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

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка мобільного додатку системи управління розумного будинку з використанням елементів гейміфікації»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

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

_____ Іван ГОЙНА
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. ІСД- 41
Іван ГОЙНА
Ім'я, ПРІЗВИЩЕ
Керівник: к.т.н.,доцент Ольга ПОЛОНЕВИЧ
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем
Ступінь вищої освіти бакалавр
Спеціальність Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС
_____Каміла СТОРЧАК
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
Гойна Івану Сергійовичу

(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Розробка мобільного додатку системи управління розумного будинку з використанням елементів гейміфікації

керівник кваліфікаційної роботи Ольга ПОЛОНЕВИЧ к.т.н, доцент
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36

2. Строк подання кваліфікаційної роботи «31» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи:
1. Науково-технічна література з теми бакалаврської роботи.
2. Принцип функціонування «розумного будинку».
3. Основні принципи гейміфікації.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Гейміфікація. Визначення та прийоми
2. Інструменти та прийоми розробки мобільних додатків
3. Розробка мобільного додатку системи управління розумного будинку з використанням елементів гейміфікації

5. Ілюстративний матеріал: *презентація*

6. Дата видачі завдання: «27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	27.02-05.03.2024	
2	Обгрунтування актуальності роботи	06.03-11.03.2024	
3	Аналіз основних прийомів гейміфікації	12.03-27.03.2024	
4	Інструменти та прийоми розробки мобільних додатків	28.03-10.04.2024	
5	Розробка мобільного додатку системи управління розумного будинку з використанням елементів гейміфікації	11.04-15.05.2024	
7	Оформлення роботи: вступ, висновки, реферат	16.05-22.05.2024	
8	Розробка демонстраційних матеріалів	23.05-24.05.2024	

Здобувач вищої освіти _____ Іван ГОЙНА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Ольга ПОЛОНЕВИЧ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавр: 54 стор., 43 рис., 6 джерел.

Мета роботи – покращення дизайну застосунків для управління розумного будинку за рахунок впровадження в його функціональну архітектуру елементів гейміфікації.

Об'єкт дослідження- управління системою «розумного будинку».

Предмет дослідження – елементи гейміфікації у застосунку управління «Розумного будинку».

Короткий зміст роботи: У роботі проаналізовано можливості використання елементів гейміфікації при розробці додатків, на прикладі додатку управління розумним будинком. Використовуючи UML створено чітку архітектуру, яка відповідає останнім вимогам розробки програмного забезпечення.

У застосунку було випробувано використання елементів гейміфікації, таких яких як 3D представлення кімнати з розумними пристроями, що має перспективу позитивно вплинути на залучення користувачів. Розроблений додаток дозволяє користувачам віддалено взаємодіяти зі своїми розумними домашніми пристроями, що значно покращує зручність їх використання. Це підтверджує доцільність і ефективність застосування таких рішень в повсякденному житті.

КЛЮЧОВІ СЛОВА: РОЗУМНИЙ БУДИНОК, ГЕЙМІФІКАЦІЯ, ДІАГРАМИ UML, РОЗУМНІ ПРИСТРОЇ, РОЗРОБКА ДОДАТКУ, УПРАВЛІННЯ РОЗУМНИМ БУДИНКОМ

ABSTRACT

Text part of the bachelor level qualification work: 54 pages, 43 pictures, 6 sources.

The purpose of the work - is to improve the design of smart home control applications by introducing gamification elements into its functional architecture.

Object of research is the development of a smart home management application.

Subject of research the study is the elements of gamification in the Smart Home management application.

Summary of the work: The work analyzes the possibilities of using gamification elements in the development of applications, using the example of a smart home management application. Using UML, a clear architecture has been created that meets the latest software development requirements.

The application tested the use of gamification elements, such as a 3D representation of a room with smart devices, which has the prospect of positively influencing user engagement. The developed application allows users to remotely interact with their smart home devices, which greatly improves their usability. This confirms the expediency and effectiveness of applying such solutions in everyday life.

KEYWORDS: SMART HOME, GAMIFICATION, UML DIAGRAMS, SMART DEVICES, APPLICATION DEVELOPMENT

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі технології розумного будинку стрімко розвиваються та знаходять все більше прихильників серед споживачів. Однак на ринку мобільних додатків для управління розумним будинком все ще відсутні комплексні рішення, які б поєднували зручний інтерфейс, високу функціональність та елементи гейміфікації для більшого залучення користувачів у керування будинком за допомогою мобільного застосунку. Більшість існуючих додатків надають користувачам базові можливості управління пристроями, але вони часто є застарілими, надають лише текстовий інтерфейс без можливості швидкого огляду всіх пристроїв одночасно та не мають інноваційних підходів до взаємодії з користувачем.

Гейміфікація у додатках для розумного будинку є відносно новим підходом, що дозволяє зробити процес управління більш захоплюючим та мотивуючим. Впровадження елементів гейміфікації, таких як система балів, досягнень та нагород всередині заспосунку, сприяє підвищенню зацікавленості користувачів, залучає їх до активної взаємодії з системою та сприяє більш ефективному використанню ресурсів будинку.

Незважаючи на потенціал, подібні рішення ще не набули широкого розповсюдження на ринку, що робить тему дослідження особливо актуальною. Створення мобільного додатку, який поєднує в собі елементи гейміфікації, може значно покращити користувацький досвід та підвищити ефективність управління розумним будинком.

Мета роботи – покращення дизайну застосунків для управління «Розумного будинку» за рахунок впровадження в його функціональну архітектуру елементів гейміфікації.

Для виконання поставленої мети, у бакалаврській роботі виконано наступні завдання:

1. Аналіз існуючих рішень та технологій для розробки мобільних додатків управління розумним будинком.

2. Проектування архітектури мобільного додатку з використанням моделей штучного інтелекту та елементів гейміфікації.

3. Реалізація та тестування мобільного додатку на базі розробленої архітектури.

Об’єкт дослідження- управління системою «розумного будинку».

Предмет дослідження – елементи гейміфікації у застосунку управління «розумного будинку».

Методи дослідження. Під час виконання завдань бакалаврської роботи були використані методи елементів системного аналізу, методи теоретичного дослідження та імітаційного моделювання.

Наукова новизна одержаних результатів. Наукова новизна бакалаврської роботи, заключається в розробці нового підходу до інтеграції гейміфікації в мобільний додаток для управління розумним будинком, що підвищує залученість користувача, ефективність та зручність використання системи.

Практична значущість одержаних результатів полягає у створенні мобільного додатку, який забезпечує зручне та ефективне управління системою розумного будинку, підвищує зацікавленість користувачів та їх взаємодію з системою.

Апробація. Основні положення і результати бакалаврської роботи доповідались на:

- Всеукраїнській науково-технічна конференція «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і Світу». ДУІКТ, 2023.

- V Міжнародній науково-технічній конференції «Сучасний стан та перспективи розвитку IoT». ДУІКТ, 2024.

1 ГЕЙМІФІКАЦІЯ. ВИЗНАЧЕННЯ ТА ПРИЙОМИ

1.1 Теорія гейміфікації

Гейміфікація - це стратегія, яка використовує ігрові елементи в неігровому контексті для стимулювання активності учасників, заохочення та мотивації учасників до досягнення певної мети. Термін "гейміфікація" означає процес надання чомусь вигляду гри. Наразі гейміфікація використовується в різних сферах, включаючи бізнес, освіту, медицину, спорт і маркетинг. Гейміфікація виявляється ефективним інструментом для привернення уваги, мотивації та залучення аудиторії, а також досягнення різноманітних цілей - від збільшення продажів до покращення процесів навчання. У сучасному світі гейміфікація використовується в різних сферах: бізнес: компанії використовують гейміфікацію для мотивації співробітників, підвищення продуктивності та покращення взаємодії з клієнтами та партнерами.

Відповідно до останніх тенденцій, на сьогодні, найкращою практикою виокристання такої стратегії є сфера освіти. Сьогодні гейміфікація використовується для мотивації студентів, підвищення їхньої мотивації до навчання та покращення результатів.

Прикладом є відомий постачальний онлайн-курсів “Coursera”, у їх навчальному просторі використовується примітивний елемент, користувача просять поставити собі ціль конкретну кількість днів які той буде витрачати на проходження курсу, таким чином його мета буде визначена і досягти її користувачу буде простріше.

