за принципами модульної архітектури з використанням шаблону MVC (Model-View-Controller).

Основні компоненти

Моделі є структурою даних та бізнес-логікою. Представлення відповідає за відображення даних користувачу. Контролери забезпечують обробку запитів та координацію взаємодії.

Реєстрація включає валідацію даних, хешування пароля та збереження в БД. Автентифікація забезпечує перевірку облікових даних та створення сесії. Авторизація на рис. 18 відповідає за контроль доступу.

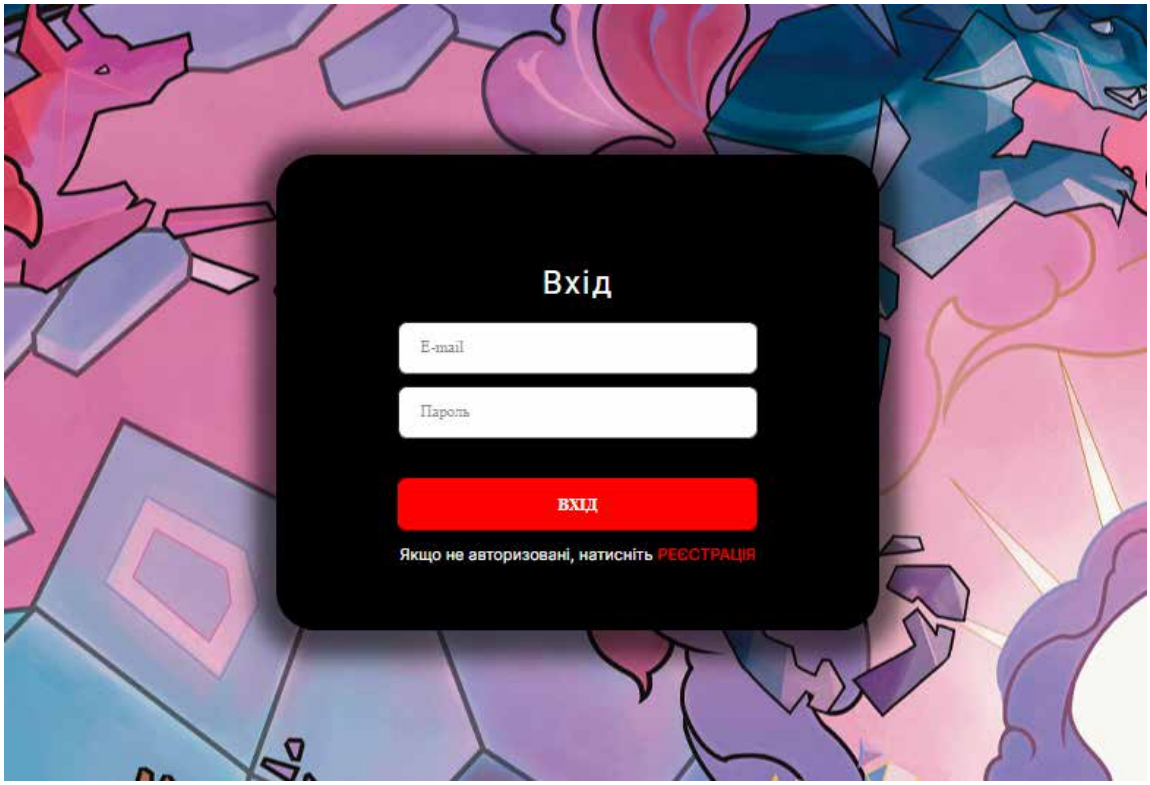

Рис. 18 Форма авторизації

Створення персонажа на рис. 19 передбачає вибір параметрів та розрахунок характеристик.

