ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка Tamagotchi game з використанням C# .NET та Angular(TypeScript)»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Євгеній САВЕЛЬЄВ
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Євгеній САВЕЛЬЄВ

Керівник: Олесь ДІБРІВНИЙ
доктор філософії (PhD)

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Савельєву Євгенію Максимовичу

1. Тема кваліфікаційної роботи: «Розробка Tamagotchi game з використанням C# .NET та Angular(TypeScript)»
керівник кваліфікаційної роботи доктор філософії (PhD) Олесь ДІБРІВНИЙ
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, методи та механіки ігрового дизайну.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Проаналізувати існуючі ігри жанру "Tamagotchi". Визначити їх переваги та недоліки.
2. Розробити концепцію гри, функціональні та нефункціональні вимоги.
3. Проаналізувати технічні засоби для розробки гри Tamagotchi.
4. Спроектувати модель архітектури та розробити гру.
5. Протестувати застосунок на відповідність вимогам.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до застосунку.
3. Концепт гри
4. Програмні засоби реалізації.
5. UseCase діаграма застосунку.
6. Діаграма PetController.
7. Database діаграма застосунку.
8. Екранні форми
9. Екранні форми
10. Відео роботи застосунку
11. Апробація результатів дослідження.

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Вивчення існуючих веб-ігор та механік, схожих на концепцію "Tamagotchi".	14.03-21.03.2024	
4	Проектування архітектури гри Tamagotchi.	22.03-09.04.2024	
5	Програмна реалізація основних функцій гри.	10.04-20.04.2024	
6	Проведення тестування гри та оцінка його продуктивності і зручності використання.	21.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Євгеній САВЕЛЬЄВ

(підпис)

Керівник
кваліфікаційної роботи _____ Олесь ДІБРІВНИЙ

(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 54 стор., 2 табл., 42 рис., 27 джерел.

Мета роботи – підтримка процесу піклування про віртуального персонажу засобами C# .NET та TypeScript Angular.

Об'єкт дослідження – процес піклування про віртуального персонажу.

Предмет дослідження – гра, що має функції піклування про віртуального персонажу.

Короткий зміст роботи: У цій роботі досліджено принципи та алгоритми, необхідні для розробки гри Tamagotchi з використанням C# та TypeScript. Описано методи створення інтерфейсу користувача, механіки взаємодії з віртуальним персонажем та інтеграцію з базою даних для зберігання даних про стан персонажа та прогрес користувача.

Використано технології C#, ASP.NET для бекенду, а також TypeScript, Framework Angular для побудови інтерфейсу користувача. Застосунок розроблено з урахуванням можливості подальшого розширення функціоналу та впровадження нових ігрових механік.

Ця гра призначена для використання як розважальний продукт, що дозволяє користувачам взаємодіяти з віртуальним персонажем, піклуючись про нього та виконуючи різні завдання для підтримання його активності та здоров'я.

КЛЮЧОВІ СЛОВА: ТАМАГОЧІ, C#, ANGULAR, ІГРОВИЙ ДИЗАЙН, ВІРТУАЛЬНИЙ ПЕРСОНАЖ, ІНТЕРАКТИВНИЙ ЗАСТОСУНОК.

ЗМІСТ

ВСТУП

Створення гри на основі інтерактивного контенту є ключовим напрямком розвитку сучасного програмного забезпечення. Особливої популярності набули веб-ігри та інші інтерактивні системи, що дозволяють користувачам взаємодіяти з віртуальними об'єктами в реальному часі. Ця тенденція зумовлена декількома факторами.

По-перше, зростання доступності Інтернету та мобільних пристроїв сприяє поширенню веб-застосунків, що надають користувачам різноманітні можливості для розваг, навчання та взаємодії з іншими. Інтерактивні веб-ігри, такі як гра Tamagotchi, дозволяють користувачам розвивати своїх віртуальних персонажів, виконувати завдання та створювати спільноти з іншими гравцями.

По-друге, сучасні технології веб-розробки, зокрема C# і TypeScript, дозволяють створювати складні та динамічні веб-застосунки, що забезпечують багатофункціональність та інтерактивність. Завдяки цим технологіям розробники мають можливість створювати багатокористувацькі середовища, обробляти велику кількість даних у реальному часі та забезпечувати плавний і інтуїтивний інтерфейс користувача.

По-третє, інтерес до віртуальних персонажів і симуляцій, який з'явився з популярністю класичних ігор на кшталт Tamagotchi, продовжує зростати. Сучасні ігри дозволяють відтворювати цю концепцію на новому технологічному рівні, використовуючи переваги сучасних веб-архітектур.

Зрештою, створення гри Tamagotchi не тільки задовольняє попит на інтерактивні веб-ігри, але й демонструє можливості використання сучасних технологій для створення інноваційного програмного забезпечення

1 ТЕОРЕТИЧНІ ОСНОВИ СТВОРЕННЯ ІГОР ДЛЯ ВІРТУАЛЬНОГО ДОГЛЯДУ ЗА ПЕРСОНАЖАМИ

1.1 Історія та розвиток концепції "Tamagotchi"

Концепція "Tamagotchi" походить з середини 1990-х років, коли вона була вперше представлена японською компанією Bandai. Оригінальний Tamagotchi був невеликим електронним пристроєм, який дозволяв користувачам піклуватися про віртуального вихованця. Гравці мали годувати його, розважати та доглядати, щоб утримувати його здоровим і щасливим.

Рис. 1.1 Оригінальна іграшка Tamagotchi від компанії Bandai

Ідея віртуального догляду за персонажами набула широкого поширення завдяки простоті гри та зручності використання пристрою. Tamagotchi швидко став глобальним феноменом, залучаючи мільйони користувачів у всьому світі. Віртуальні вихованці викликали почуття відповідальності та прив'язаності, оскільки гравці мали піклуватися про їх добробут.

Згодом концепція "Tamagotchi" розвивалася, з'являлися нові версії пристрою з розширеними можливостями та різними типами персонажів. Популярність Tamagotchi стала стимулом для появи багатьох інших ігор і застосунків, де використовувалася подібна механіка віртуального догляду. Сюди відносяться такі ігри, як "Digimon", де гравці виховують віртуальних монстрів і змагаються з іншими гравцями, та "Neopets", де користувачі піклуються про фантазійних вихованців у великому онлайн-світі.

Рис. 1.2 Обкладинка гри "Digimon World"

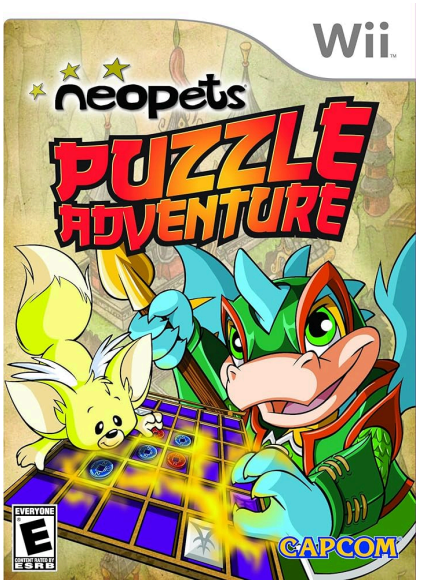


Рис. 1.3 Обкладинка гри "Neopets"

У сучасному світі концепція "Tamagotchi" знайшла нове життя завдяки веб-іграм та мобільним іграм. Багато з них використовують інтерактивність Інтернету, дозволяючи гравцям взаємодіяти з іншими користувачами, обмінюватися ресурсами та брати участь у спільних заходах. Сучасні версії Tamagotchi також включають додаткові функції, такі як міні-ігри, соціальні елементи та можливість персоналізації персонажів.

Отже, розвиток концепції "Tamagotchi" продовжує впливати на ігрову індустрію, особливо в галузі веб-ігор та мобільних застосунків. Проста ідея догляду за віртуальним персонажем стала основою для багатьох різноманітних ігор та надихнула нові способи взаємодії користувачів з віртуальним контентом.

1.2 Аналіз існуючих ігор з віртуальними персонажами

Ігри з віртуальними персонажами — це популярний жанр, який дозволяє користувачам взаємодіяти з віртуальними істотами в різних контекстах, включаючи догляд, виховання та соціальну взаємодію. У цьому підрозділі описано кілька основних категорій ігор з віртуальними персонажами та проаналізовані їхні ключові особливості.

1. Ігри з віртуальними вихованцями:

Цей тип ігор передбачає, що гравці доглядають за віртуальними вихованцями, годують їх, розважають і підтримують їхнє здоров'я та щастя. Вони можуть мати різні рівні складності та дозволяти користувачам розвивати свого вихованця з часом. Однією з найвідоміших ігор цієї категорії є "Neopets", де гравці піклуються про своїх віртуальних вихованців та взаємодіють з іншими гравцями у великому віртуальному світі.

Рис. 1.4 Ігрові персонажі гри "Neopets"

2. Ігри з соціальною взаємодією:

Цей тип ігор передбачає взаємодію між гравцями, які можуть спілкуватися, обмінюватися ресурсами та брати участь у спільних заходах. Наприклад, "Club Penguin" дозволяв користувачам створювати своїх віртуальних персонажів і взаємодіяти з іншими гравцями у віртуальному світі. У таких іграх часто є місії, завдання та події, які стимулюють соціальну взаємодію.

Рис. 1.4 Соціальна взаємодія в грі "Club Penguin"

3. Ігри з елементами симуляції:

Ці ігри дозволяють гравцям створювати та керувати віртуальними світами. Наприклад, "The Sims" дає можливість гравцям створювати персонажів, розвивати їхні відносини, будувати будинки та створювати цілі спільноти. Ці ігри часто орієнтовані на творчість і дозволяють гравцям розвивати свої персонажі у власному темпі.

Рис. 1.4 Екран гри "The Sims"

Ігри з віртуальними персонажами мають кілька загальних механік, які сприяють їхній популярності. По-перше, це взаємодія з персонажем, яка передбачає догляд, годування, розваги та інші дії, що стимулюють гравців до регулярного використання. По-друге, це елементи гейміфікації, такі як нагороди, бали, рівні та події, що підвищують інтерес гравців. По-третє, соціальна взаємодія, яка дозволяє користувачам спілкуватися, змагатися та обмінюватися ресурсами.

У результаті, аналіз існуючих ігор з віртуальними персонажами показує широкий спектр можливостей для розробників, а також підтверджує популярність цього жанру серед користувачів. Механіки та елементи гейміфікації сприяють залученню гравців і створюють досвід взаємодії з віртуальними персонажами.

1.3 Огляд методів гейміфікації у іграх

Гейміфікація — це використання ігрових елементів і механік у неігровому контексті, щоб підвищити залученість користувачів, мотивацію та задоволення від взаємодії з іграми.

Одним із найпоширеніших методів гейміфікації є система нагород і досягнень. Це може включати бали, значки, медалі або інші форми визнання, які користувачі отримують за виконання певних дій або досягнення цілей. Системи нагород стимулюють користувачів до активної взаємодії з застосунком та виконання різних завдань.

Інший метод гейміфікації передбачає використання системи рівнів, яка дозволяє користувачам підвищувати свій статус або розблоковувати нові можливості в застосунку. Рівні забезпечують відчуття прогресу та досягнення, що сприяє тривалому використанню гри.

Ігри часто включають завдання або квести, які користувачі можуть виконувати, щоб отримати нагороди або просунутися по рівнях. Завдання можуть бути щоденними, тижневими або спеціальними подіями. Вони структурують взаємодію користувачів із застосунком і надають їм конкретні цілі.

Лідерборди дозволяють користувачам порівнювати свої результати з іншими та змагатися за вищі позиції. Це створює елемент конкуренції, який мотивує користувачів брати активну участь у застосунку та досягати високих результатів.

Методи гейміфікації також можуть включати соціальні елементи, такі як взаємодія між користувачами, обмін ресурсами, створення спільнот або команди. Це дозволяє користувачам відчувати себе частиною спільноти, що підвищує їхню залученість і сприяє зростанню інтересу до гри.

Гейміфікація також може використовувати інтерактивні елементи та ігрову графіку, щоб зробити застосунок більш привабливим та захоплюючим. Це може включати анімацію, ігрові персонажі, візуальні ефекти та інші елементи, що додають ігровий контекст до гри.

Таким чином, огляд методів гейміфікації у іграх показує, що використання ігрових елементів може значно підвищити залученість користувачів і створити більш захоплюючий досвід. Різноманітність методів гейміфікації дозволяє іграм адаптуватися до потреб своєї аудиторії та стимулювати користувачів до тривалої взаємодії.

1.4 Технологічна база розробки веб-застосунків

Технологічна база розробки веб-застосунків охоплює широкий спектр інструментів, мов програмування та фреймворків, які дозволяють створювати динамічні, масштабовані та інтерактивні веб-застосунки.

Технологічна база розробки веб-застосунків постійно розвивається, включаючи нові інструменти та технології, які допомагають покращити продуктивність, забезпечити безпеку та підвищити якість користувацького досвіду. Наприклад, недавні тенденції включають використання серверного та клієнтського рендерингу, контейнеризацію застосунків, мікросервісну архітектуру та інші.

1.4.1 Мови програмування

Одним із ключових аспектів технологічної бази веб-застосунків є вибір мови програмування. Для розробки фронтенду зазвичай використовується JavaScript, тоді як для бекенду можуть використовуватися різні мови, зокрема C#, Python, Ruby, або Node.js.

- JavaScript: основна мова програмування для фронтенду. Використовується для створення динамічних та інтерактивних інтерфейсів користувача.

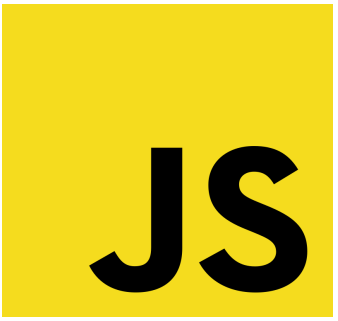


Рис. 1.5 логотип мови програмування "JavaScript"

- TypeScript: надбудова над JavaScript, що додає статичну типізацію та інші покращення, роблячи код більш стабільним і легким для підтримки.

Рис. 1.6 логотип мови програмування "TypeScript"

- C#: популярна мова програмування для бекенду, особливо у фреймворку ASP.NET. Відома своєю надійністю та високою продуктивністю.

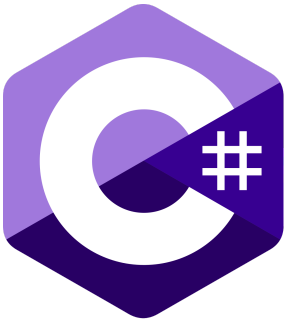


Рис. 1.7 логотип мови програмування "C#"

1.4.2 Фреймворки та бібліотеки

Фреймворки та бібліотеки надають структуровані підходи до розробки веб-застосунків, скорочуючи час розробки та забезпечуючи кращу організацію коду.

- Angular: фронтенд-фреймворк, який дозволяє створювати односторінкові

застосунки (SPA) з багатим користувацьким інтерфейсом. Framework Angular пропонує інструменти для двосторонньої прив'язки даних, модульності та тестування.

Рис. 1.8 логотип фреймворку "Angular"

- React: інший популярний фронтенд-фреймворк, відомий своєю гнучкістю та компонентним підходом.

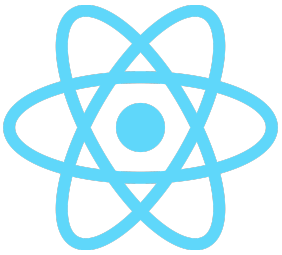


Рис. 1.9 логотип фреймворку "React"

- ASP.NET Core: фреймворк для розробки бекенду на базі C#. Підтримує створення веб-сервісів, REST API, та інтеграцію з базами даних.

Рис. 1.10 логотип фреймворку "ASP.NET Core"

1.4.3 Бази даних

Бази даних є невід'ємною частиною технологічної бази, забезпечуючи зберігання та управління даними.

- PostgresSQL: реляційна база даних, яка використовується для зберігання структурованих даних. Забезпечує високий рівень продуктивності та надійності.

Рис. 1.11 логотип фреймворку "PostgresSQL"

- MongoDB: нереляційна (NoSQL) база даних, яка дозволяє зберігати неструктуровані дані та забезпечує гнучкість при зміні схеми даних.

Рис. 1.12 логотип фреймворку "MongoDB"

1.4.4 Інструменти для контролю версій та розгортання

Для ефективної розробки та розгортання веб-застосунків використовуються інструменти контролю версій і системи автоматизації.

- Git: популярна система контролю версій, що дозволяє відстежувати зміни в коді, створювати гілки та співпрацювати з іншими розробниками.