Гейміфікація виявилася потужним інструментом для залучення та мотивації аудиторії в різних галузях і є надзвичайно популярною серед бізнесу, навчальних закладів, закладів охорони здоров'я, спортивних організацій та маркетингових компаній.

1.2 Основні прийоми гейміфікації

Одним з найважливіших аспектів гейміфікації є використання різних методів для заохочення користувачів і мотивації їх до досягнення власних цілей. Не всі ігрові механіки та підходи можуть були доречними, та

Один з найпоширеніших елементів гейміфікації - це система балів. Користувачі отримують бали, коли виконують певні дії або завдання. Наприклад, виконання щоденних вправ у фітнес-додатку, вивчення нового матеріалу в освітньому додатку або покупки в магазині. Нові функції Додайте можливість "витрачати" зібрані бали на спеціалізовані послуги та продукти. Наприклад, фітнес-застосунок може обмінювати бали на абонементи в спортзал або знижки на спортивне обладнання.

Система рівнів - ще одна ефективна техніка гейміфікації. Користувачі просуваються відповідно до свого прогресу. З кожним новим рівнем користувач отримує нові можливості та винагороди. Нові можливості Персоналізація рівнів відповідно до індивідуальних особливостей користувача. Наприклад, навчальний застосунок може встановлювати рівні на основі складності матеріалу та області знань.

Система досягнень містить перелік завдань, які користувач може виконати. Кожне досягнення винагороджується спеціальною віртуальною нагородою, щоб заохотити користувачів бути активними. Нові можливості Додайте соціальну складову до досягнень, дозволивши користувачам ділитися своїми досягненнями з друзями через соціальні мережі та внутрішній чат.

Таблиці лідерів показують прогрес користувачів у порівнянні з іншими учасниками. Це створює конкурентне середовище та підвищує мотивацію користувачів. Нова функція Додана можливість створювати персоналізовані таблиці лідерів для певних груп користувачів, наприклад, друзів або колег.

Такі методи гейміфікації можна використовувати для створення цікавого та мотивуючого досвіду для користувачів вашого додатку або сервісу. Додавання

нових елементів, таких як персоналізація та соціальні функції, може ще більше підвищити ефективність гейміфікованих додатків.

1.3 Застосування у додатках управління розумним будинком

Гейміфікація в контексті додатків управління розумним будинком може забезпечити користувачам більш зручний та захоплюючий досвід використання системи "розумного будинку". У такому випадку, застосунок має реалізовувати такі особливості:

Бали та винагороди за досягнення:

Користувачі можуть отримувати бали та винагороди за виконання певних дій, таких як:

1) Ефективне використання електроенергії
2) Ощадливе користування водою
3) Підтримка будинку у чистоті

Внутрішньо ігровий магазин

Важливою частиною додатку має бути можливість витратити накопичені бали. Для цього можна створити внутрішний магазин для розміщення різного роду продуктів. Наприклад: постійні прикраси для оформлення будинку,

сезонні прикраси(новий рік, пасха та інші)

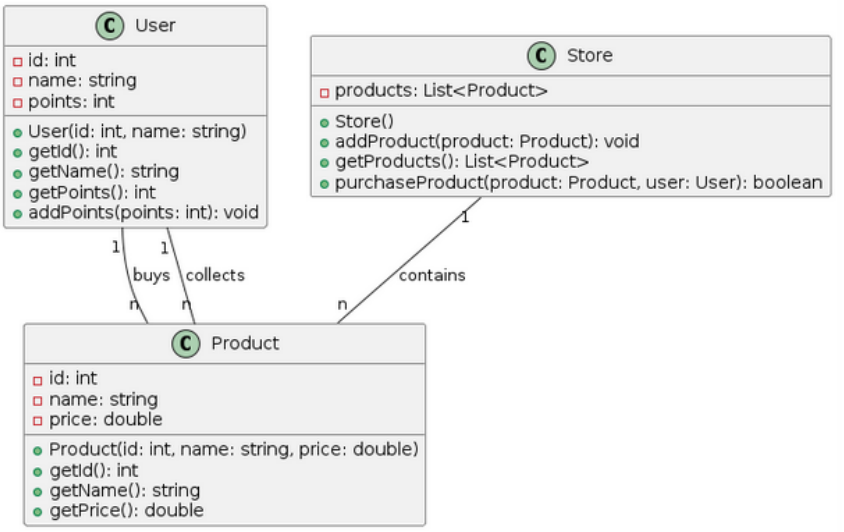

Рис.1.1 Діаграма класів балів та придбання продуктів у магазині

Рівні та досягнення:

Система рівнів може стимулювати користувачів до досягнення нових цілей та покращення їхнього досвіду використання системи "розумного будинку".

Кожен новий рівень може відкривати доступ до нових продуктів у магазині, функцій управління, можливостей або привілеїв, що мотивує користувачів до активності та залученості в управління власним будинком.

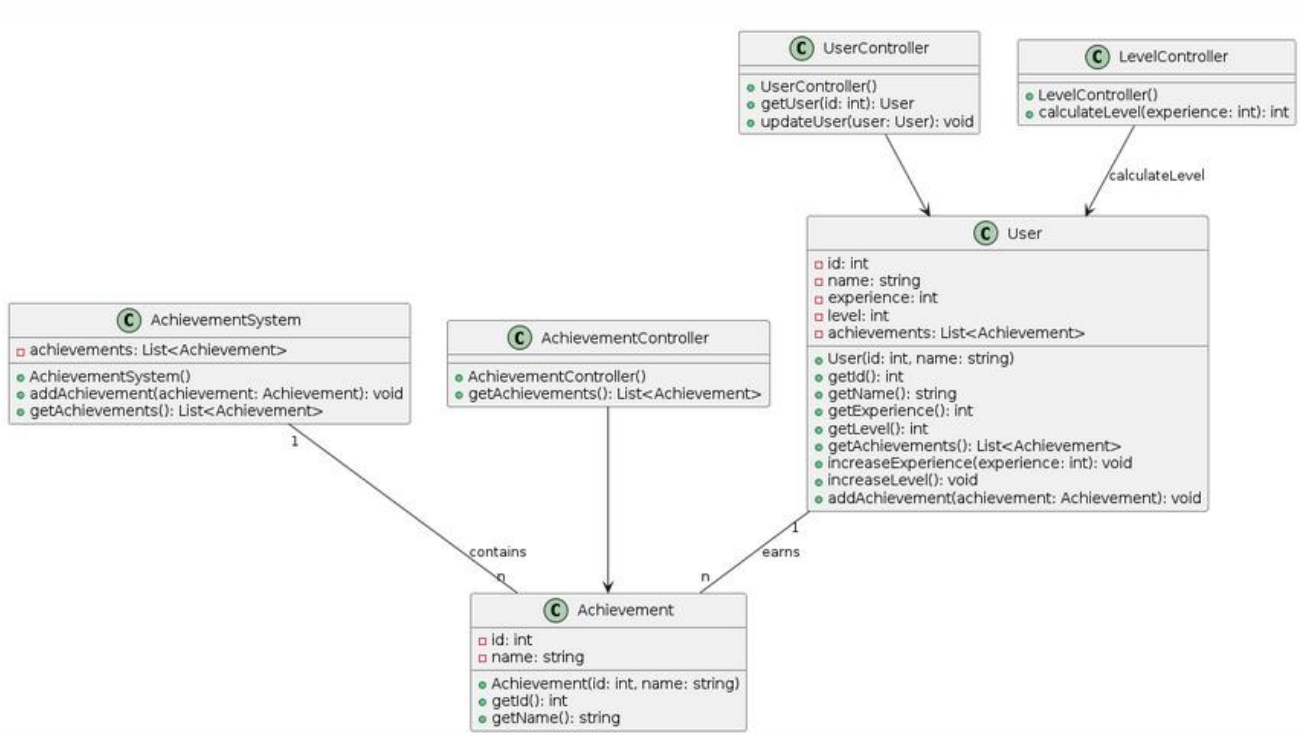

Рис.1.2 Діаграма класі системи рівнів та досягнень

Таблиця лідерів

Таблиця лідерів - це інтерактивний елемент гейміфікації, який відображає результати гравців у порівнянні з іншими учасниками гри. Вона стимулює конкуренцію між гравцями та мотивує їх до досягнення кращих результатів. Таблиця лідерів, у стандартному розумінні, складається з таких елементів:

- Ім'я користувача: Ідентифікація користувача, яка відображається в таблиці лідерів.