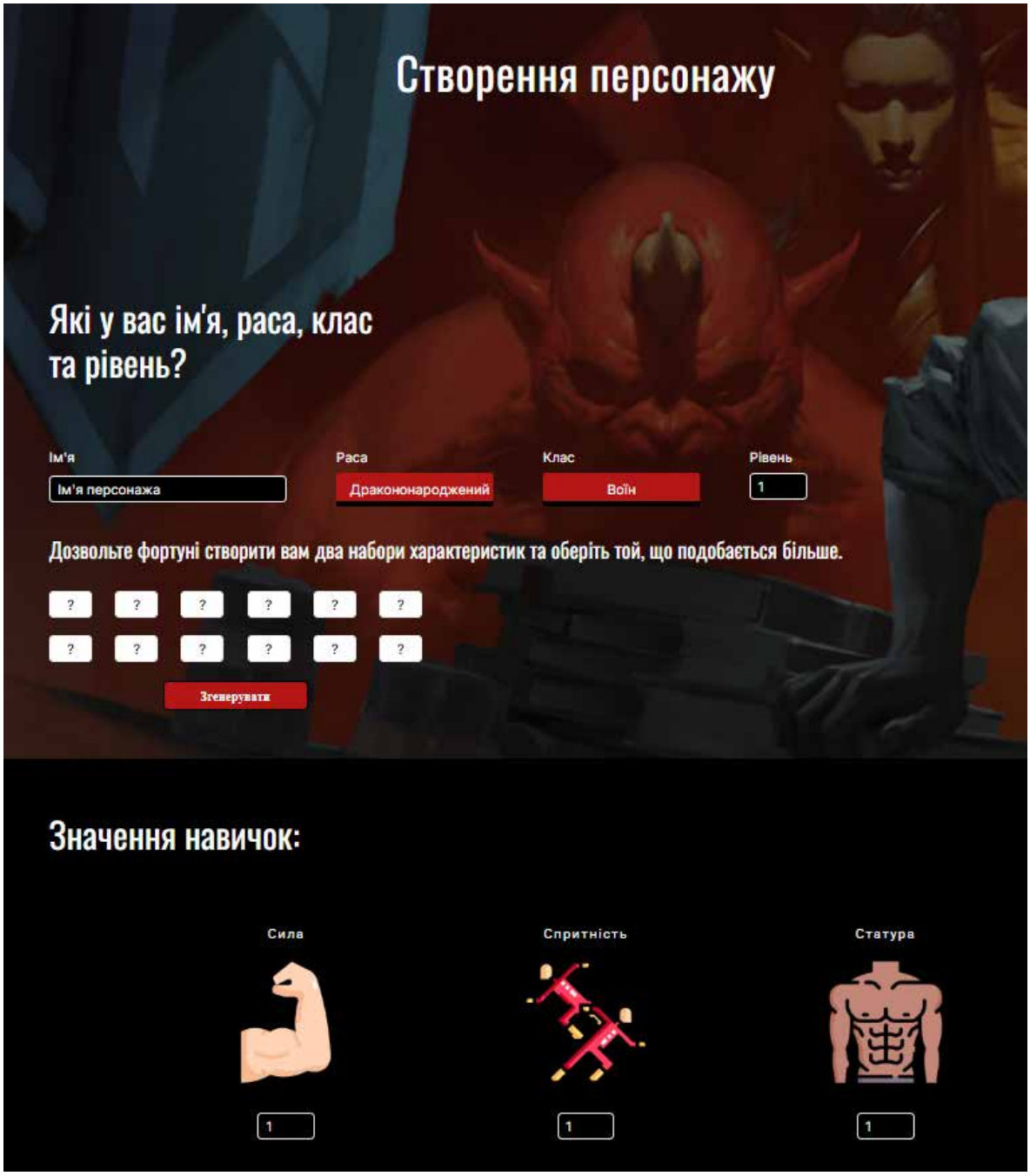

Рис. 19 Сторінка створення персонажу

Розрахунок модифікаторів атрибутів та бонусів на рис. 20 виконується на основі обраних параметрів. Управління інвентарем та спорядженням дозволяє відстежувати предмети персонажа.