Рис. 1.13 логотип системи контролю версій "Git"

- Docker: платформа для контейнеризації, яка дозволяє ізолювати застосунки та забезпечувати їхнє стабільне розгортання.

Рис. 1.14 логотип платформи для контейнеризації "Docker"

- CI/CD інструменти: системи для автоматизації інтеграції та розгортання, що допомагають швидко розгортати зміни та забезпечують безперервне тестування.

Рис. 1.15 логотип платформи для контейнеризації "Jenkins"

Технологічна база розробки веб-застосунків включає різноманітні інструменти та фреймворки, які дозволяють створювати сучасні, інтерактивні та масштабовані веб-застосунки. Вибір відповідних технологій залежить від вимог проєкту та переваг розробників.

1.5 Основи архітектури веб-застосунків

Архітектура веб-застосунків визначає, як організовані компоненти, як вони взаємодіють між собою та яким чином досягається функціональність і масштабованість. Розглянемо основні принципи архітектури веб-застосунків, а також ключові компоненти, що забезпечують надійність і продуктивність

сучасних веб-застосунків.

Більшість сучасних веб-застосунків використовують клієнт-серверну архітектуру, яка передбачає розділення функцій між фронтендом (клієнт) і бекендом (сервер). Клієнт відповідає за взаємодію з користувачем, тоді як сервер обробляє запити, виконує бізнес-логіку та взаємодіє з базою даних.

Односторінкові застосунки (SPA) — це популярний підхід, що дозволяє створювати динамічні та інтерактивні веб-застосунки. У SPA весь контент завантажується на початку, а взаємодія з користувачем відбувається без перезавантаження сторінки. Це забезпечує більш плавний і швидкий досвід для користувачів.

Фронтенд-архітектура визначає, як організовані компоненти, що відповідають за інтерфейс користувача. Бекенд-архітектура визначає, як сервер обробляє запити, виконує бізнес-логіку та взаємодіє з базою даних.

REST API є стандартним підходом для взаємодії між фронтендом і бекендом. Він дозволяє передавати дані у форматі JSON або XML, забезпечуючи гнучкість та ефективність комунікації між компонентами. REST API використовується для операцій CRUD (створення, читання, оновлення, видалення) і дозволяє фронтенду отримувати та відправляти дані до бекенду.

Бази даних — це ще один ключовий елемент архітектури веб-застосунків. Вони зберігають всю необхідну інформацію, таку як дані користувачів, ігрові дані та іншу контентну інформацію. Для взаємодії з базою даних бекенд використовує ORM (Object-Relational Mapping) інструменти або прямі запити SQL, залежно від вимог застосунку.

У результаті, основи архітектури веб-застосунків охоплюють різноманітні компоненти та принципи, що забезпечують надійність, продуктивність та масштабованість сучасних веб-застосунків. Вибір правильної архітектури має вирішальне значення для успіху застосунку та його довгострокової підтримки.

2 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ ІГОР ТА МЕХАНІК, СХОЖИХ НА КОНЦЕПЦІЮ "TAMAGOTCHI"

Ігри з віртуальними персонажами, які потребують постійного догляду та взаємодії, стали популярними завдяки їх простоті та інтерактивності. Однією з найбільш впізнаваних концепцій у цьому жанрі є Tamagotchi – віртуальний вихованець, якого потрібно годувати, розважати та доглядати. Сучасні ігри використовують цю концепцію та розширюють її різними механіками, інноваціями та додатковими функціями.

Одним із ключових аспектів таких ігор є використання графічних інтерфейсів та анімацій, щоб забезпечити привабливу візуалізацію для користувачів. Різні ігри застосовують різноманітні підходи до створення персонажів, включаючи 2D- та 3D-графіку, мультяшний чи реалістичний стиль.

Інший важливий аспект – механіка взаємодії з персонажами. Це може бути управління здоров'ям, настроєм, рівнем енергії та іншими параметрами. Гравцям надається можливість виконувати завдання, щоб підтримувати життя свого віртуального вихованця, а також заробляти бали чи нагороди, що стимулює подальшу гру.

Багато сучасних ігор також використовують соціальні елементи, дозволяючи гравцям взаємодіяти один з одним, обмінюватися ресурсами, створювати спільноти або змагатися в різних завданнях. Цей соціальний компонент сприяє залученню користувачів та утриманню їх у грі.

Деякі ігри пропонують можливість персоналізації персонажів, дозволяючи гравцям змінювати зовнішній вигляд, одяг та інші елементи. Це сприяє індивідуалізації досвіду користувача та стимулює бажання продовжувати гру.

Зрештою, розвиток ігор, схожих на Tamagotchi для гри в браузері, став можливим завдяки сучасним технологіям веб-розробки, таким як TypeScript та C#. Використання цих технологій дозволяє створювати складні, але інтуїтивно зрозумілі застосунки, які можуть масштабуватися та адаптуватися до різних

платформ, забезпечуючи при цьому плавний і привабливий досвід для користувачів.

2.1 Огляд популярних веб-ігор з інтерактивними персонажами

Веб-ігри з інтерактивними персонажами стали популярними завдяки своїй здатності залучати користувачів через взаємодію та персоналізацію. Серед таких ігор є кілька ключових категорій, які мають схожі риси з концепцією "Tamagotchi".

Перша категорія включає веб-ігри, де гравці піклуються про віртуального персонажа. Вони годують, одягають, розважають та лікують своїх віртуальних вихованців. Прикладом такої гри є "Neopets", де гравці можуть виховувати різноманітних фантазійних істот. Інші веб-ігри, як "Pet Society", пропонують схожі можливості, але з більш соціальним акцентом, дозволяючи гравцям взаємодіяти між собою.

Рис. 2.1 Ігровий екран гри "Neopets"

Рис. 2.2 Ігровий екран гри "Pet Society"

Друга категорія охоплює веб-ігри, в яких гравці керують своїми персонажами у більш складних середовищах. Вони виконують завдання, розвивають навички та беруть участь у різних подіях. Наприклад, "Club Penguin" пропонує гравцям створювати свого персонажа-пінгвіна та взаємодіяти з іншими гравцями в спільному світі.

Рис. 2.3 Ігровий екран гри "Club Penguin"

Третя категорія складається з веб-ігор, орієнтованих на будівництво та симуляцію. У них гравці створюють власні світи, де їхні персонажі живуть і взаємодіють з середовищем. Гравці можуть будувати будинки, створювати предмети та налаштовувати оточення. Прикладом такої гри є "The Sims", де гравці створюють цілі родини та управляють їхнім життям.

Рис. 2.4 Ігровий екран гри "The Sims"

Усі ці категорії мають загальні риси, такі як інтерактивність, персоналізація, соціальна взаємодія та можливість розвитку персонажів. Сучасні веб-ігри часто поєднують кілька цих категорій, створюючи багатогранні середовища, де гравці можуть досліджувати, взаємодіяти та розвиватися разом зі своїми персонажами.

Відповідно, огляд популярних ігор з інтерактивними персонажами показує широкий спектр підходів до реалізації концепції "Tamagotchi" та підкреслює зростаючий інтерес до ігор, що дозволяють гравцям взаємодіяти з віртуальними істотами та світом, в якому вони живуть.

2.2 Порівняння гри Tamagotchi з іграми аналогами

Гра Tamagotchi має свої унікальні особливості, але також може мати спільні риси з іншими віртуальними іграми, такими як NeoPets, Pet Society, Club Penguin і The Sims. Ось кілька порівняльних аспектів:

1. NeoPets і Pet Society часто спрямовані на молодших гравців, зокрема дітей, і надають можливість доглядати за віртуальними тваринами або створювати віртуальний світ. Club Penguin також спрямований на дітей, але пропонує більше можливостей для соціального взаємодії. У той час як The Sims більше спрямований на створення віртуального життя для персонажів у всіх вікових групах.
2. У грі Tamagotchi, гравець зазвичай доглядає за одним персонажем, подібно до оригінального. NeoPets, Pet Society та Club Penguin також надають можливості для догляду за віртуальними тваринами або персонажами, але можуть мати більше соціальних аспектів або ігор у великому масштабі. У The Sims гравці можуть керувати цілими сім'ями та будувати їхні домівки, працювати, розважатися та взаємодіяти з іншими персонажами у великому віртуальному світі.
3. Кожна гра має свій унікальний стиль графіки та інтерфейсу, що відображається в її цільовій аудиторії та механіках. Наприклад, NeoPets має яскравий та мультяшний вигляд, що приваблює дітей, тоді як The Sims має більш реалістичну графіку та деталізований інтерфейс для більшої глибини геймплею.
4. Club Penguin славиться своїми можливостями для соціальної взаємодії, де гравці можуть спілкуватися та грати разом у великому віртуальному світі. Також Pet Society має можливості для соціального взаємодії з іншими гравцями. У The Sims також є можливості для взаємодії з іншими гравцями, але в основному це відбувається в одиночному режимі гри.

Отже, хоча у всіх цих іграх є спільні риси, кожна з них має свої унікальні особливості та спрямована на різні аудиторії та геймплей.

Таблиця 2.1

Порівняння веб-ігор аналогів до гри "Tamagotchi"

Характеристика	Neopets	Pet Society	Club Penguin	**Tamagotchi game**
Платформа	Web	Web	Web	**Web**
Цільова аудиторія	Діти	Діти та підлітки	Діти та підлітки	**Діти та підлітки**
Графіка	2D	2D	2D	**2D**
Можливість створення аккаунту	+	+	+	+
Підтримка сучасних веб-браузерів (Chrome, Firefox, Safari, Edge)	+	+	-	+
Спілкування з персонажем через чат-бот штучного інтелекту	-	-	-	+

2.3 Аналіз ключових механік віртуального догляду за персонажами

Віртуальний догляд за персонажами є основною механікою у багатьох іграх, що ґрунтуються на концепції "Tamagotchi". Ця механіка включає декілька ключових аспектів, які забезпечують захопливість та цікавість гри для користувачів.

Один із базових елементів догляду — це потреба віртуального персонажа в їжі та воді. Гравці мають постійно стежити за рівнем голоду та спраги персонажа, надаючи йому необхідні ресурси. Для цього можуть бути використані різні типи продуктів, кожен із яких має свій ефект на стан персонажа.

Іншим важливим аспектом догляду є підтримання здоров'я персонажа. Це

може включати лікування хвороб, прийом вітамінів, а також забезпечення гігієнічних умов. Гравці можуть взаємодіяти з персонажем через медичні процедури або предмети, що покращують здоров'я.

Віртуальні персонажі потребують розваг та соціальної взаємодії. Цей елемент дозволяє гравцям виконувати різні активності, грати з персонажем у ігри або взаємодіяти з іншими персонажами. Це допомагає підтримувати хороший настрій і мотивацію персонажа.

Багато ігор пропонують механіки розвитку персонажів через навчання та отримання нових навичок. Гравці можуть брати участь у різних завданнях, щоб підвищити інтелектуальний рівень персонажа або навчити його новим здібностям.

Для ефективного догляду гравцям необхідно планувати свою взаємодію з персонажем. У деяких іграх використовується реальний час, де персонаж вимагає уваги протягом дня, що стимулює регулярний геймплей і постійну взаємодію.

2.4 Дослідження принципів гейміфікації у веб-застосунках

Гейміфікація у веб-застосунках є популярним підходом, що використовує елементи гри та ігрові механіки для залучення користувачів, підвищення їхньої мотивації та зацікавленості. Дослідження принципів гейміфікації дозволяє зрозуміти, як такі елементи можуть бути інтегровані у веб-застосунки для досягнення цілей, пов'язаних з підвищенням користувацької активності та лояльності.

Одним із основних принципів гейміфікації є надання користувачам нагород за виконання певних дій чи досягнення цілей. Це можуть бути бали, значки, рівні або інші форми визнання. Користувачі можуть накопичувати ці нагороди та отримувати доступ до нових функцій або контенту.

Ще одним важливим елементом гейміфікації є система рівнів, яка дозволяє користувачам відстежувати свій прогрес. Підвищення рівня може відкривати нові можливості чи контент, що стимулює користувачів продовжувати взаємодію з веб-застосунком.

Веб-застосунки з елементами гейміфікації часто пропонують користувачам виконувати завдання або місії. Це дозволяє структурувати взаємодію з застосунком, а також додає елемент цілеспрямованості. Завдання можуть бути щоденними, тижневими чи іншими, стимулюючи регулярне використання застосунку.

Створення елементів конкуренції між користувачами, таких як лідерборди, також є ефективним способом гейміфікації. Це дозволяє користувачам порівнювати свої результати з іншими та змагатися за вищі позиції, що сприяє підвищенню залученості та мотивації.

Інтеграція можливості персоналізації аватарів або профілів користувачів також є важливим елементом гейміфікації. Це дозволяє користувачам відчувати себе частиною спільноти, а також виражати свою індивідуальність.

Дослідження цих принципів гейміфікації у веб-застосунках показує, що використання ігрових елементів може значно підвищити взаємодію та зацікавленість користувачів, а також стимулювати їх до тривалого використання застосунків. Це робить гейміфікацію ефективним інструментом для створення захопливих та привабливих веб-застосунків.

3 РОЗРОБКА ГРИ TAMAGOTCHI

Розробка гри Tamagotchi включає використання різних інструментів та програм, що забезпечують ефективну роботу з фронтендом, бекендом, базою даних та контролем версійності. У цьому розділі ми розглянемо обрані програми, які використовуються для розробки та тестування гри.

Для розробки бекенду застосовується фреймворк ASP.NET на мові C#. Основною середою розробки є JetBrains Rider, яка забезпечує потужний набір інструментів для програмування, зокрема автозаповнення коду, рефакторинг та інтеграцію з системами контролю версій. Rider також має підтримку різних фреймворків і бібліотек, що робить його гнучким вибором для розробки бекенду.

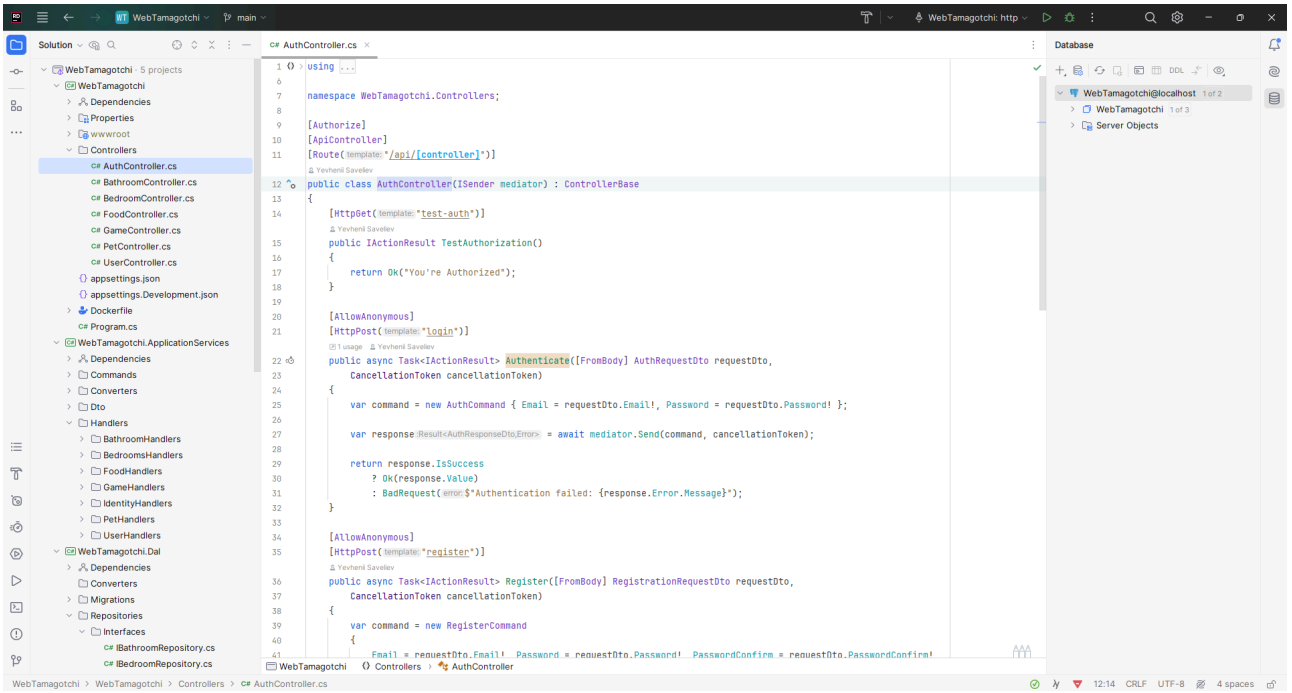


Рис. 3.1 Середа розробки JetBrains Rider

Для тестування бекенду та перевірки REST API використовується Postman. Цей інструмент дозволяє відправляти HTTP-запити до бекенду та перевіряти відповіді, що допомагає у відлагодженні та тестуванні різних сценаріїв. Postman також підтримує створення колекцій запитів, які можна використовувати для автоматизації тестування.