- Оцінка: Кількість очок або результат, який користувач отримав.

- Позиція в рейтингу: Ранжування користувач відповідно до їхніх результатів, від найкращих до найгірших.

2 ІНСТРУМЕНТИ ТА ПРИЙОМИ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

Перш ніж розпочати розробку, необхідно визначитися з інструментами та підходом до розробки мобільного додатку. У цьому розділі ми розглянемо найпопулярніші рушії для розробки мобільних додатків, такі як Android Studio, Unity, інтегровані середи розробки (Integrated Developing Environment, IDE), що є стандартами індустрії та розглянемо їх інтегрованість у рушії для розробки мобільних додатків. Також, зупинимось на інших корисних інструментах, які можуть спростити процес створення мобільних додатків.

2.1 Огляд інструментів розробки мобільних додатків

Інтегроване середовище розробки (Integrated Development Environment, IDE) — це програмне рішення, яке надає програмістам все необхідне для розробки програмного забезпечення в одному місці. IDE зазвичай включає в себе текстовий редактор, компілятор або інтерпретатор, засоби для налагодження та інші інструменти, необхідні для розробки програмного забезпечення.

Текстовий редактор — це інструмент в IDE, який дозволяє програмістам писати та редагувати вихідний код. Сучасні текстові редактори підтримують синтаксичне підсвічування, автодоповнення коду, підказки для методів та інші функції, що покращують продуктивність розробки.

Компілятор — це програма, що перетворює вихідний код, написаний на мові програмування високого рівня, в машинний код, який може виконувати процесор.

Інтерпретатор — це програма, що виконує код безпосередньо, без попередньої компіляції в машинний код. Інтерпретатор інтерпретує інтерпретовані мови програмування, популярним прикладом такої мови сьогодні є мова програмування Python.

Засоби для налагодження (debugging tools) дозволяють програмістам виявляти та виправляти помилки у програмному забезпеченні. Вони включають в

себе можливості для встановлення точок зупину (breakpoints), покрокового виконання коду, моніторингу змінних та інші корисні інструменти.

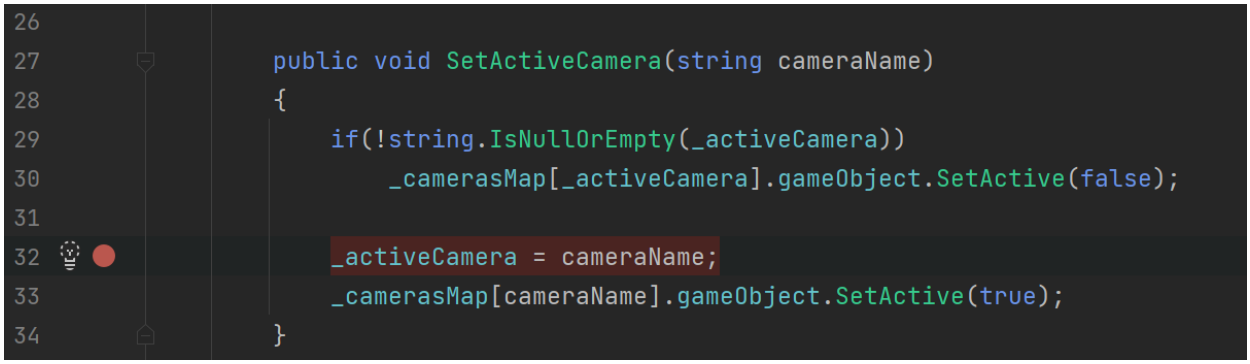

Рис.2.1 Приклад встановленої точки зупинки на тридцять другому рядку у IDE Rider

У сфері розробки ігор та додатків кілька інтегрованих середовищ розробки (IDE) виділяються як найважливіші інструменти для розробників. Ці IDE пропонують повний набір функцій та інструментів, які спрощують процес розробки ігор. Усі вони мають схожий інтерфейс але кожен з них маю свої особливості які роблять їх кращими з якоїсь точки зору. Серед таких інструментів можна виділити декілька стандартів індустрії:

JetBrains Rider

Rider IDE - це потужне та універсальне інтегроване середовище розробки, яке широко використовується для розробки ігор на популярних ігрових рушіях, таких як Unity та Unreal Engine. Воно пропонує повний набір функцій, пристосованих для розробки ігор, включаючи аналіз коду, інструменти налагодження та зручний інтерфейс. Rider забезпечує відмінну підтримку мов програмування C# та C++, що робить його ідеальним вибором для розробників ігор, які працюють з Unity або Unreal Engine. Його безшовна інтеграція з цими ігровими рушіями спрощує процес розробки та підвищує продуктивність, що робить його найкращим вибором для багатьох розробників ігор.

Особливістю цієї середи розробки є висока інтегрованість з рушієм Unity. З власного досвіду використання можу сказати, що це надзвичайно пришвидшує

процес розробки. Якщо ж звернути увагу на Unreal Engine, інтеграція інтегрованої середи розробки з цим рушієм поки не вражає через високу складність підключення Rider до проекту цього рушію.

JetBrains Rider має кілька переваг, які роблять його привабливим вибором для розробників порівняно з іншими IDE (інтегрованими середовищами розробки), що є стандартами індустрії. Ось основні плюси використання Rider:

Підтримка різних мов програмування:

Rider підтримує не лише C# та інші мови, що використовуються в екосистемі .NET, але й багато інших мов програмування, таких як JavaScript, TypeScript, HTML, CSS, SQL, і навіть деякі мови для Data Science, як-от Python. Це робить його універсальним інструментом для різних типів проєктів.

Інтеграція з екосистемою JetBrains:

Rider використовує ті самі базові технології, що й інші IDE від JetBrains, такі як IntelliJ IDEA, PhpStorm, WebStorm та інші. Це означає, що користувачі, які знайомі з іншими продуктами JetBrains, знайдуть інтерфейс і функціональність Rider знайомими та зручними.

Швидкість і продуктивність:

Rider відомий своєю високою продуктивністю. Він побудований на платформі ReSharper, але на відміну від ReSharper для Visual Studio, працює автономно, що дозволяє уникнути проблем із продуктивністю, які можуть виникати при використанні ReSharper як плагіна.

Розширені можливості рефакторингу та аналізу коду:

Rider пропонує потужні інструменти для рефакторингу, аналізу коду та виправлення помилок, які дозволяють швидко і безпечно змінювати код. Це допомагає підтримувати високу якість коду і підвищує продуктивність розробників.

Інтеграція з системами контролю версій:

Rider має вбудовану підтримку різних систем контролю версій, таких як Git та інші, що спрощує процес управління версіями коду і співпраці в команді.

Кросплатформенність:

Rider доступний для Windows, MacOS та Linux, що робить його хорошим вибором для широкого кола розробників, які працюють на різних операційних системах.

Зручний і інтуїтивний інтерфейс:

Rider має сучасний і інтуїтивний користувацький інтерфейс, який спрощує навігацію по проекту та управління кодом.