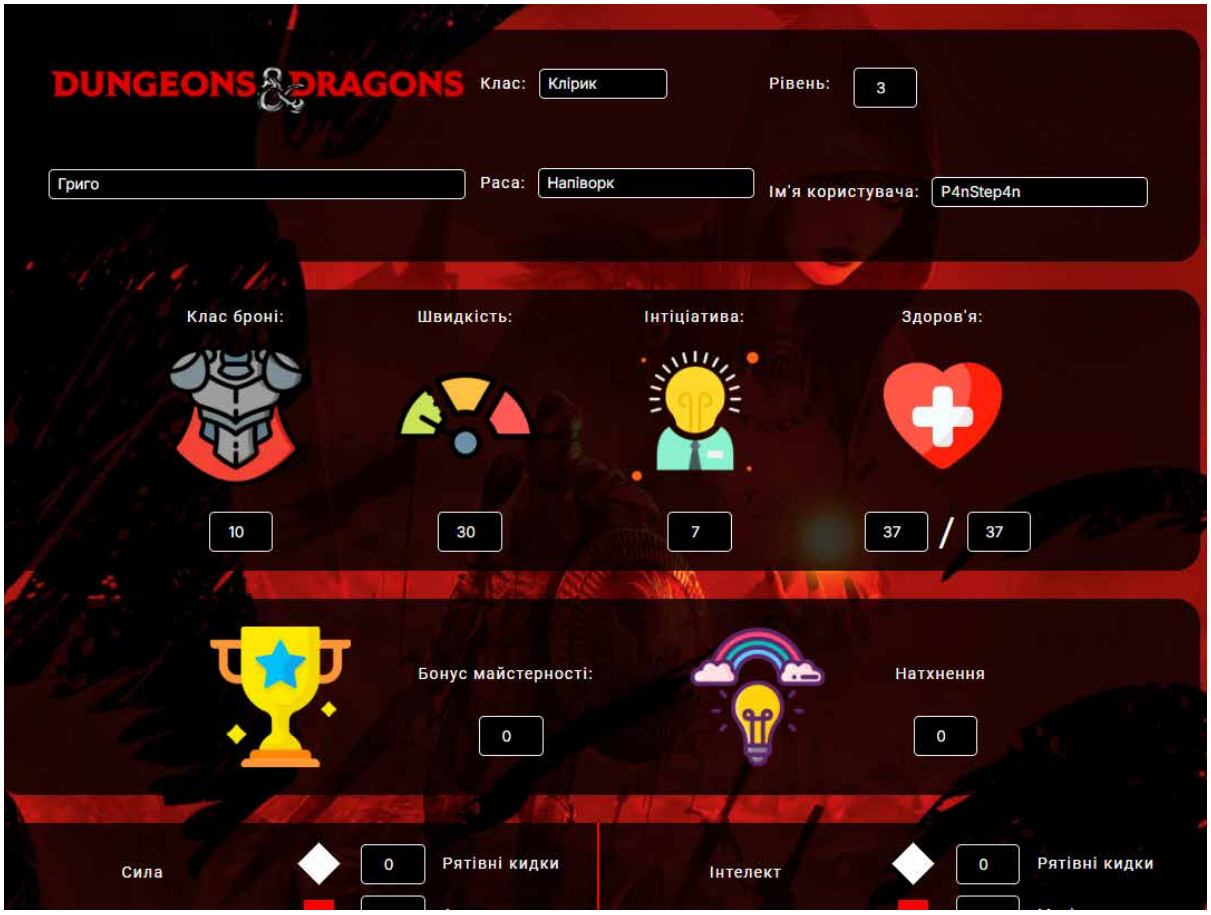

Рис. 20 Сторінка персонажу

Створення кампанії на рис. 21 включає налаштування та призначення ролей учасникам. Управління сесіями забезпечує відстеження прогресу гравців. Генерація зустрічей передбачає балансування складності відповідно до рівня персонажів.