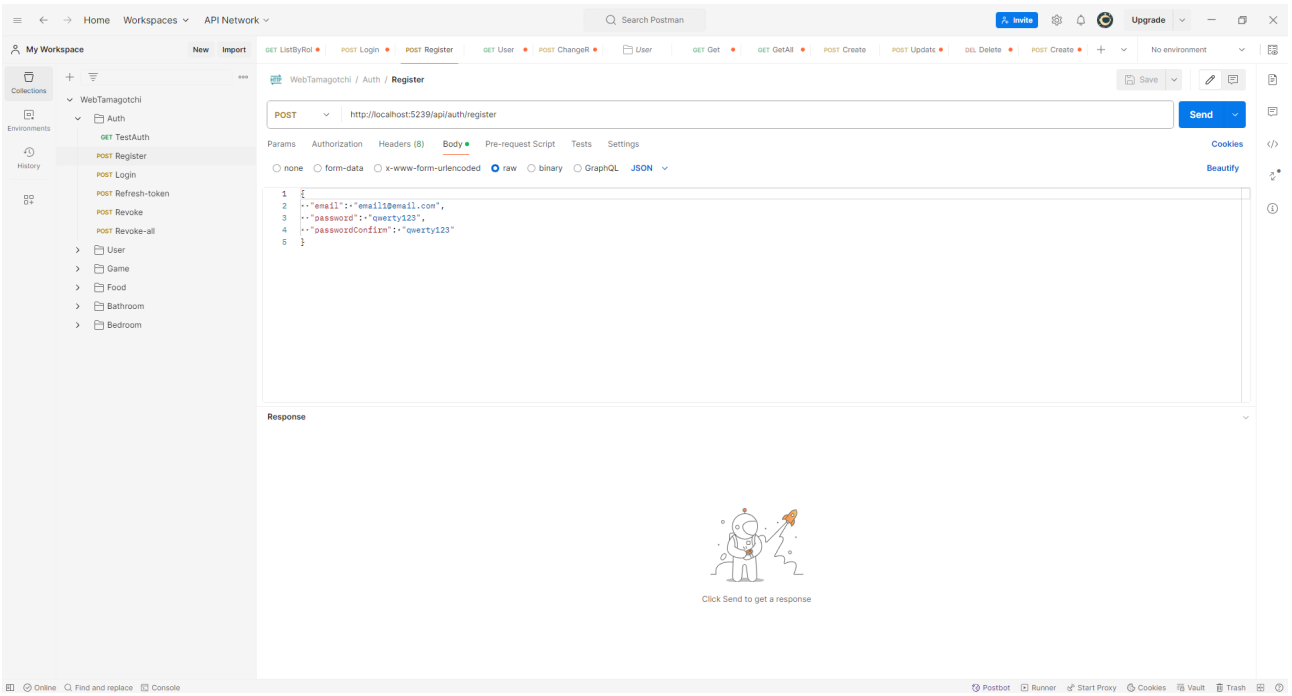


Рис. 3.2 Інструмент Postman

Для розробки чату в грі використовувася OpenAI API - це потужний інструмент штучного інтелекту, який дозволяє використовувати передові моделі генерації тексту. Цей чат-бот базується на моделі Generative Pre-trained Transformer (GPT), що надає можливість створювати природні та змістовні відповіді на запитання гравців та взаємодіяти з ними у реальному часі.

За допомогою налаштувань запитів, чат-бот повністю імітує поведінку віртуального персонажа, що дозволяє відтворити його реальні емоції. Також чат-бот з віртуальним персонажем можна використоувати як ассистента зі штучним інтелеком, для вирішення тривіальних задач або порад.

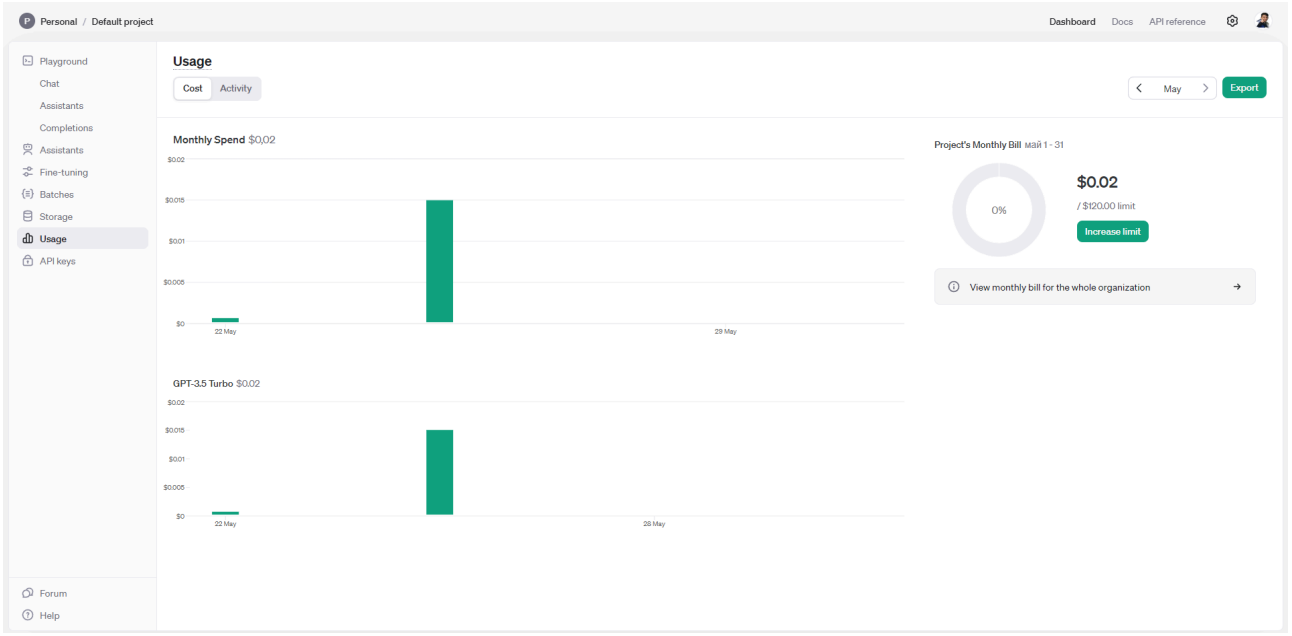


Рис. 3.3 OpenAI API Dashboard

Фронтенд застосунку розробляється з використанням фреймворку Angular. Для цього використовується Visual Studio Code (VSCode) — популярна середа розробки з підтримкою різних плагінів та розширень. VSCode дозволяє розробникам ефективно працювати з Angular, забезпечуючи автозаповнення коду, інтеграцію з системами контролю версій та можливості відлагодження.

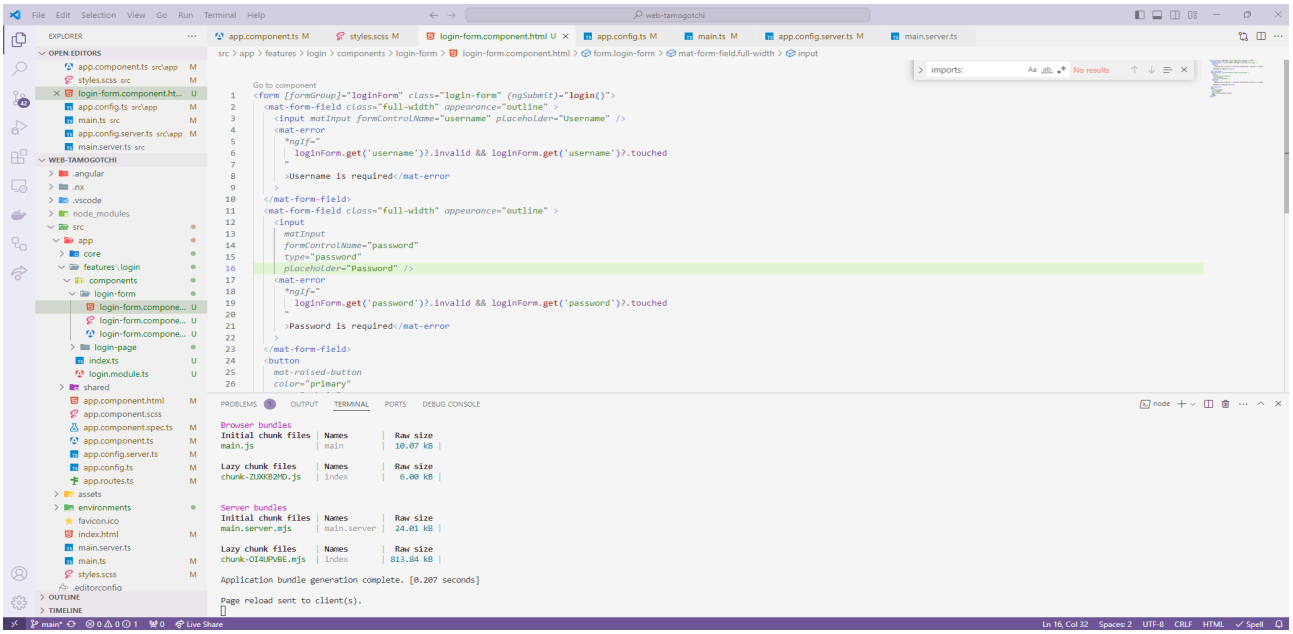


Рис. 3.4 Середа розробки Visual Studio Code

Для тестування фронтенду використовується BrowserStack. Цей інструмент дозволяє тестувати гру на різних браузерах і пристроях, забезпечуючи сумісність та виявлення проблем, пов'язаних із різними конфігураціями. BrowserStack допомагає переконатися, що фронтенд працює стабільно на всіх основних платформах.

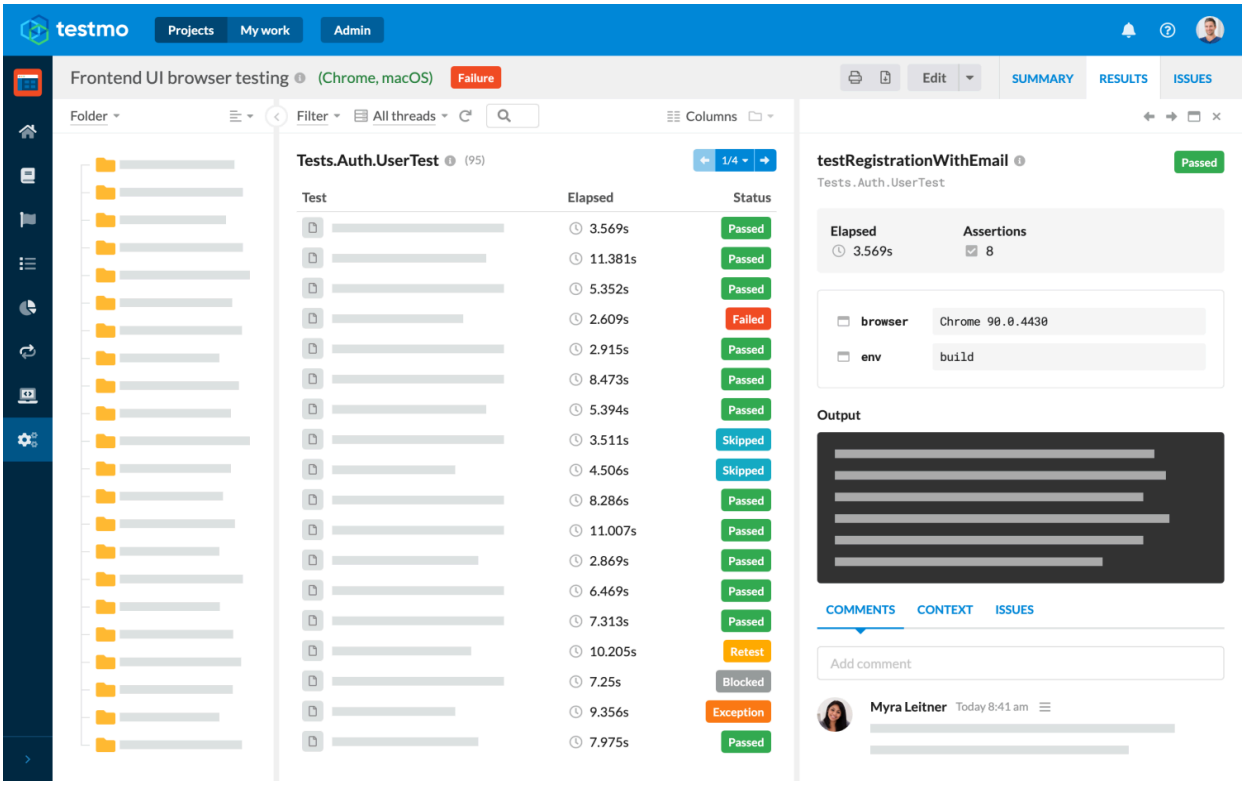


Рис. 3.5 Інструмент BrowserStack

База даних є ключовою частиною архітектури гри. Для роботи з базою даних використовується DBeaver — універсальний інструмент, який підтримує взаємодію з різними реляційними та нереляційними базами даних. DBeaver дозволяє створювати, редагувати та виконувати запити SQL, а також відстежувати структуру бази даних і її вміст.

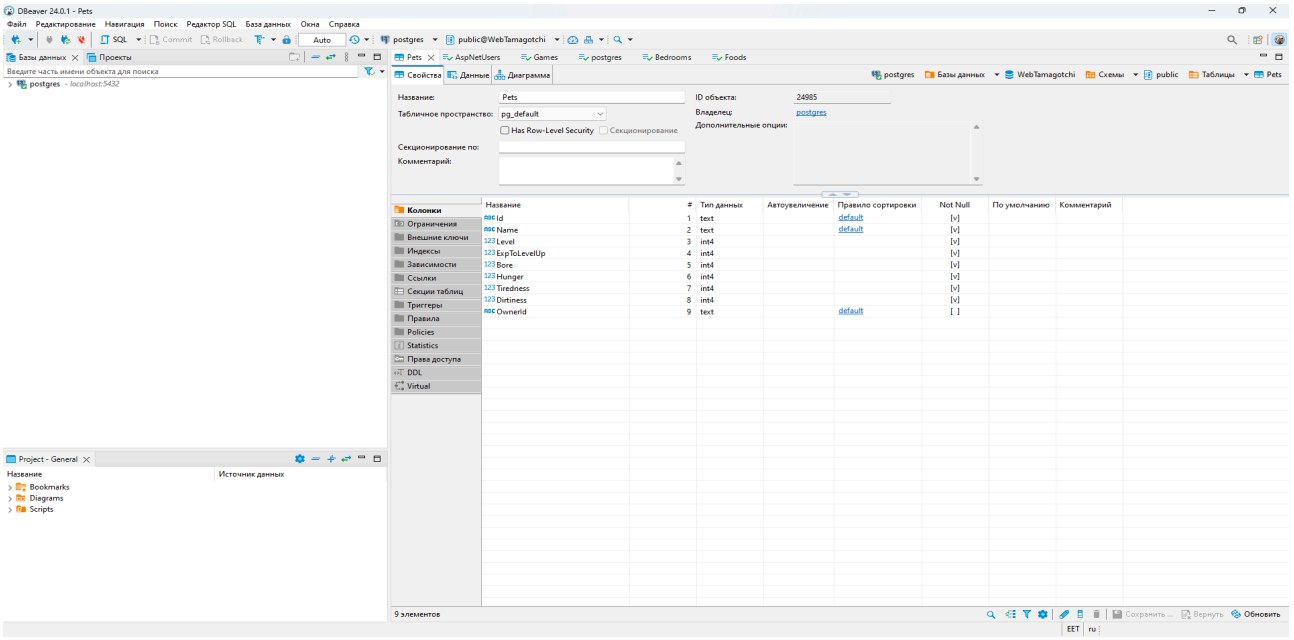


Рис. 3.6 Інструмент DBeaver

Для контролю версійності коду використовується GitHub. Цей інструмент дозволяє розробникам відстежувати зміни в коді, створювати гілки, працювати в командах та відправляти запити на злиття. GitHub також забезпечує можливість автоматизації процесів, таких як інтеграція та розгортання.