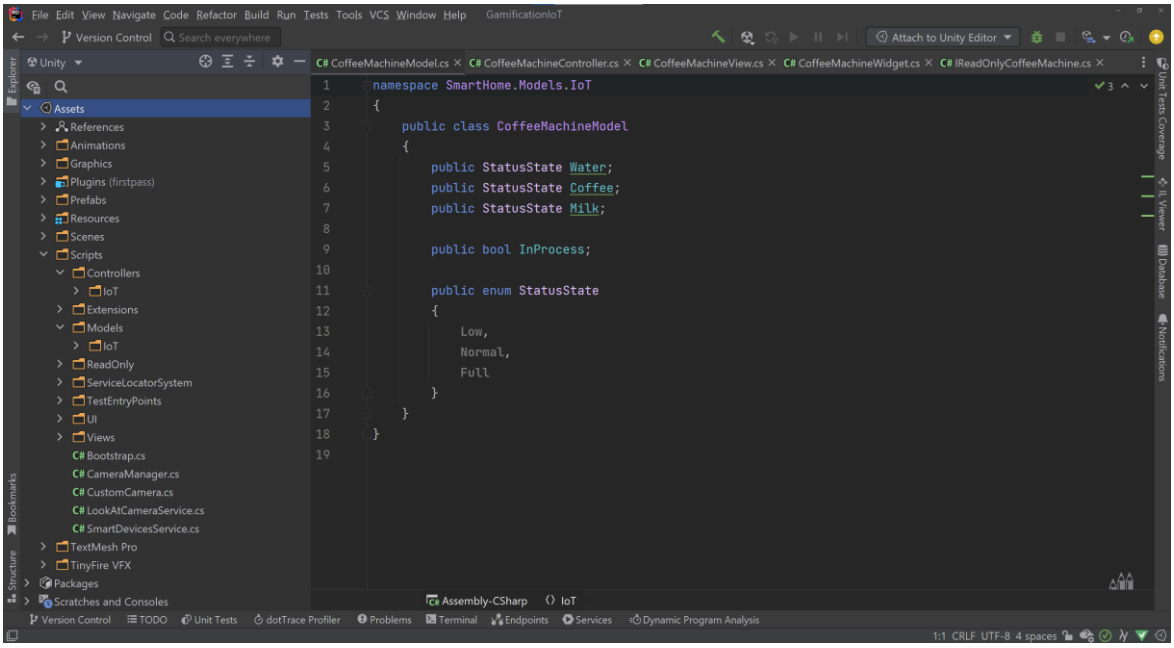

Рис.2.2 Інтерфейс інтегрованого середовища розробки JetBrains Rider

JetBrains Rider має багато переваг, але є й деякі недоліки, особливо у порівнянні з іншими IDE, а також в контексті розробки мобільних додатків. Ось основні мінуси використання Rider:

Ціна

Rider є комерційним продуктом і його використання потребує платної підписки. Це може бути суттєвим мінусом для індивідуальних розробників або малих компаній, особливо якщо вони вже використовують безкоштовні або більш доступні альтернативи.

Щоб у повній мірі скористатися всіма можливостями цього інструменту, доведеться придбати його ліцензію або набір ліцензій. У наборі можуть бути додаткові інструменти, що є плагінами до інших інструментів або окремими програмами.

Менша екосистема плагінів:

Порівняно з Visual Studio, Rider має меншу екосистему плагінів. Хоча JetBrains Marketplace пропонує багато розширень, користувачі можуть зіткнутися з обмеженнями, якщо їм будь потрібні специфічні розширення.

Потреба у високих системних ресурсах:

Більшість продуктів JetBrains, у тому числі і Rider, вимагають значних системних ресурсів для оптимальної роботи. Це може бути проблемою для розробників, які працюють на старих або менш потужних комп'ютерах.

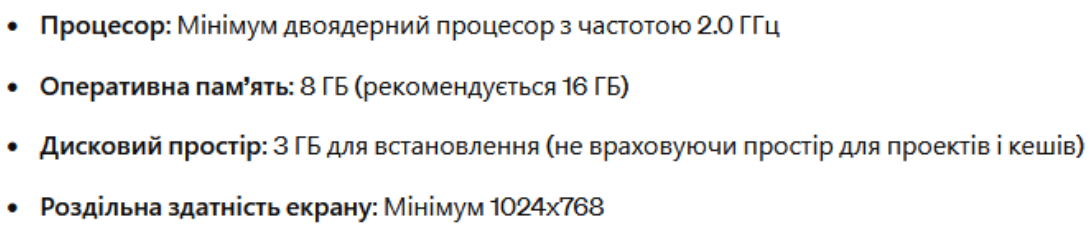

Рис.2.3 Системні характеристики для оптимальної роботи JetBrains Rider

Microsoft Visual Studio 2022

Visual Studio 2022 — це потужне інтегроване середовище розробки (IDE), яке використовується для створення програмного забезпечення для Windows, Mac, Linux, iOS та Android, включаючи веб-додатки, хмарні, класичні та мобільні додатки, служби та ігри.

Visual Studio 2022 є еволюційним кроком у розвитку лінійки Visual Studio, пропонуючи значні покращення продуктивності, нові функції та кращу інтеграцію з сучасними технологіями у порівнянні із попередніми версіями.

Visual Studio 2022 відрізняється від попередніх версій наступними ключовими характеристиками:

-Покращена продуктивність:

Завдяки оптимізації роботи, Visual Studio 2022 забезпечує швидше завантаження, зменшення затримок та загалом більш плавну роботу IDE. Це особливо важливо при роботі з великими проєктами.

-Підтримка 64-бітної архітектури:

Visual Studio 2022 є першою версією, яка повністю підтримує 64-бітну архітектуру, що дозволяє ефективніше використовувати ресурси сучасних комп’ютерів.

-Інтелектуальні інструменти для розробників:

Вбудовані інструменти на основі штучного інтелекту, такі як IntelliCode, пропонують рекомендації щодо коду, покращуючи продуктивність розробників.

-Покращений інструментарій для налагодження:

Включає нові можливості для відлагодження коду, такі як Hot Reload, що дозволяє розробникам вносити зміни до коду під час виконання програми без необхідності перезапуску.

-Підтримка різних мов програмування та платформ:

Visual Studio 2022 підтримує широкий спектр мов програмування, включаючи C#, C++, Python, JavaScript тощо. Крім того, IDE дозволяє розробляти додатки для різних платформ, включаючи Windows, Linux, Android та iOS.

-Редактор коду шейдерів:

Visual Studio підтримує розмітку синтаксису шейдерів, наприклад мови HLSL(High-Level Shader Language).

Інтерфейс Visual Studio 2022 був значно покращений для зручності користувачів. Він включає:

- Модернізований дизайн:

Інтерфейс став більш інтуїтивним та зручним, забезпечуючи легкий доступ до основних функцій та інструментів.

- Темні та світлі теми:

Користувачі можуть обирати між кількома темами оформлення для комфортної роботи в різних умовах освітлення.

- Гнучка конфігурація робочого простору:

Можливість налаштовувати панелі інструментів, вкладки та інші елементи інтерфейсу відповідно до особистих потреб і вподобань розробника.

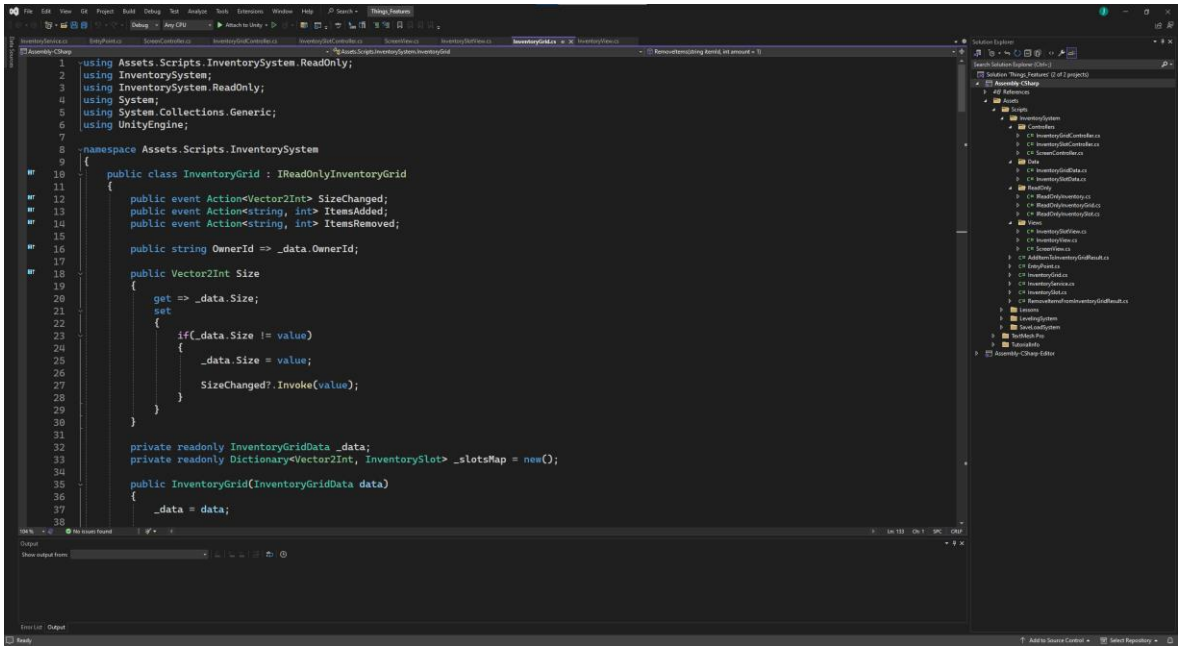

Рис.2.4 Інтерфейс інтегрованого середовища розробки Microsoft Visual Studio 2022

Також, це інтегроване середовище розробки має інтелектуальні інструменти для покращення якості коду. Ці інструменти допомагають виявляти помилки, підвищують читабельність коду та полегшують його підтримку, серед таких:

1) IntelliCode

Набір інструментів на базі штучного інтелекту, які надають рекомендації щодо коду в режимі реального часу. Основні функції IntelliCode включають:

- Рекомендації щодо коду:

IntelliCode аналізує кодову базу проєкту та пропонує контекстні рекомендації, засновані на найкращих практиках програмування. Наприклад, пропозиції по автозавершенню коду базуються на популярності використання певних методів у проєктах з відкритим кодом.

- Покращення автозаповнення:

IntelliCode пропонує найбільш релевантні варіанти автозаповнення на основі аналізу коду. Це прискорює процес написання коду та знижує кількість помилок.

- Підтримка кількох мов програмування:

IntelliCode підтримує C#, C++, Python, TypeScript/JavaScript та інші мови програмування .

2) Code Analysis and Refactoring

Visual Studio 2022 включає потужні інструменти для аналізу коду та його рефакторингу, що допомагають підтримувати високу якість коду:

- Статичний аналіз коду:

Інструмент автоматично перевіряє код на наявність потенційних помилок, недоліків і відповідності стилю кодування. Це дозволяє виявити проблеми на ранніх стадіях розробки.

- Рефакторинг коду:

Visual Studio пропонує різноманітні можливості для рефакторингу, такі як перейменування змінних, виділення методів, зміна структури класів тощо. Ці інструменти допомагають підтримувати чистоту та читабельність коду, знижуючи ризик появи помилок.

- Live Code Analysis:

Проводиться аналіз коду в реальному часі, надаючи рекомендації безпосередньо під час написання коду. Це забезпечує негайний зворотний зв'язок і допомагає уникати поширених помилок

Підведемо підсумки, сучасні інтегровані середовища розробки значно покращують процес розробки, допомагаючи розробникам писати якісний, чистий та продуктивний код з використанням AI-рекомендацій до рефакторингу та інтегрованого тестування, ці інструменти сприяють ефективній і продуктивній роботі.

Серед розглянутих програм, що є стандартами індустрії, складно обрати кращий, проте можна виділити основні переваги. Якщо ваш проект потребує інтеграції з розробками компанії Microsoft то кращим варіантом для вас може стати Microsoft Visual Studio. З іншого боку, якщо ваш проект вимагає швидкого написання коду, кращої інтелектуальної підтримки та вищої інтегрованості з рушіями Unity та Unreal Engine то вам варто розглянути JetBrains Rider.

2.2 Рушії розробки мобільних додатків

Android Studio

Android Studio - це офіційне інтегроване середовище розробки (IDE) для розробки додатків виключно для систем Android. Інструмент представляє собою самостійну машину за допомогою якої можна створювати додатки для операційної системи Android. Заснована на потужному редакторі коду та інструментах розробника від IntelliJ IDEA завдяки чому не потребує встановлення додаткових IDE. Android Studio пропонує ще більше можливостей, які підвищують вашу продуктивність при створенні додатків для Android, таких як:

- Гнучка система збірки на основі Gradle

Gradle - це інструмент автоматизації збірки, відомий своєю гнучкістю у створенні програмного забезпечення. Інструмент автоматизації збірки використовується для автоматизації створення додатків. Процес збірки включає компіляцію, компонування та пакування коду. За допомогою інструментів автоматизації збірки процес стає більш послідовним. Цей інструмент дає змогу зібрати сирий проект у APK(Android Package Kit), формат файлу, який Android використовує для розповсюдження та встановлення програм.

- Швидкий та багатофункціональний емулятор

Інструмент, що дозволяє тестувати додатки без необхідності збирати проект, відправляти APK на мобільний пристрій і встановлювати на нього застосунок. Замість цього, Android Studio пропонує скористатися емулятором завдяки чому, тестування проходитиме прямо у розробника на комп’ютері, що на порядок пришвидшить процес розробки додатку.

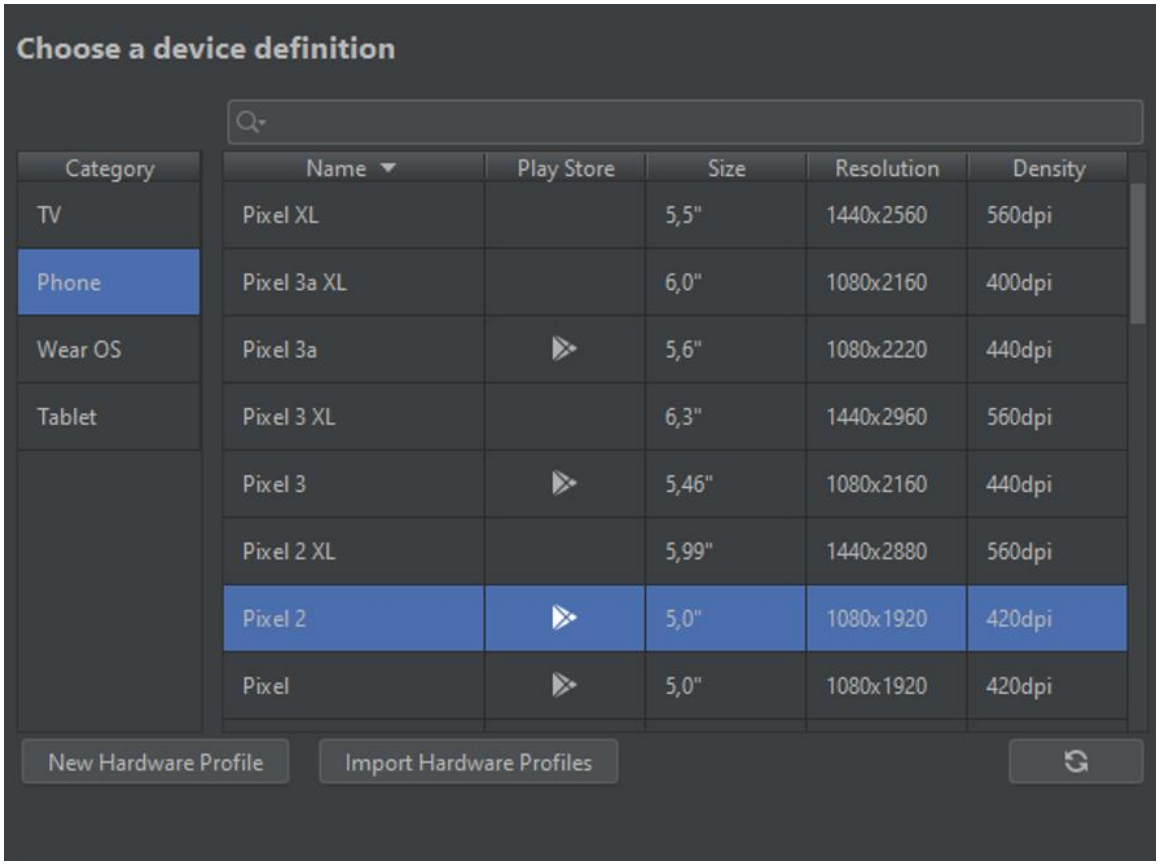

Рис.2.5 Меню вибору пристрою для емуляції роботи додатку

Уніфіковане середовище, в якому ви можете розробляти для всіх пристроїв Android

Інтегроване середовище розробки Android Studio дає можливість розробляти додатки для будь якої версії операційної системи Android.