Рис. 21 Сторінка кампанії

Відображення карти забезпечує рендеринг та масштабування ігрового поля на рис. 21. Управління фігурками дозволяє розраховувати переміщення на полі. Відстеження ініціативи та параметрів токенів допомогає планувати дії та відстежувати перебіг ігрової сесії.

Рис. 21 Сторінка сесії

Система кидків кубиків забезпечує генерацію випадкових чисел для визначення результатів дій. Система сесій реалізує механіки ініціативи, атаки та пошкодження. Система навичок дозволяє виконувати перевірки характеристик та навичок персонажів.

Взаємодія забезпечується через інтерфейси та події за принципами. Слабка зв'язаність між модулями дозволяє незалежно розробляти та тестувати компоненти. Високе згуртування всередині модулів забезпечує логічну цілісність функціональності. Подієва модель комунікації дозволяє ефективно обмінюватися даними між компонентами.

Розроблені модулі забезпечують ефективну реалізацію функціональних вимог веб-застосунку. Модульна архітектура та сучасні підходи до розробки дозволяють створити продуктивний та масштабований веб-застосунок для настільної рольової гри Dungeons & Dragons.

4. РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇСИСТЕМИ

4.1 Тестування системи

Тестування програмного забезпечення — це процес перевірки відповідності програмного продукту вимогам і виявлення дефектів. Метою тестування веб-застосунку *Dungeons & Dragons* є забезпечення його надійності, функціональності та зручності для користувачів. Основні завдання охоплюють перевірку всіх компонентів, своєчасне виявлення помилок, оцінку продуктивності під навантаженням, зручність інтерфейсу та дотримання вимог безпеки.

Модульне тестування зосереджене на окремих частинах системи: перевіряються моделі даних персонажів, логіка бойової системи, взаємодії елементів гри, а також контролери, які обробляють HTTP-запити, валідацію та помилки. **Інтеграційне тестування** перевіряє взаємодію між модулями — наприклад, автентифікація, керування персонажами, бойова механіка, ігровий стіл і кампанії — з акцентом на доступ, синхронізацію і правильність логіки.

Системне тестування включає перевірку функціональності, продуктивності, безпеки та зручності всієї системи в цілому. Тестуються ключові функції: створення персонажів, бойові сесії, ігровий стіл, обробка багатьох користувачів, захист даних і загальна UX-якість. **Приймальне тестування** відбувається на етапах альфа- та бета-тестування з залученням внутрішніх і зовнішніх користувачів для остаточного схвалення системи.

Ручне тестування дозволяє тестувальникам оцінити інтерфейс, зручність навігації, а також знайти неочевидні проблеми через дослідницьке тестування. **Автоматизоване тестування** охоплює регресійні перевірки, навантажувальні тести й API-тестування для забезпечення стабільності після змін і перевірки роботи інтерфейсів.

Тестові сценарії охоплювали автентифікацію, створення й редагування персонажів різних класів і рас, візуалізацію ігрового столу, розміщення фігурок та туман війни. У процесі тестування були виявлені й усунуті дефекти — зокрема, помилковий розрахунок модифікаторів та уповільнення через великі зображення. Ці проблеми було виправлено шляхом коригування формул, додавання валідації розміру файлів і оптимізації графіки.

Результати тестування підтвердили відповідність функціональним вимогам, усунення критичних помилок, стабільність роботи системи, продуктивність і комфортність інтерфейсу для користувачів.

Виявлені дефекти та їх виправлення

В процесі тестування було виявлено та успішно виправлено ряд дефектів різного рівня критичності. Серйозною проблемою був неправильний розрахунок модифікатора атрибуту, що впливало на характеристики персонажів та ігровий баланс. Для вирішення цієї проблеми було проведено детальний аналіз алгоритму розрахунку та внесено необхідні корективи у формулу, після чого проведено повторне тестування для підтвердження коректності розрахунків.

При роботі з великими зображеннями було виявлено проблему з їх завантаженням, що призводило до уповільнення роботи системи та можливих збоїв. Для вирішення цієї проблеми було впроваджено механізм валідації розміру файлів перед завантаженням та додано автоматичну оптимізацію зображень, що дозволило значно покращити продуктивність системи при роботі з графічним контентом.