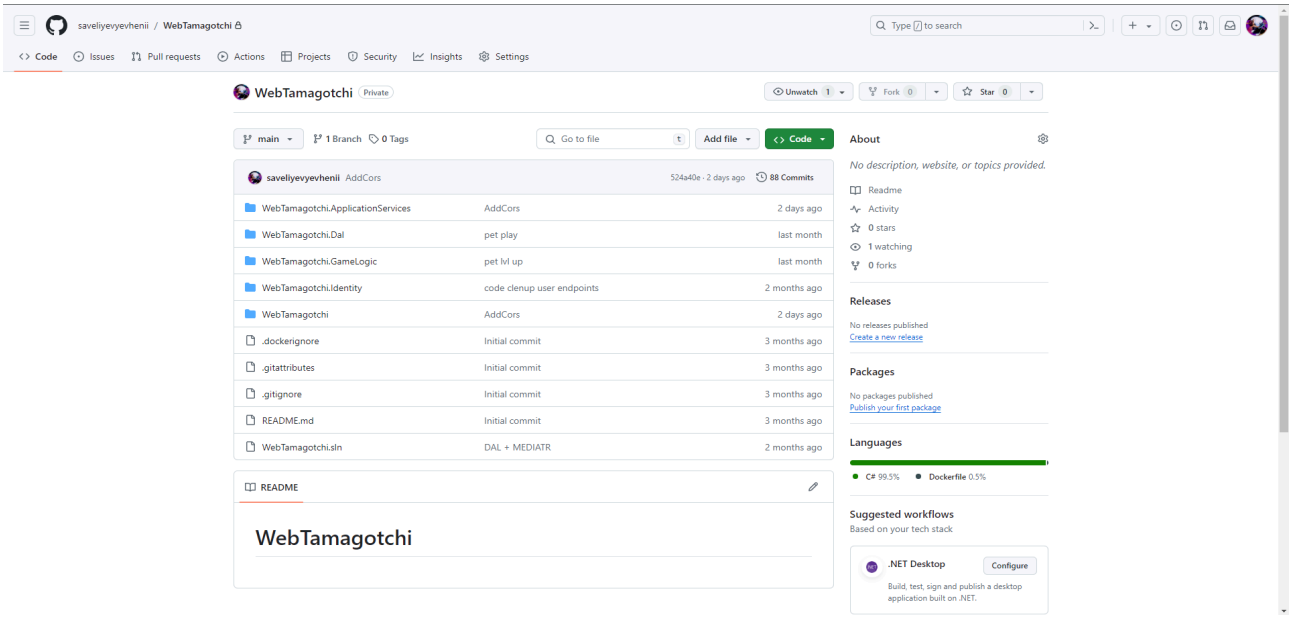


Рис. 3.7 Tamagotchi back-end GitHub репозиторій

Загалом, обрані програми для розробки гри Tamagotchi забезпечують ефективний і зручний робочий процес, дозволяючи розробникам швидко та якісно

створювати й тестувати застосунок. Інструменти для контролю версійності та тестування допомагають підтримувати високу якість коду та забезпечують плавний процес розгортання.

3.1 Архітектура та структура застосунку

Архітектура гри Tamagotchiрозроблена з урахуванням принципів модульності, розширюваності та надійності. Основні компоненти архітектури включають фронтенд, бекенд і базу даних, які взаємодіють між собою через чітко визначені інтерфейси та протоколи. Також вона забезпечує чіткий поділ відповідальності між компонентами, що дозволяє легко розширювати функціональність, а також сприяє кращому тестуванню та підтримці застосунку.

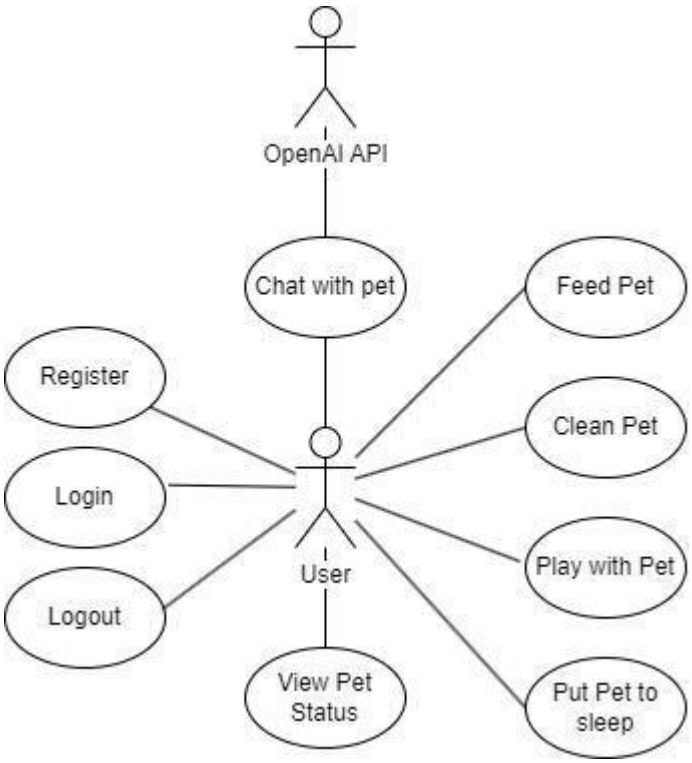


Рис. 3.8 Use Case Diagram

Діаграма прецедентів для гри Tamagotchi відображає взаємодію користувача із системою через набір визначених прецедентів. Користувач є головним актором, який взаємодіє з віртуальним персонажем (петом), доглядаючи за ним через різні дії, такі як годування, гра, чистка та укладання спати.

Основні прецеденти користувача включають:

- Реєстрація: дозволяє новому користувачу створити акаунт.
- Вхід: забезпечує доступ існуючого користувача до свого акаунту.
- Перегляд стану пета: показує поточний стан здоров’я, голоду, чистоти та енергії пета.
- Годування пета: збільшує рівень голоду пета.
- Гра з петом: зменшує рівень нудьги пета.
- Чистка пета: покращує чистоту пета.
- Укладання пета спати: збільшує рівень енергії пета.
- Чат-бот з петом: доволяє користувачам спілкуватися з петом в реальному часі

3.1.1 Розробка Фронтенду

Фронтенд відповідає за інтерфейс користувача та забезпечує взаємодію з користувачем. Для розробки фронтенду використовується мова програмування Type Script та Framework Angular, що дозволяє створювати динамічні та реактивні інтерфейси. Framework Angular забезпечує двосторонню прив'язку даних і модульну структуру, що дозволяє легко масштабувати та підтримувати застосунок. Для покращення інтерфейсу використовуються Angular Material, який забезпечує набір стильних та зручних у використанні компонентів, таких як кнопки, картки, панелі та діалогові вікна.

Основні компоненти фронтенду включають:

- Компоненти Angular: окремі частини інтерфейсу, такі як панель інструментів, вікна персонажа, індикатори стану тощо. Використання Angular Material дозволяє швидко створювати сучасні та зручні компоненти інтерфейсу.
- Сервіси Angular: містять бізнес-логіку та функціональність, що використовується кількома компонентами. Сервіси дозволяють обробляти дані та взаємодіяти з бекендом через REST API.
- Модулі Angular: групи компонентів і сервісів, що забезпечують

функціональну ізоляцію та організацію коду. Такий підхід полегшує розробку та підтримку фронтенду.

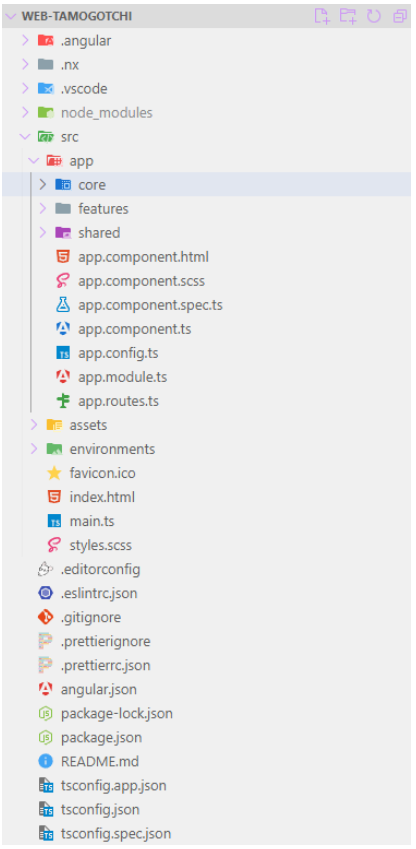


Рис. 3.9 Структура фронтенд проекту

Структура додатку розділена на окремі модулі: features, shared, core. Кожен з цих модулів має своє конкретне призначення, що дозволяє легше керувати залежностями між компонентами та сервісами, а також поліпшує розділення коду. Наприклад, модулі функціональності (features) містять компоненти та сервіси, специфічні для певної функціональності додатку, тоді як модуль shared містить спільні ресурси, які можуть бути використані в різних частинах додатку.

Директорія shared містить компоненти, директиви та сервіси, які можуть бути повторно використані в різних частинах додатку. Наприклад, компонент header, розташований у shared/components/header, може бути використаний у кількох модулях функціональності без необхідності його дублювання. Це не тільки зменшує обсяг коду, але й полегшує його підтримку: зміни, внесені до

спільного компонента, автоматично відображаються у всіх місцях, де він використовується.

Усі файли та директорії згруповані за їх призначенням, що полегшує навігацію та розширення проекту. Наприклад, всі компоненти розташовані у відповідних піддиректоріях у features або shared, а сервіси — у services. Така організація структури робить проект більш прозорим і зрозумілим для розробників. Новий учасник команди може легко зрозуміти структуру проекту і знайти необхідні файли без зайвих зусиль.

Для покращення продуктивності застосовується техніка Lazy Loading, яка дозволяє завантажувати модулі функціональності (features) динамічно, тільки коли вони дійсно потрібні користувачу. Це значно зменшує час початкового завантаження додатку і покращує користувацький досвід, особливо у великих додатках з багатьма функціональностями. Використання Lazy Loading допомагає оптимізувати використання ресурсів і забезпечити більш швидку реакцію додатку на дії користувача, зменшуючи навантаження на систему та прискорюючи процес взаємодії.

Для забезпечення якості коду на фронтенді використовуються інструменти Prettier та ESLint. Prettier автоматично форматує код згідно з визначеними стандартами стилю, що допомагає підтримувати його читабельність і єдину структуру. ESLint, зі свого боку, виконує статичний аналіз коду, виявляючи потенційні помилки, некоректні практики та забезпечуючи дотримання визначених правил написання коду. Використання цих інструментів сприяє підвищенню якості коду, полегшує його обслуговування та зменшує кількість помилок, що в кінцевому підсумку підвищує загальну надійність і продуктивність гри.

Одним із ключових аспектів фронтенду є реалізація механізмів логіну та реєстрації. Це включає створення форм для введення даних користувача, таких як ім'я користувача, електронна пошта та пароль. Angular Material надає зручні компоненти для створення цих форм, забезпечуючи перевірку валідації та зворотний зв'язок для користувачів. Після успішного логіну користувачі можуть

отримати доступ до основного інтерфейсу застосунку.

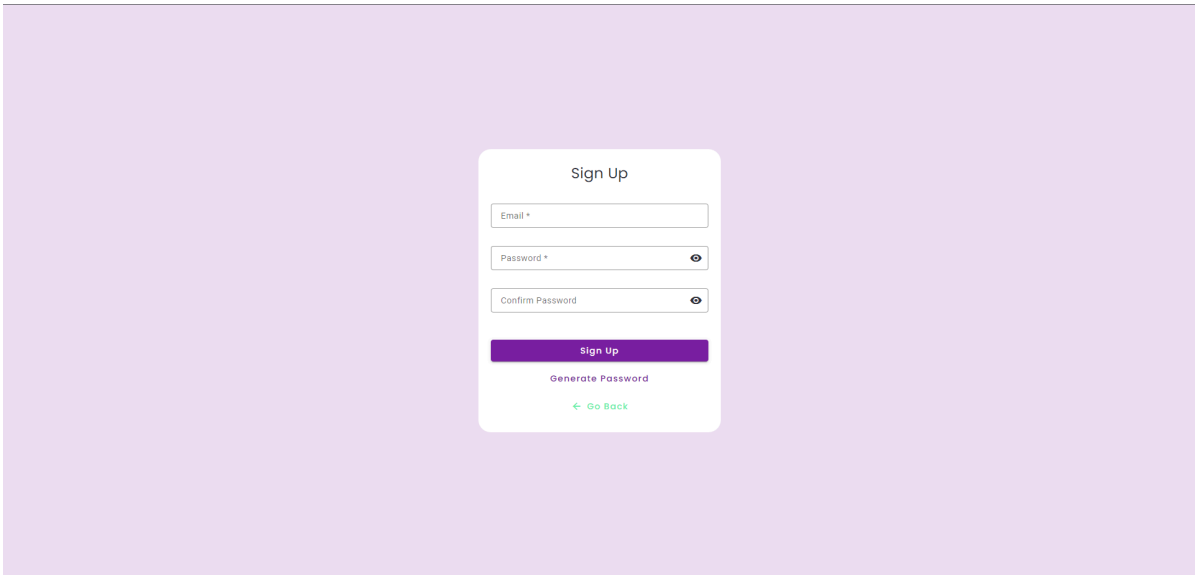


Рис. 3.10 Sign Up сторінка проекту

Для зручності користувачів на формі реєстрації була створена кнопка "Generate Password", яка дозволяє автоматично згенерувати надійний пароль. Ця функція допомагає користувачам уникнути складнощів з вигадуванням складних паролів, забезпечуючи при цьому високий рівень безпеки. Автоматично згенерований пароль містить комбінацію великих і малих літер, цифр та спеціальних символів, що значно ускладнює його зламування. Крім того, користувачі можуть зберегти або скопіювати згенерований пароль для подальшого використання, що робить процес реєстрації більш простим і безпечним.