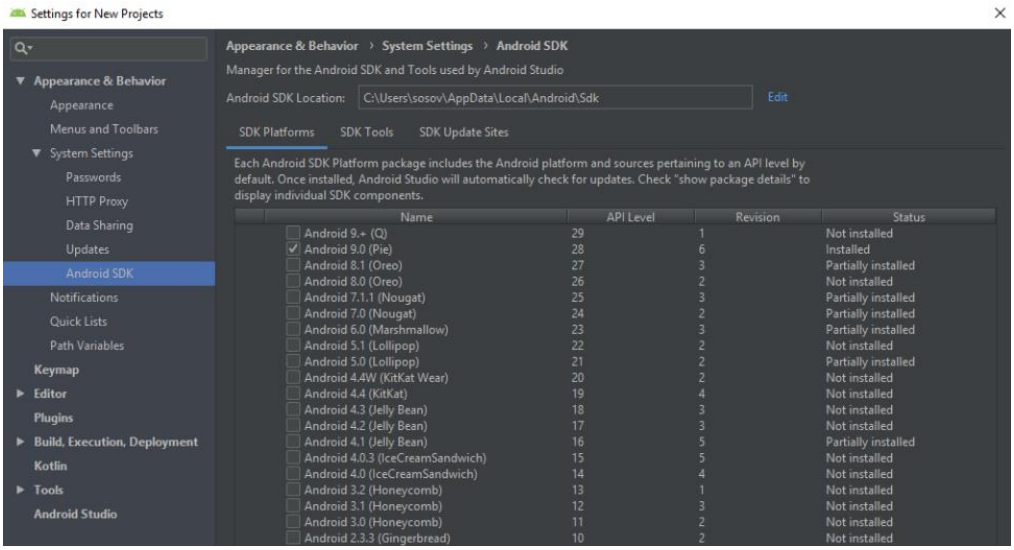

Рис. 2.6 Меню вибору версії операційної системи Android

Редагування в реальному часі для оновлення компонентів в емуляторах та фізичних пристроях в режимі реального часу

Популярний спосіб розробки, який повсюди застосовується для розробки інтерфейсу додатку. Android Studio надає можливість вносити зміни у код додатку і одразу бачити зміни у окремому вікні додатку, ця функція має назву “Live Edit”.

Вбудована підтримка Google Cloud Platform та інтегрованість з сервісами Google.

Структура проекту Android Studio

Кожен проект в Android Studio містить один або декілька модулів з файлами вихідного коду та файлами ресурсів.

За замовчуванням Android Studio відображає файли вашого проекту у поданні проекту Android, як показано на рисунку 2.7.

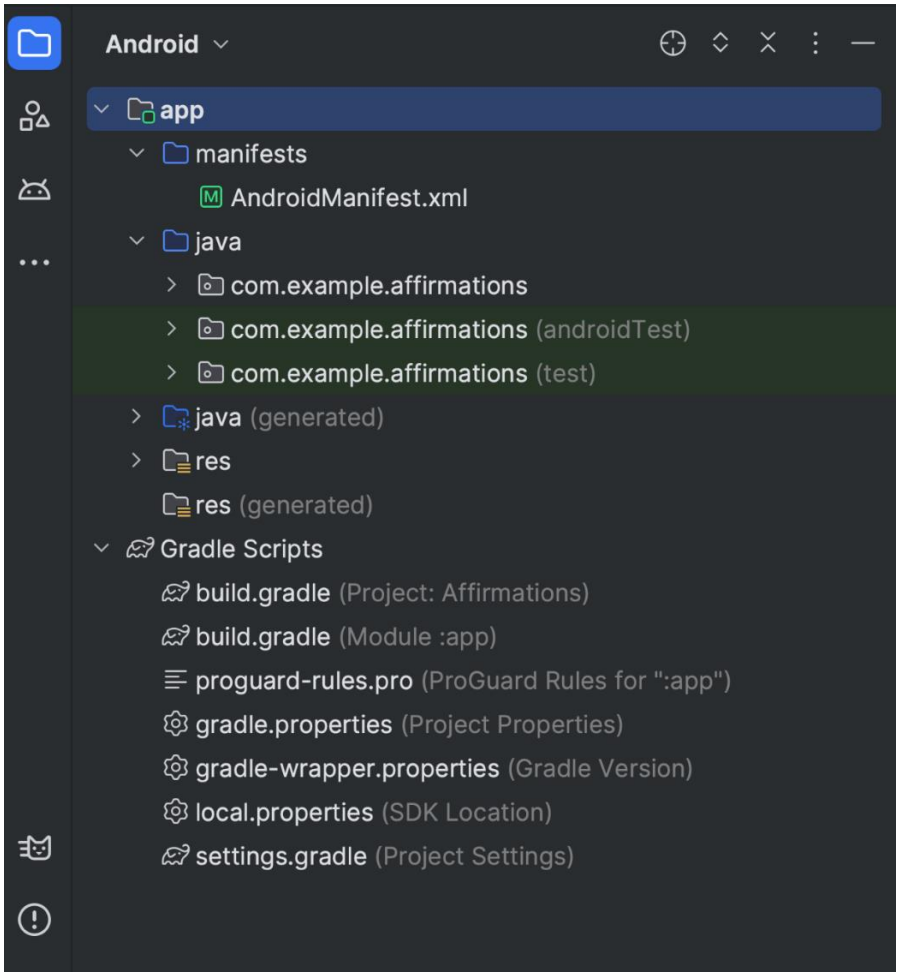

Рис. 2.7 Провідник проекту Android Studio

Це подання організовано за модулями, щоб забезпечити швидкий доступ до ключових вихідних файлів вашого проекту. Всі файли збірки видно на верхньому рівні, у розділі Gradle Scripts. Кожен модуль програми містить наступні папки:

manifests: Містить файл AndroidManifest.xml.

java: Містить файли вихідного коду Kotlin та Java, включаючи тестовий код JUnit.

res: Містить усі некодові ресурси, такі як рядки інтерфейсу користувача та растрові зображення.

Інструменти для налагодження та профілювання

Інструменти для налагодження та профілювання Android Studio допомагає налагоджувати та покращувати продуктивність коду, включаючи інструменти для вбудованого налагодження та аналізу продуктивності.

Налагодження у рядку

Рядкове налагодження використовується, щоб покращити процес написання коду у вікні редактору за допомогою вбудованої перевірки посилань, виразів і значень змінних.

Налагодження у рядку включає у себе:

- Вбудовані значення змінних
- Об'єкти, що посилаються на вибраний об'єкт
- Значення, що повертаються методами
- Лямбда та операторні вирази
- Значення підказок

Профілі продуктивності Android Studio надає профілі продуктивності, щоб ви могли легко відстежувати використання пам'яті та процесора вашого додатку, знаходити де розподілені об'єкти, виявляти витоки пам'яті, оптимізувати графічну продуктивність та аналізувати мережеві запити.

Переваги Android Studio:

- Широкий функціонал та можливості: Android Studio має великий набір інструментів та функцій, що полегшують розробку мобільних додатків, включаючи

інструменти для розробки інтерфейсу користувача, програмування, тестування та налагодження.

- Підтримка Kotlin: Android Studio повністю підтримує мову програмування Kotlin, що дозволяє розробникам використовувати її для створення мобільних додатків нарівні з Java. Kotlin використовує зручний синтаксис та короткі конструкції, що дозволяє швидше писати код і прискорює процес розробки мобільних додатків. Kotlin повністю сумісний з Java, тому розробники можуть поступово впроваджувати Kotlin у існуючі проекти на Java.
- Інтеграція з Google сервісами: Android Studio має вбудовану підтримку для інших інструментів Google, таких як Firebase, що полегшує розробку додатків, що використовують ці сервіси.
- Багатофункціональність: Android Studio має багатофункціональне середовище розробки з великою кількістю інструментів для розробки, налагодження та тестування мобільних додатків.

Недоліки Android Studio:

- Вимоги до системи: вимагає потужного комп'ютера з великою кількістю оперативної пам'яті, щоб працювати ефективно.
- Складність в освоєнні: може бути складною для освоєння, особливо для тих, хто тільки починає з Android розробкою.
- Споживання ресурсів: Android Studio використовує значну кількість системних ресурсів, що може призвести до повільної роботи системи.
- Обмеженість платформ: програма дозволяє розробляти додатки лише для операційної системи Android тому, якщо ваша ціль досягти якомога більшої кросплатформеності слід розглянути інші 3D рушії.