Основні результати тестування демонструють значний прогрес у забезпеченні якості продукту. Було успішно підтверджено відповідність всім функціональним вимогам, що гарантує повноцінну реалізацію необхідного функціоналу.. Проведена оптимізація продуктивності забезпечила швидку та ефективну роботу всіх компонентів.

4.2 Вимоги до апаратного та програмного забезпечення

Веб-застосунок Dungeons & Dragons побудований за трирівневою архітектурою, що включає клієнтську частину, серверну частину та базу даних. Клієнтська частина реалізована як веб-інтерфейс, що забезпечує взаємодію користувача з системою через браузер. Серверна частина відповідає за обробку запитів, реалізацію бізнес-логіки та управління даними. База даних забезпечує надійне зберігання інформації про користувачів, персонажів, кампанії та ігрові ресурси.

Вимоги до апаратного забезпечення:

Клієнтська частина

- Мінімальні вимоги:
  - Процесор: двоядерний, 2.0 ГГц
  - Оперативна пам'ять: 4 ГБ
  - Вільний дисковий простір: 1 ГБ
  - Мережеве з'єднання: 5 Мбіт/с
- Рекомендовані вимоги:
  - Процесор: чотириядерний, 2.5 ГГц або потужніший
  - Оперативна пам'ять: 8 ГБ
  - Вільний дисковий простір: 2 ГБ
  - Мережеве з'єднання: 10 Мбіт/с

Серверна частина

- Мінімальні вимоги:
  - Процесор: чотириядерний, 2.5 ГГц
  - Оперативна пам'ять: 8 ГБ
  - Дисковий простір: 50 ГБ SSD
  - Мережеве з'єднання: 100 Мбіт/с

- Рекомендовані вимоги:
    - Процесор: восьмиядерний, 3.0 ГГц
    - Оперативна пам'ять: 16 ГБ
    - Дисковий простір: 100 ГБ SSD
    - Мережеве з'єднання: 1 Гбіт/с

Вимоги до програмного забезпечення:

Клієнтська частина

- Операційна система:
    - Windows 10/11
    - macOS 10.15 або новіше
    - Linux (Ubuntu 20.04 або новіше)
- Веб-браузер:
    - Google Chrome 90+
    - Mozilla Firefox 88+
    - Microsoft Edge 90+
    - Safari 14+
- Серверна частина
    - Операційна система:
    - Ubuntu Server 20.04 LTS або новіше
    - Windows Server 2019 або новіше
- Програмне забезпечення:
    - Python 3.9 або новіше
    - PostgreSQL 13.0 або новіше

4.3 Склад інсталяційного пакету

Інсталяційний пакет веб-застосунку Dungeons & Dragons містить всі необхідні компоненти для розгортання системи відповідно до трирівневої архітектури. Нижче наведено детальний опис складу пакету та призначення кожного компонента.

Клієнтська частина

- Файли веб-інтерфейсу:
  - HTML, CSS та JavaScript файли для користувацького інтерфейсу
  - Статичні ресурси (зображення, шрифти, іконки)
  - Файли локалізації
- Конфігураційні файли:
  - Налаштування підключення до серверної частини
  - Параметри кешування
  - Налаштування маршрутизації
  - Серверна частина
- Програмні модулі:
  - Файли Python-застосунку
  - Node.js серверні компоненти
  - Модулі бізнес-логіки
  - API-контролери
- Конфігураційні файли:
  - Параметри підключення до бази даних
  - Змінні середовища
  - Налаштування безпеки
- База даних
  - Файли бази даних SQLite:
  - Файл бази даних SQLite

    - Скрипти міграції схеми
    - Скрипти початкового наповнення даними
- Конфігураційні файли:
    - Налаштування з'єднання з SQLite
    - Параметри резервного копіювання бази даних
    - Додаткові компоненти
- Скрипти розгортання:
    - Автоматизовані скрипти встановлення
    - Скрипти налаштування середовища
    - Скрипти перевірки системних вимог
- Документація:
    - Інструкція з встановлення
    - Опис процесу розгортання
    - Рекомендації з налаштування
    - Інструкція з усунення типових проблем