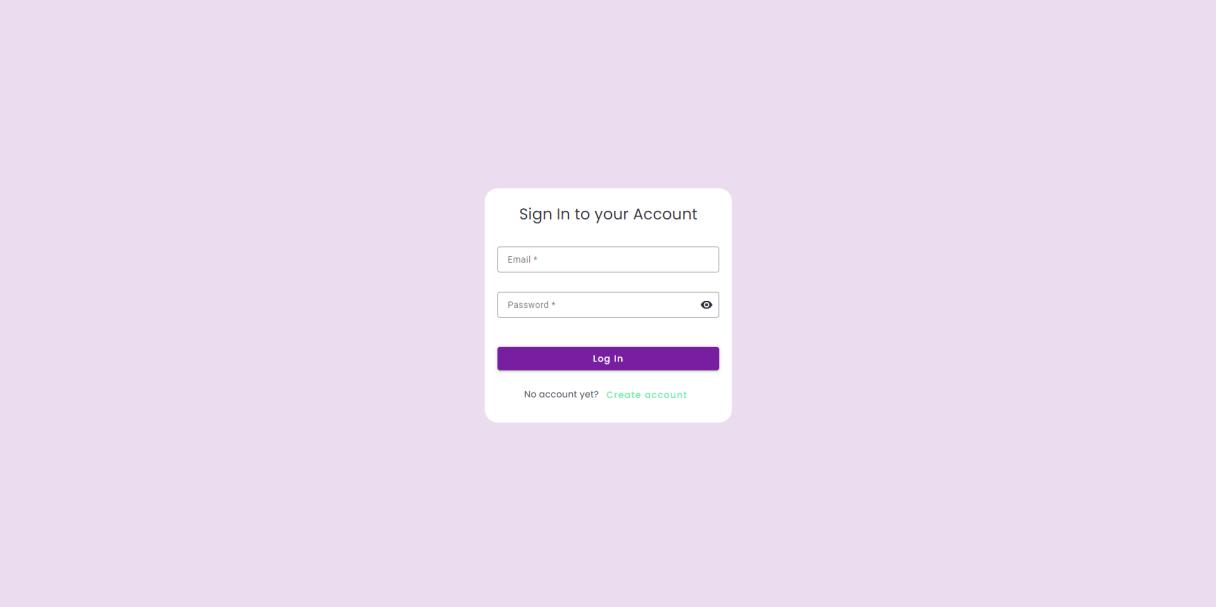


Рис. 3.11 Sign In сторінка проекту

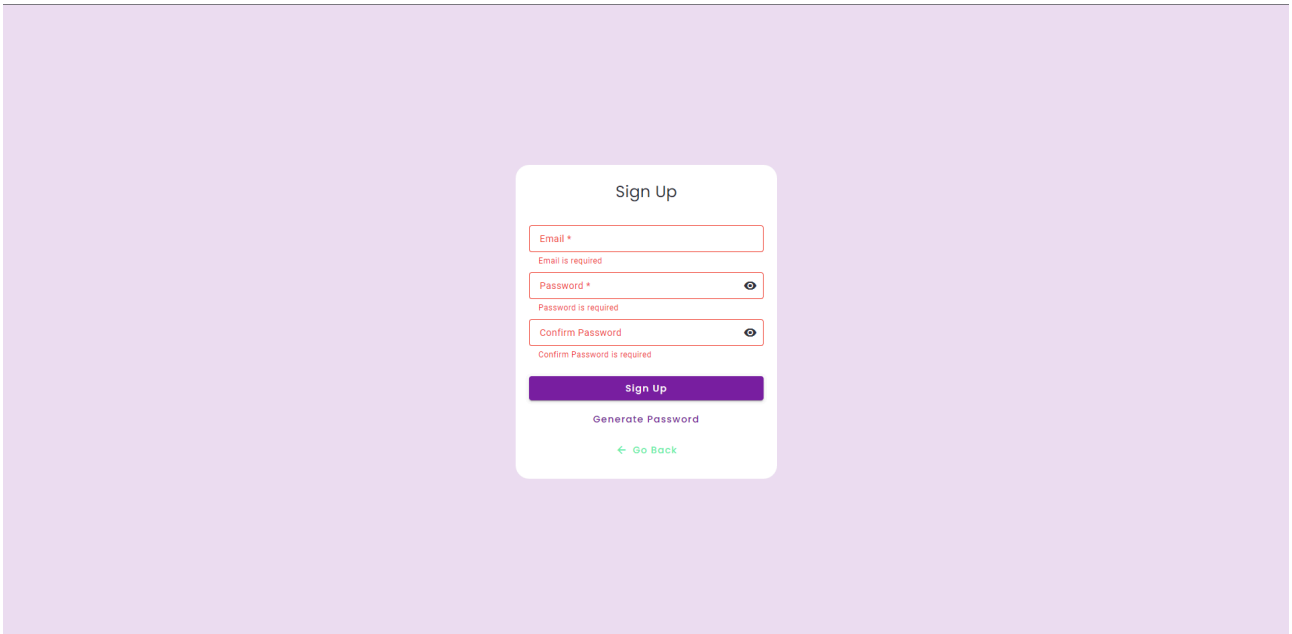


Рис. 3.12 Sign Up валідація полів

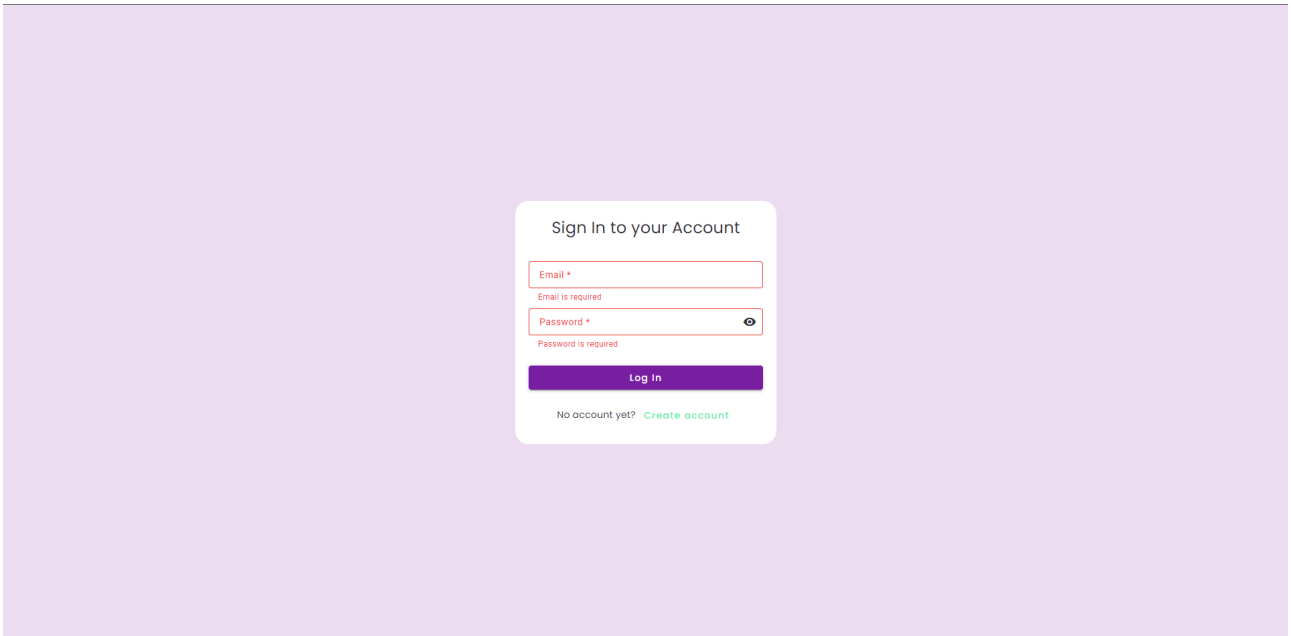


Рис. 3.13 Sign In валідація полів

Валідація полів забезпечує перевірку даних, введених користувачем, до того, як вони будуть надіслані на сервер. Це дозволяє попередити помилки, підвищити безпеку та забезпечити позитивний користувацький досвід.

Після логіну користувачі потрапляють на основний екран з тамагочі. Цей екран відображає віртуального персонажа та його поточний стан. Angular дозволяє створювати динамічний та інтуїтивно зрозумілий інтерфейс, де користувачі можуть взаємодіяти зі своїм тамагочі, годувати, розважати та доглядати за ним. Angular Material допомагає створювати привабливі та зручні у використанні елементи, такі як кнопки, індикатори стану та діалогові вікна для взаємодії з персонажем.

Рис. 3.14 Головна сторінка застосунку

На головній сторінці застосунку в верхньому правому куті розташована меню з юзернеймом аккаутна та кнопкою "Вийти". Ця кнопка дозволяє користувачам безпечно завершити свою сесію та вийти з аккаунту. При натисканні на кнопку "Вийти", користувач буде перенаправлений на сторінку логіну, де може увійти знову або створити новий аккаунт.

Одним з ключових елементів головної сторінки є меню перегляду стану улюбленця та його рівня. Це меню включає інформацію про поточний стан нудьги, усталості, голоду та чистоти улюбленця. Крім того, користувачі можуть побачити рівень свого улюбленця, що відображає прогрес гри.

На головній сторінці також є меню активностей, яке дозволяє користувачам взаємодіяти з улюбленцем. Це меню містить різноманітні дії, які користувач може виконати, щоб покращити стан свого улюбленця, наприклад, годування, ігри, миття, або укладання спати. Кожна дія має певний вплив на стан улюбленця, допомагаючи підтримувати його здоровим та щасливим.

Центральним елементом головної сторінки є відображення улюбленця за допомогою анімацій. Улюбленець представлений у вигляді анімованого персонажа, який реагує на дії користувача в реальному часі. Анімації змінюються в залежності від стану улюбленця та обраних активностей, створюючи враження

живої взаємодії.

На головній сторінці гри ркалізована інтерактивна функція, яка може зацікавити користувачів - чат-бот з віртуальним персонажем. Цей чат-бот дозволить користувачам спілкуватися з віртуальним персонажем в реальному часі, дізнаватися корисні поради, отримувати відповіді на питання та навіть вести цікаві розмови. Це не лише додає унікальність нашій грі, але й створить ілюзію інтеракції з живими об'єктами, що підвищить його привабливість для користувачів.

Чат-бот з віртуальним персонажем може стати не лише засобом розваги, але й корисним інструментом для користувачів. Він може надавати поради та рекомендації з різних сфер життя, які можуть бути корисними для користувачів. Наприклад, віртуальний персонаж може допомагати в плануванні дня, пропонувати корисні поради зі здорового способу життя або навіть виконувати функції особистого помічника.

Крім того, чат-бот з віртуальним персонажем може створювати унікальні можливості для ігрового взаємодії. Користувачі можуть вести з персонажем цікаві діалоги, виконувати завдання або вирішувати різні головоломки разом з ним. Така взаємодія може зробити процес використання гри цікавішим та захоплюючим для користувачів, залучаючи їх у світ ігрової взаємодії з віртуальними персонажами.

Головна сторінка застосунку гри Tamagotchi поєднує в собі зручність користування та інтуїтивний інтерфейс, забезпечуючи користувачам легкий доступ до основних функцій гри та дозволяючи ефективно взаємодіяти з віртуальним улюбленцем.

3.1.2 Розробка Бекенду

Бекенд відповідає за обробку бізнес-логіки, управління даними та взаємодію з базою даних. Для бекенду використовується C# з фреймворком ASP.NET, що забезпечує високий рівень продуктивності та надійності. ASP.NET підтримує патерн Model-View-Controller (MVC), який дозволяє чітко розподілити відповідальність і структурувати код для легшого обслуговування та розширення.

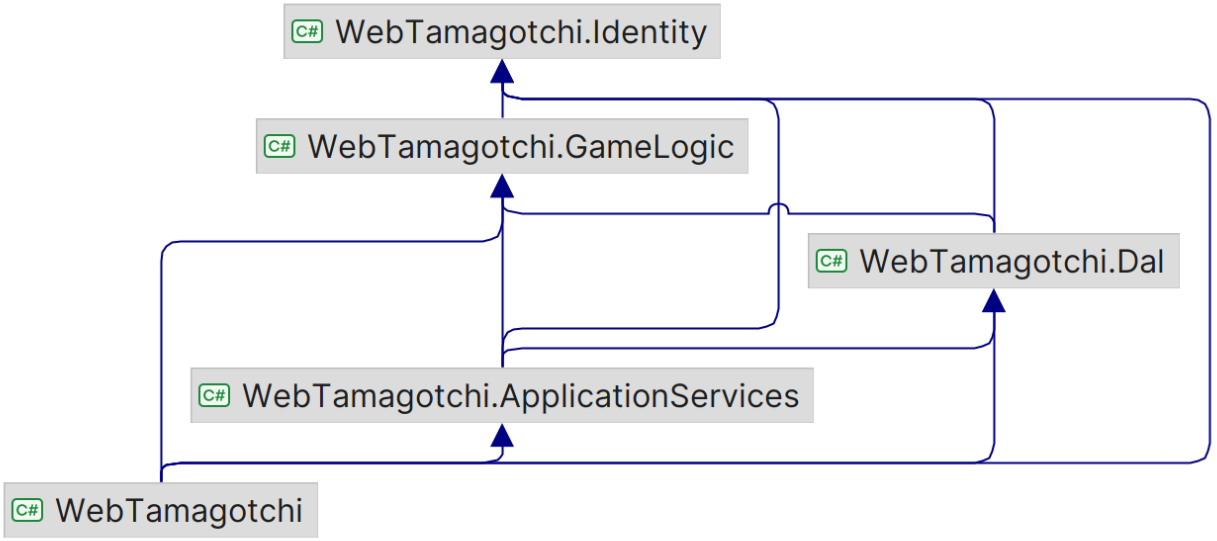


Рис. 3.15 Діаграма бекенд проекту

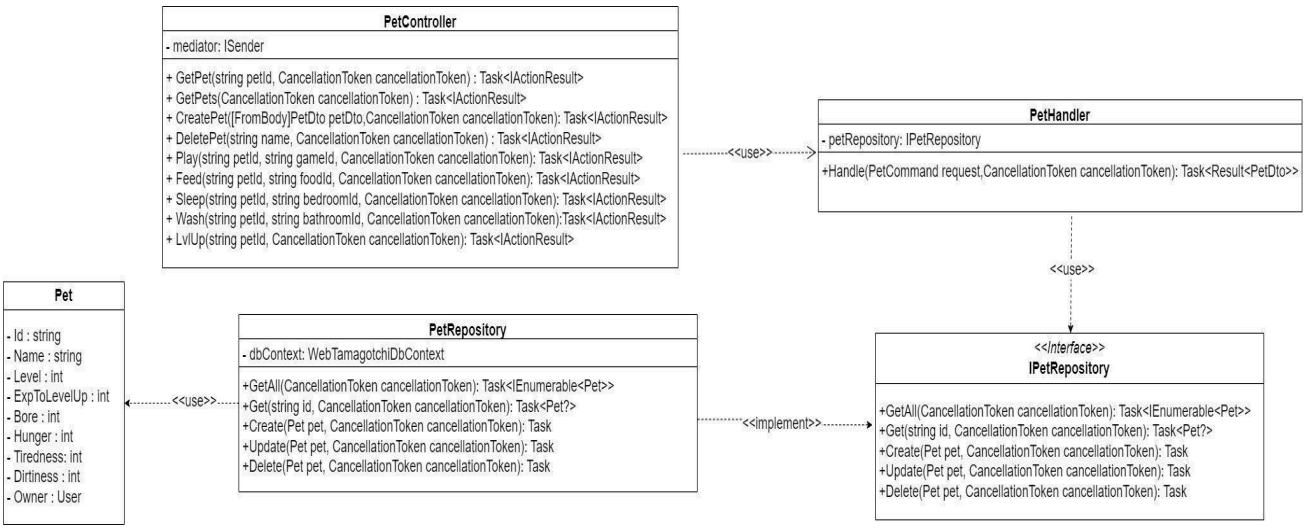


Рис. 3.16 Діаграма PetConroller

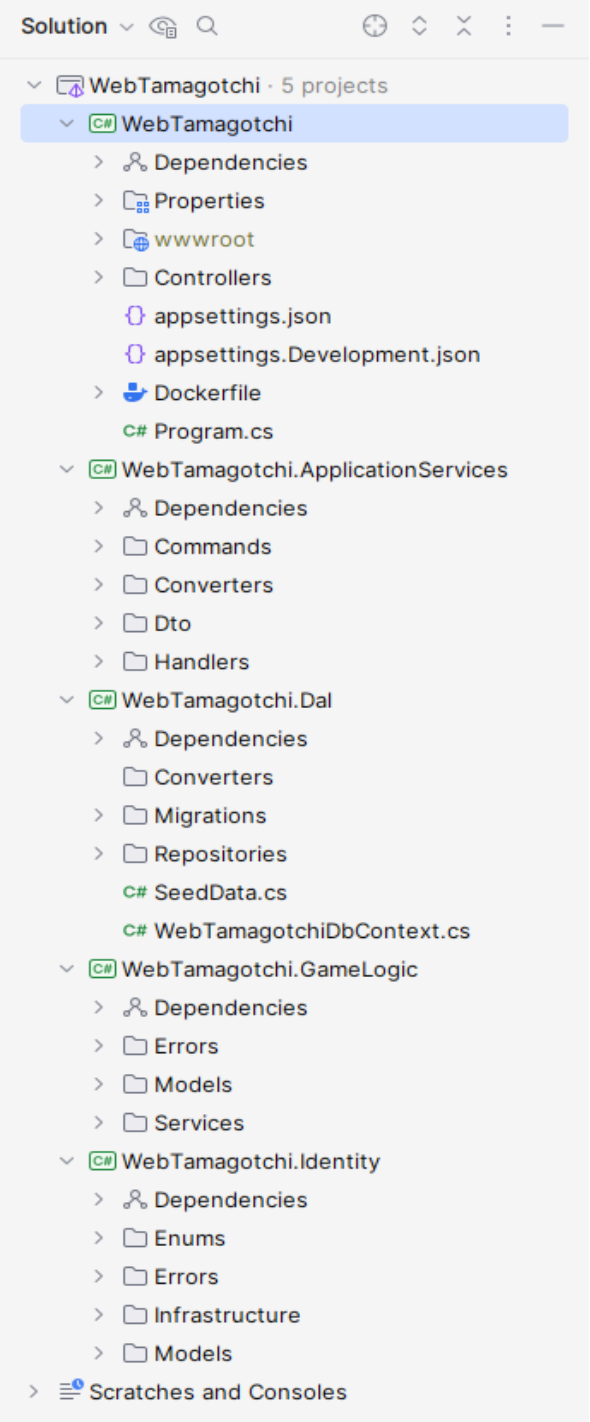


Рис. 3.17 Структура бекенд проекту

Основні компоненти бекенду включають:

- Контролери: в рамках MVC-патерну обробляють запити від фронтенду та керують взаємодією з сервісами. Вони визначають, які дії необхідно виконати на бекенді, та повертають результати у відповідному форматі.
- Сервіси: ці компоненти містять бізнес-логіку, обробляють дані та виконують

операції CRUD (створення, читання, оновлення, видалення). Сервіси взаємодіють із моделями та відповідають за виконання основних бізнес-операцій.

- Моделі: визначають структуру даних, що використовуються в застосунку. Вони містять властивості, які відповідають полям у базі даних, та забезпечують основні операції з цими даними.

Для покращення взаємодії між компонентами використовується бібліотека Mediator, яка реалізує патерн "Посередник". Mediator дозволяє контролерам і сервісам взаємодіяти через посередника, що зменшує залежність між компонентами та підвищує модульність застосунку.