Unity

Unity - це кросплатформенний рушій для розробки ігор та інтерактивних додатків. Він був створений компанією Unity Technologies і вперше випущений у 2005 році. Unity надає можливість для розробки ігор та інтерактивних додатків на різних платформах, таких як ПК, мобільні пристрої, консолі, віртуальна та доповнена реальність.

Основними особливостями Unity є потужний графічний та фізичний рушій, вбудована система анімації та штучного інтелекту, підтримує мову програмування C#, а також можливість легкої інтеграції з іншими додатками та сервісами.

Unity використовується в різних галузях індустрії, де потрібна розробка ігор та інтерактивних додатків. Основні сфери використання включають:

- Відеоігрова індустрія: Unity є одним з найпопулярніших і найбільш використовуваних рушіїв для розробки відеоігор. Він дозволяє розробникам створювати ігри різних жанрів та рівнів складності для різних платформ.
- Освіта: Unity використовується в освітніх закладах для навчання студентів розробці ігор та інтерактивних додатків. Він надає можливість створювати навчальні програми та тренажери для різних предметів та областей знань.
- Медицина: Unity використовується в медичних додатках для створення симуляцій, тренажерів та інтерактивних програм для навчання медичним процедурам та діагностиці.
- Архітектура та дизайн: Unity використовується для візуалізації архітектурних проектів, створення віртуальних турів по будівлях та моделювання інтер'єрів.

Рушій Unity має модульну архітектуру, що дозволяє розробникам легко розширювати його функціональність за допомогою різноманітних плагінів та розширень та створення власних копомнентів. Це робить Unity потужним і універсальним інструментом для розробки ігор та інтерактивних додатків.

Компонент - це базова одиниця побудови об'єктів у Unity. Компоненти використовуються для надання об'єктам різних властивостей та функціональності. Наприклад, компоненти "Mesh Filter" та “Mesh Renderer” визначають зовнішній

вигляд об'єкта, компонент "Rigidbody" відповідає за фізичну взаємодію об'єкта з іншими об'єктами у грі, компонент "Script" дозволяє приєднати скрипт, написаний на мові програмування C#, до об'єкта та визначити його поведінку.

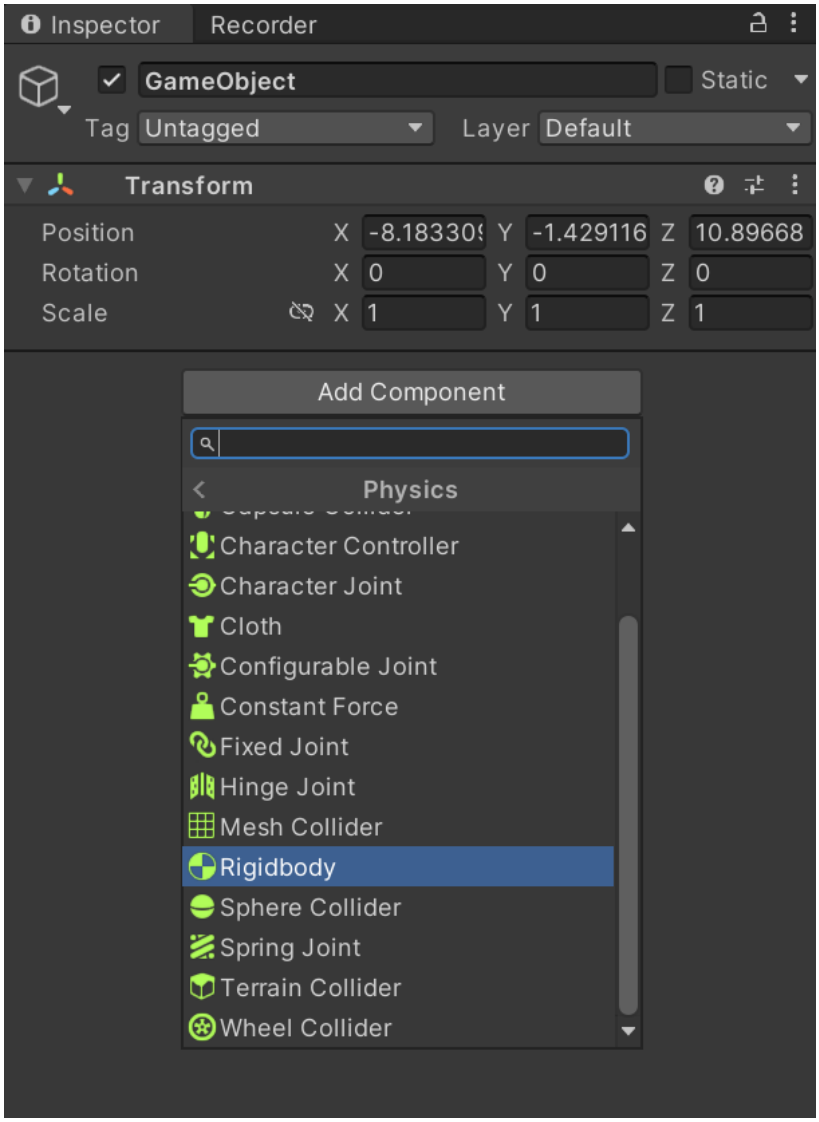

Рис. 2.8 Додавання компонентів фізичної системи у Unity

Unity має підсистеми, що забезпечують роботу компонентів, серед таких:

- Фізична система (Physics System): Відповідає за моделювання фізики об'єктів у грі, включаючи рух, зіткнення та взаємодію. В основі фізичної системи Unity лежить фізичний движок NVIDIA PhysX. Фізична система дозволяє створювати об’єкти з колізією, що дасть змогу об’єктам на сцені фізично взаємодіяти між собою. Для роботи з фізичною системою в Unity існують компоненти Rigidbody та різноманітні колізії (Collider).

- Графічна система (Graphics System): Забезпечує відображення графіки у грі, включаючи текстури, освітлення, тіні та спеціальні ефекти. Unity використовує потужний графічний движок, що підтримує різні ефекти освітлення та анімацію. Unity не обмежується стандартною графікою, вона надає можливість імпортувати різні системи відображення графіки, наприклад Universal Render Pipeline.

Universal Render Pipeline – це оптимальне рішення для графіки у сучасних проектах розроблених на Unity, що дає змогу швидко змінювати графіку без необхідності змінювати код проекту. Також, це дає доступ до ширшого спектру налаштувань графіки.

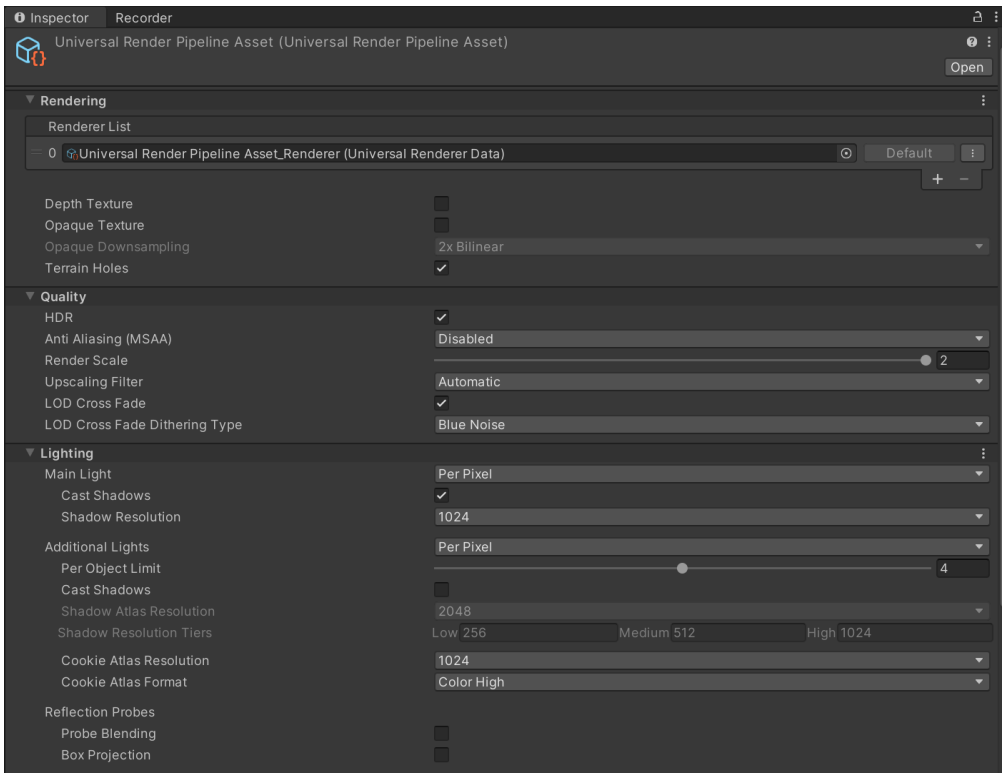

Рис.2.9 Налаштування графіки з Universal Render Pipeline в Unity

- Аудіо система (Audio System): Відповідає за відтворення звуків у грі, включаючи музику, звукові ефекти та діалоги персонажів, а також за можливість зчитувати звуки за допомогою компоненту “Audio Listener”.