Процес встановлення

1. Перевірка системних вимог
2. Встановлення необхідного програмного забезпечення
3. Розгортання серверної частини
4. Налаштування бази даних
5. Встановлення та налаштування веб-сервера
6. Розгортання клієнтської частини
7. Налаштування з'єднань між компонентами
8. Перевірка працездатності системи

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pallets Projects. Flask: The Python micro framework for building web applications [Електронний ресурс]. – Режим доступу: https://flask.palletsprojects.com/en/stable/ (дата звернення: 2.03.2025).¶
2. Grinberg M. Flask Web Development: Developing Web Applications with Python. – O'Reilly Media, 2018. – 316 с.¶
3. SQLite Documentation. SQLite: Self-contained, high-reliability, embedded, SQL database engine [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html (дата звернення: 11.04.2025).¶
4. Wizards of the Coast. Dungeons & Dragons Player's Handbook. – Wizards of the Coast, 2014. – 320 с.¶
5. Mozilla Developer Network. Web technology for developers [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web (дата звернення: 03.03.2025).¶
6. Owlbear Rodeo. A more bearable virtual tabletop [Електронний ресурс]. – Режим доступу: https://www.owlbear.rodeo/ (дата звернення: 11.03.2025).¶
7. Sly Flourish. Owlbear Rodeo: A Simple D&D Virtual Tabletop [Електронний ресурс]. – Режим доступу: https://slyflourish.com/owlbear_rodeo.html (дата звернення: 10.05.2025).¶
8. Czepeku. What's The Best VTT (Virtual Tabletop) For You? [Електронний ресурс]. – Режим доступу: https://www.czepeku.com/blog/choosing-your-ideal-virtual-tabletop (дата звернення: 15.03.2025).¶
9. W3Schools. AJAX Introduction [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/js/js_ajax_intro.asp (дата звернення: 15.03.2025).¶
10. Mozilla Developer Network. Ajax - MDN Web Docs Glossary [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Glossary/AJAX (дата звернення: 12.05.2025).¶

11. GeeksforGeeks. Use Case Diagram – Unified Modeling Language (UML) [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/use-case-diagram/ (дата звернення: 11.04.2025).¶
12. GeeksforGeeks. Sequence Diagrams – Unified Modeling Language (UML) [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/ (дата звернення: 11.05.2025).¶
13. Lucidchart. UML Use Case Diagram Tutorial [Електронний ресурс]. – Режим доступу: https://www.lucidchart.com/pages/uml-use-case-diagram (дата звернення: 13.03.2025).¶
14. GeeksforGeeks. How to Build a Web App using Flask and SQLite in Python [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/how-to-build-a-web-app-using-flask-and-sqlite-in-python/ (дата звернення: 5.05.2025).¶
15. Cisco Learning. Flask and SQLite: Consuming API Endpoints with Python Requests [Електронний ресурс]. – Режим доступу: https://ciscolearning.github.io/cisco-learning-codelabs/posts/flask-sqlite-with-requests/ (дата звернення: 13.05.2025).¶
16. Netguru. 24 Top Frontend Technologies to Use in 2025 [Електронний ресурс]. – Режим доступу: https://www.netguru.com/blog/front-end-technologies (дата звернення: 11.04.2025).¶
17. CACMS Institute. The Evolution of Front-End Development: From HTML/CSS to Modern JavaScript Frameworks [Електронний ресурс]. – Режим доступу: https://medium.com/@cacmscdm/the-evolution-of-front-end-development-from-html-css-to-modern-javascript-frameworks-e2d133d6d347 (дата звернення: 3.05.2025).¶