На бекенді використовується бібліотека OpenAI. Ця бібліотека дозволяє нам інтегрувати в нашу гру функціонал штучного інтелекту, зокрема, застосовуючи чат-бота GPT (Generative Pre-trained Transformer). Завдяки бібліотеці OpenAI, ми можемо імплементувати чат-бота GPT у гру, що дозволить користувачам спілкуватися з віртуальним персонажем у реальному часі.

У процесі розробки гри Tamagotchi використовується інструмент Swagger для документування та тестування API. Swagger дозволяє розробникам описати структуру та функціональність API у форматі OpenAPI (раніше відомому як Swagger Specification). Завдяки Swagger, розробники можуть автоматично генерувати документацію API на основі визначень у коді. Це полегшує розуміння та використання API як в самому проекті, так і зовні. Swagger також надає можливість тестувати API прямо з інтерфейсу документації, що дозволяє впевнитися в його працездатності та правильності перед впровадженням.

У грі Tamagotchi використання Swagger допомагає забезпечити якість та доступність API для взаємодії з фронтендом та іншими системами, що покращує загальний досвід користувачів гри.

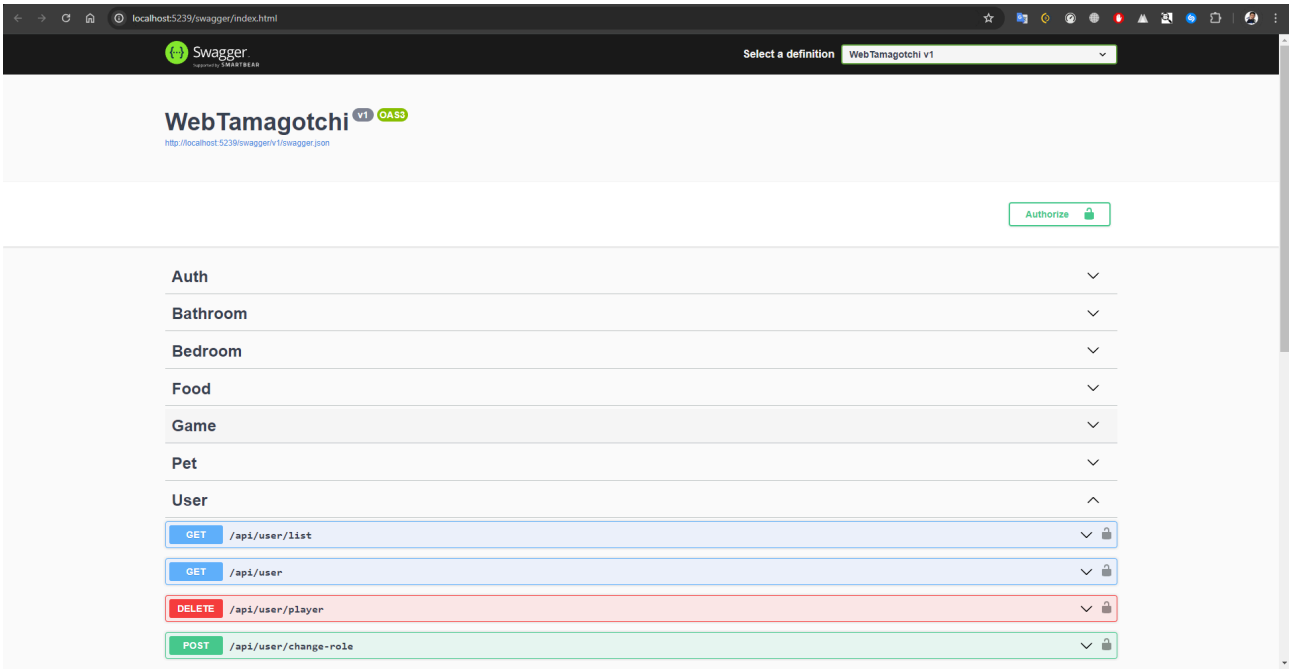


Рис. 3.18 Сторінка Tamagotchi у Swagger UI

У розробці гри Tamagotchi для вирішення завдань аутентифікації та авторизації користувачів використовується .NET Identity разом з JWT (JSON Web Token). .NET Identity — це система аутентифікації та управління доступом, яка дозволяє керувати користувачами, їх ролями та дозволами. Використання JWT дозволяє створювати та перевіряти токени доступу, які використовуються для авторизації запитів до захищених ресурсів API.

Зокрема, розроблені ендпоїнти для різних сутностей у грі (Pet, User, Bathroom, Bedroom, Food, Game, Gpt) дозволяють отримувати, редагувати, видаляти та створювати дані. Наприклад:

- Ендпоїнти для різних сутностей, таких як ванна кімната (Bathroom), спальня (Bedroom), та їжа (Food) дозволяють гравцеві збагачувати гру різноманітними взаємодіючими можливостями та функціями.
- Ендпоїнт для користувачів (User) дозволяє отримувати та оновлювати дані користувачів.
- Ендпоїнт для тваринок (Pet) надає можливість створювати нових тваринок, керувати їхнім станом, годувати і доглядати за ними.
- Ендпоїнт для аутентифікації (Auth) використовується для реєстрації та

аутентифікації користувачів у грі Tamagotchi. Цей ендпоїнт надає можливість користувачам створювати облікові записи та входити до системи, використовуючи власні облікові дані.

- Ендпоїнт для інтеграції за чатом Gpt (OpenAi) надає можливість використоувати чат та за допомогою налаштувань, спілкуватися з віртуальним персонажем в реальному часі.

3.1.3 Розробка Бази даних

База даних використовується для зберігання інформації про користувачів, персонажів та інші дані, необхідні для функціонування застосунку. Для цього проекту використовується реляційна база даних, зокрема PostgreSQL, яка відома своєю надійністю, продуктивністю та підтримкою складних запитів. PostgreSQL забезпечує можливість роботи з великими обсягами даних і має розширену підтримку реляційних можливостей.

Основні компоненти бази даних включають:

- Таблиці: зберігають різноманітні дані, включаючи інформацію про користувачів, персонажів, події, транзакції та інші критично важливі елементи для функціонування застосунку. Кожна таблиця має чітко визначену структуру, що містить поля з різними типами даних, такими як рядки, числа або дати.
- Зв'язки між таблицями: реляційна модель баз даних дозволяє встановлювати зв'язки між таблицями, використовуючи первинні та зовнішні ключі. Це забезпечує цілісність даних і дозволяє створювати складні запити для отримання інформації з кількох таблиць.
- Запити SQL: SQL (Structured Query Language) є стандартною мовою для взаємодії з реляційними базами даних. Запити SQL використовуються для виконання операцій CRUD (створення, читання, оновлення, видалення), а також для виконання складних операцій, таких як об'єднання таблиць, агрегування даних і сортування. SQL також дозволяє створювати тригери та збережені процедури для автоматизації певних завдань у базі даних.

- Індекси: для забезпечення високої продуктивності та швидкого доступу до даних використовуються індекси. Індекси дозволяють швидше знаходити потрібну інформацію та підвищують ефективність запитів.

База даних є основою для зберігання та обробки інформації в грі Tamagotchi. Завдяки реляційній структурі та потужності PostgreSQL, база даних може ефективно обслуговувати різні операції, забезпечуючи стабільну та надійну роботу застосунку.

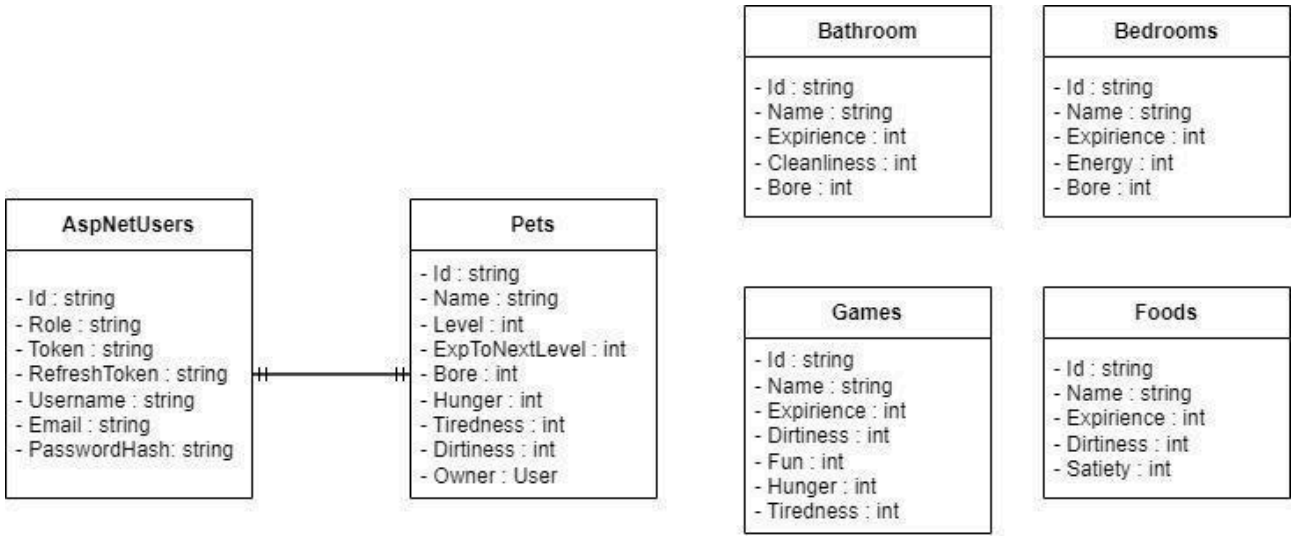


Рис. 3.19 Схема бази даних

База даних для гри Tamagotchi містить декілька таблиць, які зберігають різні типи даних, необхідних для функціонування застосунку:

1. AspNetUsers: Ця таблиця містить інформацію про користувачів, які мають облікові записи у системі. Кожен користувач має унікальний ідентифікатор (UserId), електронну адресу, хешований пароль та інші атрибути. Ця таблиця має зв'язок з іншими таблицями через зовнішні ключі.
2. Pets: Ця таблиця містить дані про віртуальних петів користувачів. Кожен пет має унікальний ідентифікатор, ім'я, характеристики, такі як голод, втома, брудність і т.д. Pets має зв'язок з AspNetUsers через зовнішній ключ UserId, який вказує на конкретного користувача, що володіє певним петом.
3. Games: Таблиця, що містить дані про ігри, які доступні у застосунку. Кожна

гра має свій унікальний ідентифікатор, назву, також очки веселощів, голоду, усталості та рівня.

4. Foods: Таблиця, в якій зберігаються дані про їжу, яку можуть отримати пети користувачів. Кожний вид їжі має свій унікальний ідентифікатор, назву, також очки ситості та рівня.
5. Bathrooms: Таблиця, яка містить дані про кімнати для ванної. Кожна ванна кімната має свій унікальний ідентифікатор, назву, також очки чистоти та рівня.
6. Bedrooms: Таблиця, що зберігає дані про кімнати для сну. Кожна спальня має свій унікальний ідентифікатор, назву, також очки енергії та рівня.

3.2 Інтеграція з базою даних і бекенд

Інтеграція з базою даних та розробка бекенду є ключовими аспектами побудови гри Tamagotchi. Цей пункт розглядає, як ці компоненти взаємодіють та забезпечують надійне зберігання, отримання та обробку даних.

Бекенд забезпечує обробку бізнес-логіки, отримання даних від фронтенду, їх обробку та повернення необхідної інформації. Він також відповідає за взаємодію з базою даних і управління даними. Для розробки бекенду використовується C# з фреймворком ASP.NET.

База даних зберігає всю інформацію, необхідну для функціонування гри. У цьому проекті використовується реляційна база даних, PostgreSQL, яка забезпечує надійне зберігання даних і підтримку складних запитів.

Для інтеграції з базою даних використовуються такі підходи:

- Entity Framework: це об'єктно-реляційний інструмент (ORM), що дозволяє працювати з базою даних за допомогою C#. Entity Framework спрощує взаємодію з базою даних, дозволяючи розробникам працювати з даними як з об'єктами.

- Запити SQL: для складних операцій або оптимізації продуктивності можуть використовуватися прямі запити SQL. Вони забезпечують більш точний контроль над взаємодією з базою даних.
- Міграції бази даних: для створення та оновлення структури бази даних використовуються міграції. Це дозволяє розробникам змінювати структуру бази даних без втрати даних і забезпечує плавний процес розробки.

Бекенд взаємодіє з фронтендом через REST API, що дозволяє передавати дані у форматі JSON . Контролери приймають запити від фронтенду, передають їх у відповідні сервіси, а потім повертають оброблені дані назад до фронтенду. Це забезпечує гнучку та ефективну комунікацію між різними частинами застосунку.

3.3 Реалізація інтерфейсу користувача та ігрової логіки

Реалізація інтерфейсу користувача та ігрової логіки є ключовими аспектами створення гри Tamagotchi. Тут описані підходи та технології, використані для створення інтуїтивного та інтерактивного інтерфейсу, а також реалізацію логіки, що забезпечує динамічну взаємодію з віртуальними персонажами.

Інтерфейс користувача (UI) є центральним елементом, через який користувачі взаємодіють із застосунком. Для його розробки використовується TypeScript та Framework Angular, який дозволяє створювати динамічні та реактивні компоненти.

Основні аспекти реалізації інтерфейсу користувача:

- Інтерфейс має бути простим та зрозумілим, щоб користувачі могли легко взаємодіяти з грою без складних інструкцій. Використання інтуїтивних елементів, таких як кнопки, значки та іконки, допомагає зробити інтерфейс зручним для користувачів.
- Інтерфейс має реагувати на дії користувачів у реальному часі. Framework Angular забезпечує двосторонню прив'язку даних, що дозволяє змінювати інтерфейс відповідно до дій користувача.
- Гра Tamagotchi використовує анімації та візуальні ефекти для додання життя

та динаміки ігровому досвіду. Анімації використовуються для взаємодії з персонажем, а візуальні ефекти – для відображення змін у стані персонажа.

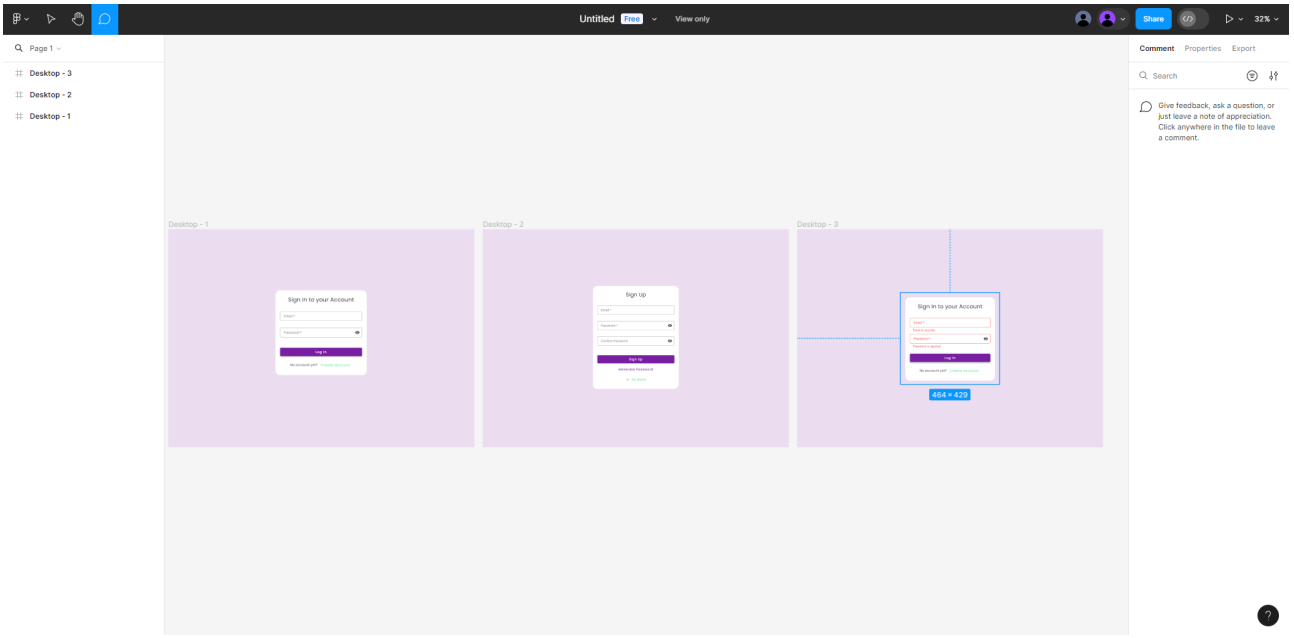


Рис. 3.20 Дизайн проект в Figma

Проектування дизайну інтерфейсу користувача (UI) для застосунку Tamagotchi здійснювалось з використанням сучасного інструменту для дизайну інтерфейсів – Figma. Figma була обрана через свої потужні можливості для спільної роботи, гнучкість і зручність у використанні.

В Figma були створені всі необхідні макети інтерфейсу, які включали головні екрани застосунку, навігаційні елементи, кнопки, форми та інші компоненти. Цей інструмент дозволив легко візуалізувати різні елементи інтерфейсу та забезпечити їхню узгодженість і гармонійний вигляд. Крім того, Figma підтримує створення інтерактивних прототипів, що дозволило тестувати взаємодію користувача з інтерфейсом ще на етапі проектування.

Дизайн інтерфейсу Tamagotchi був орієнтований на забезпечення зручності та інтуїтивної зрозумілості для користувачів. Всі елементи були розроблені з урахуванням сучасних тенденцій в UI/UX дизайні, що включає мінімалістичний підхід, використання приємних кольорових схем та анімацій для підвищення

взаємодії з користувачем. Крім того, особлива увага приділялася адаптивності інтерфейсу, щоб забезпечити його коректне відображення на різних пристроях – від настільних комп'ютерів до мобільних телефонів.

Для забезпечення анімацій та передачі стану персонажа в грі, було створено колекцію спрайтів. Спрайти представляють собою серію зображень, що відображають різні стани та дії віртуального персонажа, такі як кліпання, сон, радість та інші активності. Використання спрайтів дозволяє досягти плавних та реалістичних анімацій, що значно покращує візуальне сприйняття гри і робить взаємодію користувачів з персонажем більш захоплюючою та динамічною.

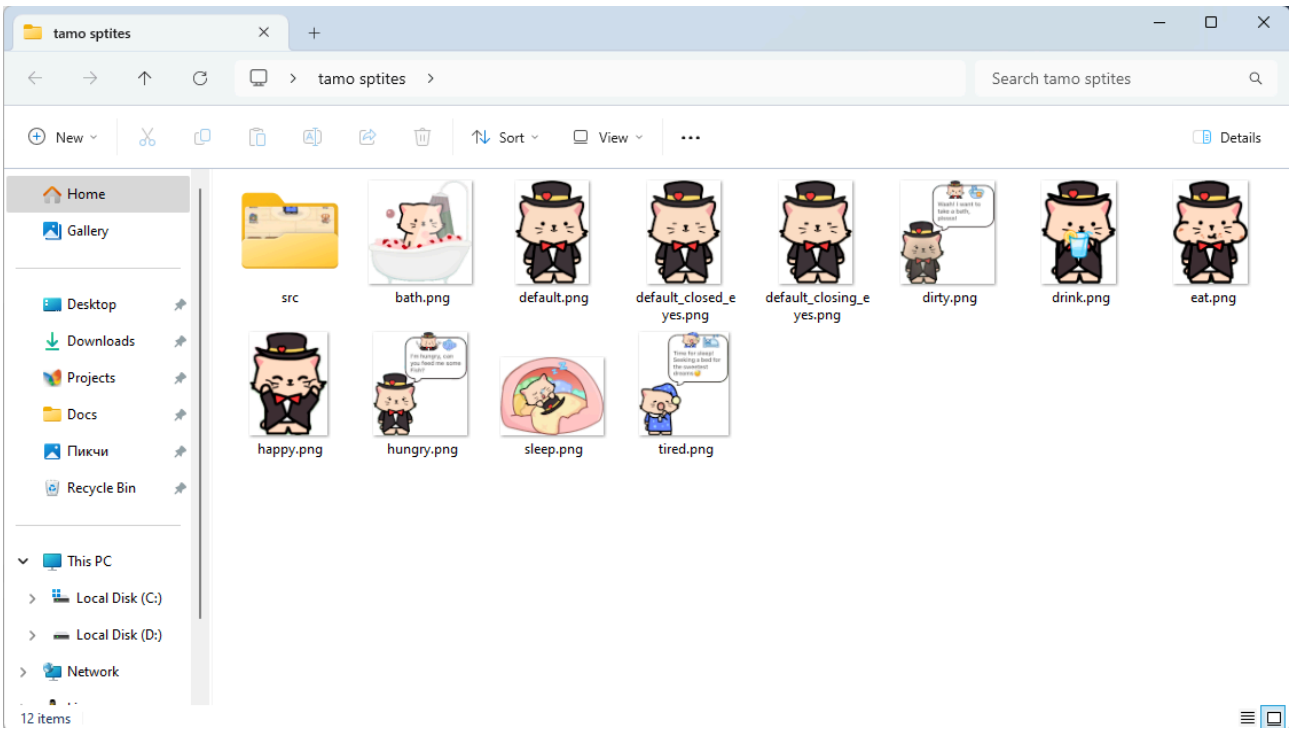


Рис. 3.21 Колекція спрайтів Tamagotchi

Логіка гри Tamagotchi базується на взаємодії користувача з його віртуальним петом. Кожен пет має свій рівень, який може збільшуватись за рахунок догляду за ним. Догляд за петом полягає у виконанні різних видів дій, що включають годування, приймання ванни, сон та ігри.

За кожен успішно виконаний догляд користувач отримує певну кількість очок рівня, які використовуються для підвищення рівня пета. Рівень пета впливає на його загальний стан і характеристики. Характеристики пета, такі як голод,

брудність, втома та скука, відображають його поточний стан і впливають на його здоров'я та настрій. Ці характеристики автоматично оновлюються під час взаємодії користувача з петом через інтерфейс користувача, а їх зміни зберігаються та оновлюються у базі даних за допомогою бекенда.

Ігрова логіка визначає, як користувачі взаємодіють з грою, та включає механіки догляду за персонажем і розвитку ігрового процесу. Основні аспекти ігрової логіки:

- Віртуальний персонаж має різні параметри, такі як рівень, здоров'я, енергія, настрій тощо. Ігрова логіка забезпечує зміну цих параметрів у залежності від дій користувача.
- Користувачі можуть виконувати завдання та різноманітні дії, щоб підтримувати та розвивати свого персонажа. Це включає годування, ігри та інші форми взаємодії.
- Ігрова логіка включає систему досягнень шляхом підвищення рівня віртуального персонажем, який мотивує користувачів доглядати за своїм віртуальним улюбленцем.

Взаємодія між інтерфейсом користувача та ігровою логікою здійснюється через фреймворк Angular, де компоненти інтерфейсу зв'язуються з сервісами, які містять ігрову логіку. При взаємодії з користувачем, компоненти надсилають запити до сервісів, які обробляють логіку ігрового процесу та повертають результати для відображення в інтерфейсі.

У результаті, реалізація інтерфейсу користувача та ігрової логіки у грі Tamagotchi забезпечує інтерактивний та захоплюючий досвід для користувачів, дозволяючи їм взаємодіяти з віртуальним персонажем та виконувати різні завдання.

3.4 Функціональне та модульне тестування застосунку

Функціональне та модульне тестування гри Tamagotchi є критично важливими етапами розробки, які забезпечують якість, надійність та коректність роботи застосунку. Тестування дозволяє виявити помилки, перевірити, чи відповідає функціонал вимогам, і гарантує, що всі модулі взаємодіють належним чином.

3.4.1 Функціональне тестування

Функціональне тестування орієнтоване на перевірку того, чи відповідають різні частини застосунку запланованій функціональності. Воно охоплює наступні аспекти:

- Тестування інтерфейсу користувача: це включає перевірку елементів інтерфейсу, таких як кнопки, поля введення, меню, а також перевірку їхньої взаємодії з користувачем. Тестування включає також перевірку реакції на користувацькі дії та валідацію даних, введених користувачем.
- Тестування ігрової логіки: функціональне тестування перевіряє, чи правильно працюють ігрові механіки, такі як годування персонажа, виконання завдань, взаємодія з подіями, і чи оновлюються відповідні параметри (здоров'я, настрій, енергія тощо).
- Тестування API та взаємодії з бекендом: це включає перевірку REST API, яким фронтенд взаємодіє з бекендом. Тестування перевіряє, чи відповідають відповіді API очікуваним даним, і чи обробляються запити правильно.

Таблиця 3.1

Тест кейси функціонального тестування

Номер	Назва	Попередні умови	Кроки	Очікуваний результат
ТС-01	Реєстрація нового користувача	Користувач не має існуючого акаунта	1.Відкрити сторінку реєстрації 2.Ввести валідні дані у всі необхідні поля (ім'я, email, пароль, підтвердження паролю) 3.Натиснути кнопку "Реєстрація".	Користувач успішно зареєстрований і перенаправлений на сторінку входу або профілю
ТС-02	Вхід у систему	Користувач має існуючий акаунт	1.Відкрити сторінку входу 2.Ввести валідні email та пароль 3.Натиснути кнопку "Вхід"	Користувач успішно входить у систему і перенаправлений на головну сторінку
ТС-03	Перегляд стану пета	Користувач увійшов у систему, має створеного пета	1.Перейти на сторінку профілю пета	Відображається поточний стан пета (здоров'я, голод, чистота, енергія)
ТС-04	Годування пета	Користувач увійшов у систему, має створеного пета	1.Перейти на сторінку профілю пета 2.Натиснути кнопку "Годувати"	Рівень голоду пета зменшується, оновлений стан відображається на екрані

Продовження таблиці 3.1

Тест кейси функціонального тестування

Номер	Назва	Попередні умови	Кроки	Очікуваний результат
ТС-06	Миття пета	Користувач увійшов у систему, має створеного пета	1.Перейти на сторінку профілю пета 2.Натиснути кнопку "Мити"	Рівень чистоти пета збільшується, оновлений стан відображається на екрані
ТС-07	Укладання пета спати	Користувач увійшов у систему, має створеного пета	1.Перейти на сторінку профілю пета 2.Натиснути кнопку "Спати"	Рівень енергії пета збільшується, оновлений стан відображається на екрані
ТС-08	Вихід із системи	Користувач увійшов у систему	1.Натиснути кнопку "Вихід" у головному меню	Користувач успішно виходить із системи на сторінку входу

3.4.2 Модульне тестування

Модульне тестування є важливим етапом процесу розробки програмного забезпечення, яке дозволяє виявляти помилки на ранніх стадіях розробки. Для гри Tamagotchi модульне тестування включає перевірку окремих компонентів і функцій застосунку для забезпечення їхньої коректної роботи.

Цілі модульного тестування:

1. Забезпечення того, що кожен модуль виконує свої функції відповідно до специфікацій.
2. Виявлення та усунення дефектів на початкових етапах розробки, що знижує

витрати на їх виправлення у майбутньому.

3. Підтримання високої якості коду через автоматизовані тести, що допомагають уникнути регресій під час внесення змін.

Основні елементи модульного тестування для гри Tamagotchi:

1. Модульні тести для логіки бізнесу:
    - Годування пета: Перевірка функції, яка забезпечує зміну стану голоду пета при годуванні.
    - Гра з петом: Тестування функцій, що відповідають за зміну настрою та енергії пета під час гри.
    - Збереження стану пета: Перевірка коректності збереження та завантаження стану пета в базі даних.
2. Модульні тести для компонентів користувацького інтерфейсу:
    - Відображення інформації про пета: Перевірка правильного відображення стану пета (голод, настрій, енергія) на головній сторінці.
    - Інтерактивні елементи: Тестування кнопок та інших елементів інтерфейсу для взаємодії з петом, щоб переконатися, що вони викликають відповідні функції.
3. Модульні тести для API:
    - Запити та відповіді: Перевірка коректної обробки запитів до API для отримання та зміни стану пета.
    - Аутентифікація: Тестування механізмів аутентифікації та авторизації користувачів через API.

3.4.3 Інструменти та методи тестування

Вибір інструментів та методів тестування є ключовим аспектом забезпечення якості програмного забезпечення. Для застосунку Tamagotchi, який поєднує в собі серверну частину, побудовану на C# та .NET, та клієнтську частину на основі TypeScript і Angular, необхідно вибрати інструменти, що забезпечують комплексне тестування кожного компонента.

Для функціонального та модульного тестування використовуються різні інструменти, зокрема:

1. Jasmine та BrowserStack: для для модульного тестування компонентів і сервісів Angular.
2. XUnit: для створення і виконання модульних тестів для серверної частини C# та .NET.
3. Postman: для тестування REST API та перевірки відповіді на запити.

Таким чином, комплексний підхід до вибору інструментів та методів тестування, що охоплює як серверну, так і клієнтську частину застосунку, дозволяє забезпечити високу якість програмного забезпечення. Використання сучасних фреймворків для модульного, інтеграційного та енд-то-енд тестування дозволяє виявляти помилки на ранніх етапах розробки, підвищувати надійність системи та забезпечувати задоволеність користувачів.

4 ТЕСТУВАННЯ ЗАЛУЧЕННЯ КОРИСТУВАЧІВ ТА ОЦІНКА КОРИСТУВАЦЬКОГО ДОСВІДУ

4.1. Методика проведення тестування

Тестування проводиться з метою визначення, наскільки застосунок відповідає очікуванням користувачів, наскільки добре інтерактивний контент утримує увагу, та чи є елементи, що потребують покращення.

Методика тестування включає наступні кроки:

1. Для тестування залучається група добровольців, що представляє цільову аудиторію застосунку. Вона різноманітна за віком, гендером та досвідом у іграх.
2. Учасникам тестової групи пропонується виконати набір завдань у межах застосунку, такі як взаємодія з віртуальним персонажем, виконання завдань, участь у подіях та інші аспекти гри.
3. Під час тестування ведеться спостереження за взаємодією користувачів із застосунком. Учасники також надають зворотний зв'язок через опитування та інтерв'ю, де описують свої враження, складнощі та пропозиції щодо покращення.

4.2. Результати тестування та висновки

У цьому розділі аналізуються результати тестування та оцінюється користувацький досвід для визначення ефективності застосунку та можливих шляхів його покращення.

Основні аспекти аналізу результатів включають:

1. Оцінюється, наскільки інтерфейс застосунку зручний та інтуїтивно зрозумілий для користувачів. Враховуються кількість помилок, які допускають користувачі, а також швидкість виконання завдань.
2. Вивчається, наскільки добре застосунок утримує увагу користувачів, чи

достатньо цікавий контент і механіки, та чи мотивує він до тривалого використання.

3. Аналізуються відповіді учасників тестової групи, щоб визначити їхнє задоволення від використання застосунку, а також їхні пропозиції щодо покращення.

Результати тестування та аналізу дозволяють визначити сильні та слабкі сторони застосунку, а також надати рекомендації для подальшого розвитку та покращення користувацького досвіду. Це сприяє підвищенню якості гри Tamagotchi та його привабливості для користувачів.

ВИСНОВКИ

Проведено аналіз ігор жанру "Tamagotchi", визначено їх переваги та недоліки. Аналіз охоплював популярні ігри жанру, такі як Neopets, Pet Society, Club Penguin, щоб виявити ключові фактори, що впливають на задоволення гравців.

Розроблено концепцію гри, функціональні та нефункціональні вимоги. В рамках концепції гри було визначено основні механіки догляду за віртуальними улюбленцями, а також специфікації для інтерактивності, зручності користування та продуктивності гри.

Проаналізовано технічні засоби для розробки гри Tamagotchi. Обрано мову програмування C# та фреймворк .NET для бекенду, а також TypeScript та Angular для фронтенду, з урахуванням їхньої здатності забезпечити стабільність та масштабованість гри.

Розроблено гру Tamagotchi. Архітектура гри розроблена з акцентом на модульність та зручність обслуговування, що включало використання OpenAI API для інтеграції розумного чату з віртуальними персонажами.

Протестовано гру на відповідність вимогам. Гра пройшла ретельне тестування, яке включало перевірку функціональних можливостей та нефункціональних аспектів.

Робота пройшла апробацію на IV Всеукраїнській Науково-практичноій конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті». За результатами апробації опубліковано тези доповідей:

1. Савельєв Є.М., Дібрівний О.А. Створення гри жанру "TAMAGOTCHI" у веб-застосунку: переваги та недоліки. Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 265.

2. Савельєв Є.М., Дібрівний О.А. Впровадження гейміфікації у веб-застосунок: вплив на користувацький досвід. Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 171.

.

ПЕРЕЛІК ПОСИЛАНЬ

1. Microsoft. (2024). .NET Identity overview. URL: https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0

2. JWT Authentication with ASP.NET Core 8.0. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/?view=aspnetcore-8.0

3. Swagger. (2024). Swashbuckle.AspNetCore GitHub Repository. URL: https://github.com/domaindrivendev/Swashbuckle.AspNetCore

4. Entity Framework Core. (2024). Docs. URL: https://docs.microsoft.com/en-us/ef/core/

5. DBeaver Documentation. (2024). URL: https://dbeaver.com/docs/wiki/

6. Docker Documentation (2024). URL: https://docs.docker.com/guides/

7. Scott, A. (2020). Pro ASP.NET Core Identity: Understanding the .NET Identity Framework and Its Application to ASP.NET Core MVC Projects. Apress.