18. Özbek K. Building a Simple Library API with Flask and SQLite [Електронний ресурс]. – Режим доступу: https://medium.com/@kaanberkozbek/building-a-simple-library-api-with-flask-and-sqlite-9a5fbcdac324 (дата звернення: 2.03.2025).¶
19. DigitalOcean. How to Build a Flask Python Web Application from Scratch [Електронний ресурс]. – Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3 (дата звернення: 13.05.2025).¶
20. UML Diagrams. Examples of UML diagrams [Електронний ресурс]. – Режим доступу: https://www.uml-diagrams.org/index-examples.html (дата звернення: 11.03.2025).¶

ДОДАТКИ

ДОДАТОК А

Основні моделі реляційної БД

Модель `User` (користувачі)

```
class User(db.Model, UserMixin):
    __tablename__ = 'user'
    id_user = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(120), nullable=False)
    characters = db.relationship('Character', backref='user', lazy=True)

    def get_id(self):
        return str(self.id_user)
```

Модель `Character` (персонажі)

```
class Character(db.Model):
    __tablename__ = 'character'
    id_character = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), nullable=False)
    level = db.Column(db.Integer, nullable=False)
    strength = db.Column(db.Integer, nullable=False)
    dexterity = db.Column(db.Integer, nullable=False)
    constitution = db.Column(db.Integer, nullable=False)
    intelligence = db.Column(db.Integer, nullable=False)
    wisdom = db.Column(db.Integer, nullable=False)
```

```
    charisma = db.Column(db.Integer, nullable=False)
    armor_class = db.Column(db.Integer, nullable=False)
    speed = db.Column(db.Integer, nullable=False)
    initiative = db.Column(db.Integer, nullable=False)
    health_current = db.Column(db.Integer, nullable=False)
    health_max = db.Column(db.Integer, nullable=False)
    proficiency_bonus = db.Column(db.Integer, nullable=False)
    inspiration = db.Column(db.Integer, nullable=False)
    id_attack = db.Column(db.Integer, db.ForeignKey('attack.id_attack'),
nullable=False)
    id_note = db.Column(db.Integer, db.ForeignKey('note.id_note'),
nullable=False)
    id_class = db.Column(db.Integer, db.ForeignKey('class.id_class'),
nullable=False)
    id_racial_group = db.Column(db.Integer,
db.ForeignKey('racial_group.id_racial_group'), nullable=False)
    id_user = db.Column(db.Integer, db.ForeignKey('user.id_user'), nullable=False)
```

Модель `Campaign` (кампанії)

```
class Campaign(db.Model):
    __tablename__ = 'campaign'
    id_campaign = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), nullable=False)
    description = db.Column(db.Text, nullable=False)
    id_user = db.Column(db.Integer, db.ForeignKey('user.id_user'), nullable=False)
    owner = db.relationship('User', backref='campaigns', lazy=True)
    members = db.relationship('CampaignMember', backref='campaign', lazy=True)
```

ДОДАТОК Б

Приклад документа збереження ігрової сесії.


```
{
  "_id": "ObjectId('60a1e2b3c7f9d42e8c7890ab')",
  "session_name": "Бій у копальнях",
  "session_description": "Бій у копальнях відмінний від традиційних бойових сесій, тут кожен персонаж має право вибирати свою стратегію.",
  "session_id": 123,
  "timestamp": "2023-11-15T18:30:00Z",
  "virtual_table": {
    "grid_size": { "width": 20, "height": 15 },
    "objects": [
      {
        "id": "obj_1",
        "type": "token",
        "character_id": 456,
        "position": { "x": 5, "y": 7 },
        "health_current": 45,
        "health_max": 50,
        "armor_class": 12,
        "initiative": 15,
      },
      {
        "id": "obj_2",
        "type": "token",
        "position": { "x": 10, "y": 8 },
        "health_current": 15,
        "health_max": 20,
        "armor_class": 10,
        "initiative": 8,
      },
```

```
      {
        "id": "obj_3",
        "type": "object",
        "position": { "x": 15, "y": 10 },
      }
    ]
  },
}
```