8. Freeman, A., & Robson, J. (2021). Pro ASP.NET Core MVC 6th Edition. Apress.

9. Steven Strogatz. (2017). ASP.NET Core Application Development: Building an application in four sprints (Developer Reference). Microsoft Press.

10. Eisman, M. (2021). Learning ASP.NET Core 6: Develop modern web applications with ASP.NET Core 6, Entity Framework Core 6, and .NET 6. Packt Publishing.

11. Troelsen, A., & Japikse, P. (2017). Pro C# 7: With .NET and .NET Core. Apress.

12. Galloway, J., Wilson, B., Allen, K., & Matson, D. (2020). Professional ASP.NET MVC 5. Wrox.

13. Crockford, D. (2008). JavaScript: The Good Parts. O'Reilly Media.

14. Hemann, C., & Burbank, K. (2016). Mastering TypeScript 3.0. Packt Publishing.

15. Freeman, A. (2018). Pro Angular 6. Apress.

16. Brahim, M. (2020). Reactive Programming with Angular and ngrx: Learn to Harness the Power of Reactive Programming with RxJS and ngrx Extensions. Packt Publishing.

17. Wenzel, L. (2020). Mastering Angular 10: Build modern web applications with Angular 10. Packt Publishing.

18. Dixon, R. (2021). ASP.NET Core in Action, Second Edition. Manning Publications.

19.Sharp, J. (2018). Microsoft Visual C# Step by Step (9th Edition). Microsoft Press.

20. Griffiths, I. (2019). Programming C# 8.0: Build Cloud, Web, and Desktop Applications. O'Reilly Media.

21. Horsman, A. (2019). Enterprise Application Development with C# 9 and .NET 5: Build software solutions that scale across enterprise computing environments. Packt Publishing.

22. Fritz, J., & Dykstra, B. (2018). ASP.NET Core Application Development: Building an application in four sprints. Microsoft Press.

23. Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining "gamification". In Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (pp. 9-15). ACM.

24. Huotari, K., & Hamari, J. (2012). Defining gamification: a service marketing perspective. In Proceeding of the 16th International Academic MindTrek Conference (pp. 17-22). ACM.

25. Zichermann, G., & Cunningham, C. (2011). Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps. O'Reilly Media.

26. Werbach, K., & Hunter, D. (2012). For the Win: How Game Thinking Can Revolutionize Your Business. Wharton Digital Press.

27. Burke, B. (2014). Gamify: How Gamification Motivates People to Do Extraordinary Things. Routledge.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка Tamagotchi game з використанням C# .NET та Angular(TypeScript)

Виконав студент 4 курсу
групи ПД-41
Савельєв Євгеній Максимович
Керівник роботи
доктор філософії, доцент кафедри ІПЗ Дібрівний Олесь Андрійович

Київ – 2024

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: підтримка процесу піклування про віртуального персонажу засобами C# .NET та TypeScript Angular.

Об’єкт дослідження: процес піклування про віртуального персонажу.

Предмет дослідження: гра, що має функції піклування про віртуального персонажу.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати існуючі ігри жанру "Tamagotchi". Визначити їх переваги та недоліки.
2. Розробити концепцію гри, функціональні та нефункціональні вимоги.
3. Проаналізувати технічні засоби для розробки гри Tamagotchi.
4. Розробити гру Tamagotchi.
5. Протестувати застосунок на відповідність вимогам.

3

АНАЛІЗ АНАЛОГІВ

Характеристика	Neopets	Pet Society	Club Penguine	Tamagochi game
Платформа	Web	Web	Web	Web
Цільова аудиторія	Діти	Діти та підлітки	Діти та підлітки	Діти та підлітки
Графіка	2D	2D	2D	2D
Можливість створення аккаунту	+	+	+	+
Підтримка сучасних веб-платформ (Chrome, Firefox, Safari, Edge)	+	+	-	+
Спілкування з персонажем через чат-бот штучного інтелекту	-	-	-	+

4

ВИМОГИ ДО ЗАСТОСУНКУ

Функціональні вимоги:

1. Реєстрація та аунтифікація користувачів.
2. Перегляд стану та рівня віртуального персонажа.
3. Зміна стану та рівня віртуального персонажа шляхом годування, гри, миття та сну;
4. Збереження прогресу догляду за віртуальним персонажем.
5. Віртуальний персонаж має змогу повідомляти текстом, про його поточний стан.
6. Спілкування з віртуальним персонажем шляхом використання чат-боту з штучним інтелектом.

Нефункціональні вимоги:

1. Локалізація гри англійською мовою.
2. Захист даних користувачів(хешування паролю).
3. Підтримка сучасних платформ (Chrome, Firefox, Safari, Edge).

5

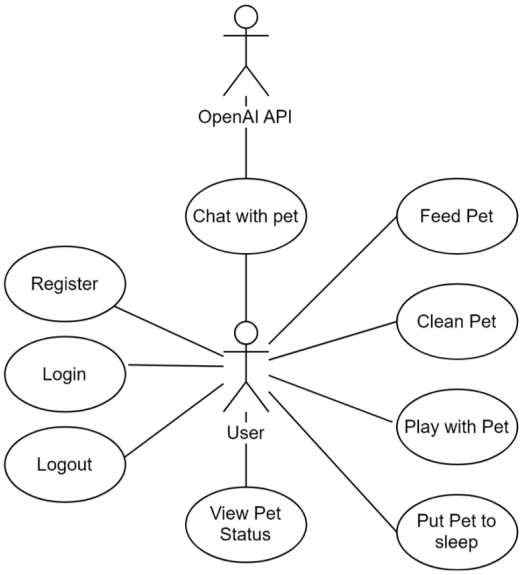
КОНЦЕПТ ГРИ

1. Жанр: "Tamagotchi"
2. Цільова аудиторія: Діти та підлітки
3. Платформа: Web з підтримкою сучасних веб-платформ.
4. Графіка: 2D.
5. Ідея: Інтерактивна гра, де користувачі доглядають та спілкуються з віртуальним персонажем. Користувачі мають змогу взаємодіяти зі своїм віртуальним персонажем через різні механіки гри, включаючи годування, сон, гра, миття для підвищення рівня персонажа та спілкуватися з ним через чат-бот штучного інтелекту.
6. Ключові особливості: чат-бот з штучним інтелектом для спілкування з віртуальним персонажем.
7. Ігрові механіки: годування, миття, гра та сон з віртуальним персонажем, підвищення рівня, чат-бот.

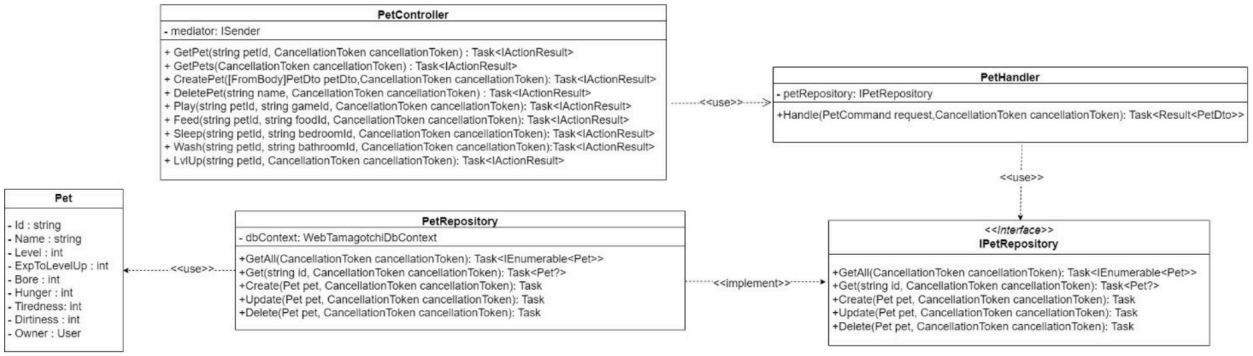
6

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ


USECASE ДІАГРАМА ЗАСТОСУНКУ



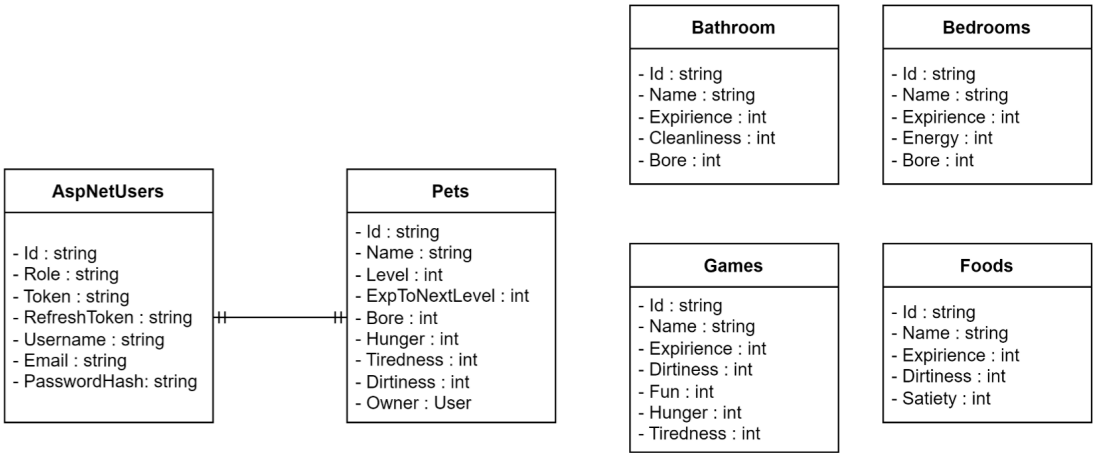
ДІАГРАМА КЛАСІВ PET CONTROLLER



09 of 16

9

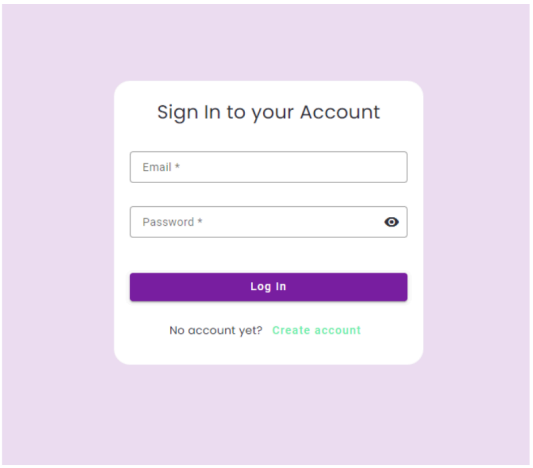
СХЕМА БАЗИ ДАНИХ ЗАСТОСУНКУ



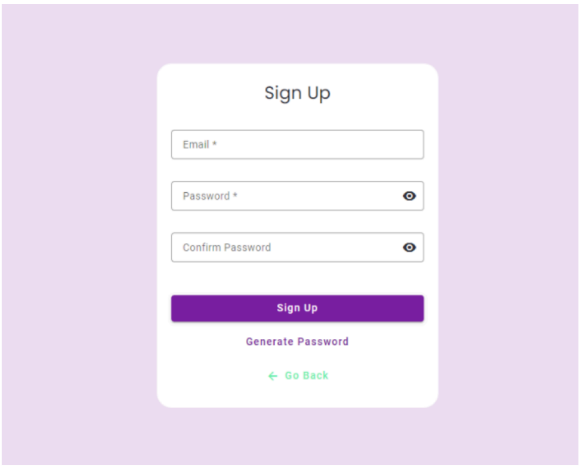
10 of 16

10

ЕКРАННІ ФОРМИ



SignIn

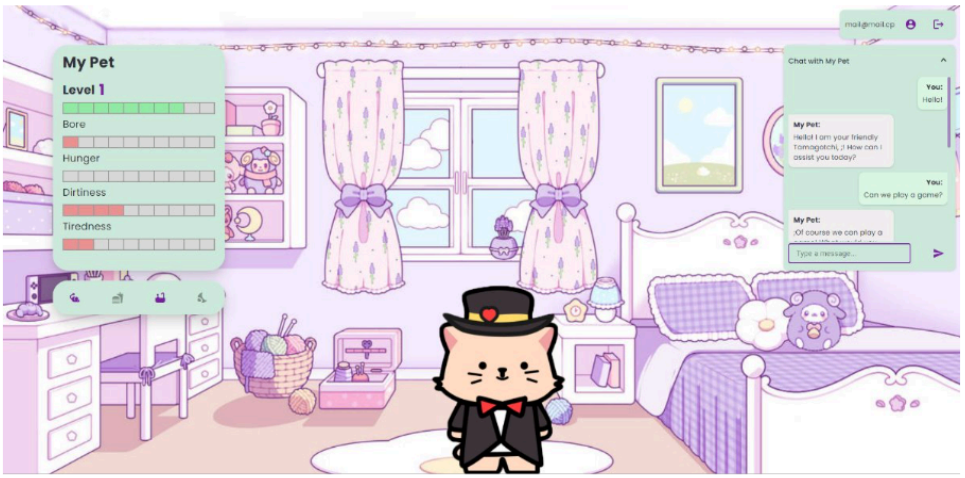


SignUp

11 of 16

11

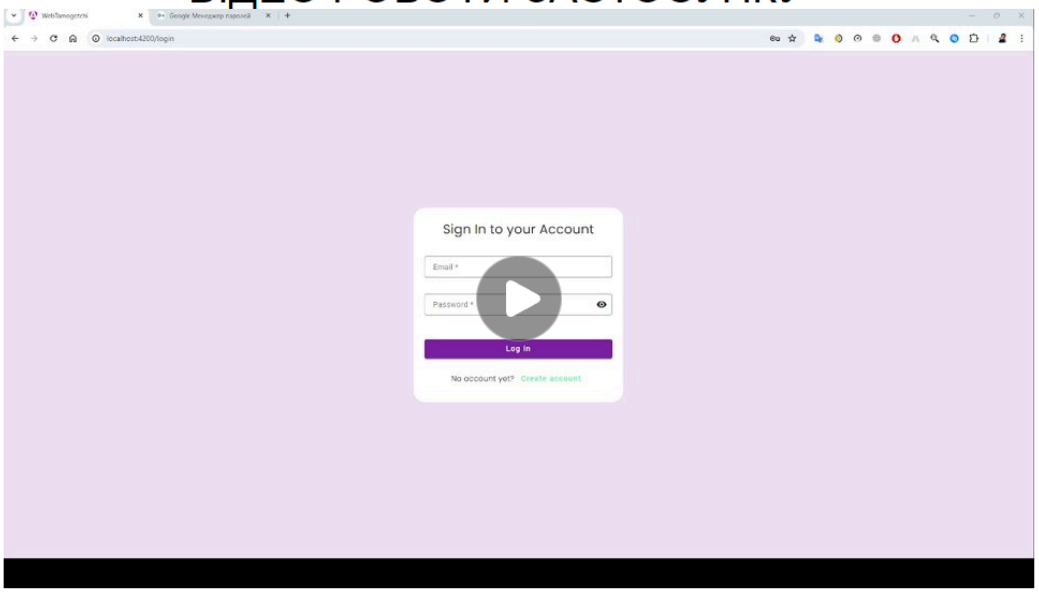
ЕКРАННІ ФОРМИ



Головна сторінка

12

ВІДЕО РОБОТИ ЗАСТОСУНКУ



13

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Савельєв Є.М., Дібрівний О.А. Створення гри жанру "TAMAGOTCHI" у веб-застосунку: переваги та недоліки. Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 265.

- Савельєв Є.М., Дібрівний О.А. Впровадження гейміфікації у веб-застосунок: вплив на користувацький досвід. Всеукраїнська Науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 171.

15

ВИСНОВКИ

1. Проведено аналіз ігор жанру "Tamagotchi", визначено їх переваги та недоліки. Аналіз показав подібність ігор та одноманітність функцій.
2. Розроблено концепцію гри, функціональні та нефункціональні вимоги. Основною перевагою гри виступає чат-бот з штучним інтелектом для спілкування з віртуальним персонажем.
3. Проаналізовано технічні засоби для розробки гри Tamagotchi. Обрано мову програмування C# з .NET, а також TypeScript з Angular. Для чат-боту обраний відкритий API від OpenAI.
4. Розроблено гру Tamagotchi. Розроблені процеси аунтефікації, догляду за віртуальним персонажем, та чат-бот з штучним інтелектом.
5. Протестовано гру на відповідність функціональних та нефункціональних вимог.