- Введення (Input System): Забезпечує обробку введення гравця з клавіатури, миші, геймпада та інших пристроїв. Сьогодні існує дві системи введення стара(Old Input System) та нова(New Input System). Стара система

потребувала додаткової роботи програмістів для реалізації гнучкої системи вводу, щоб отримувати введення від гравця з будь якого пристрою. Нова система, має зручний інтерфейс для налаштування всіх можливих уведень гравця, що зменшує навантаження на програмістів та дає змогу швидше налаштовувати введення від гравця у нових проектах з сучасною системою введення.

- Мережева система (Networking System): Дозволяє реалізувати мережеву гру, включаючи можливість багатокористувацької гри через Інтернет або локальну мережу. Unity п’ятої версії все ще не має власного, зручного інструменту для створення багатокористувацьких або клієнт-серверних додатків, але Unity 6, що вже є у відкритому доступі, має новий інструмент для розробки таких додатків.

- Анімаційна система (Animation System): Дозволяє створювати та керувати анімацією об'єктів у грі, включаючи персонажів, тварин та інші об'єкти. Для цього у Unity є інструмент “Animator” та “Animation”.

Animator – інструмент, що дозволяє збудувати граф анімацій, налаштувати переходи між анімаціями та тригери, шо викликають ці анімації.

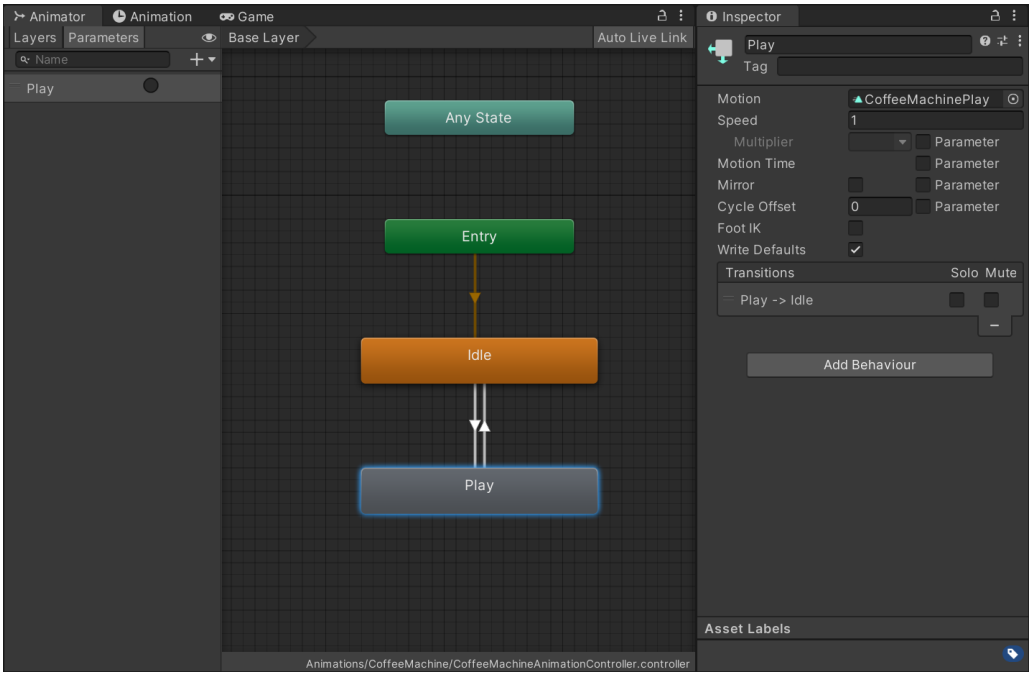

Рис.2.10 Вікно Animator з графом анімації у Unity

Вікно “Animation” містить інструменти для створення, редагування та запису анімації. Основним елементом анімації є ключ, він зберігає дані про зміну об’єкту та позицію на часовому проміжку анімації.

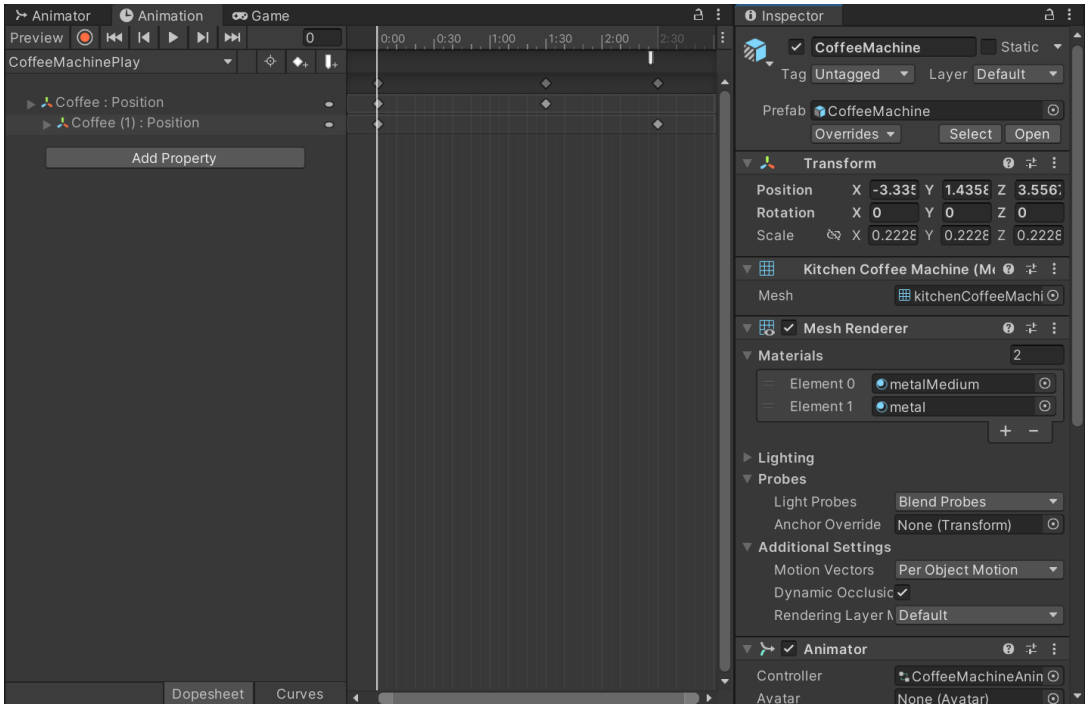

Рис. 2.11 Вікно Animation у Unity

Раніше Unity мав підтримку кількох мов програмування, зокрема C#, JavaScript. Однак, з розвитком платформи, основний акцент було зроблено на C# через його потужність, гнучкість та підтримку спільноти. У 2017 році підтримку JavaScript було офіційно припинено, що залишило C# як основну мову програмування для Unity.

Сьогодні, C# є основною мовою програмування для Unity. Ця мова відома своєю простотою, потужністю та гнучкістю, що робить її ідеальною для розробки ігор та інтерактивних додатків. C# забезпечує розробникам високу продуктивність та підтримує об'єктно-орієнтовану парадигму програмування.

Окремим інструментом для створення додатків в Unity є Visual Script(візуальні скрипти), що дає можливість творцям розробляти механіку ігрового процесу або логіку взаємодії за допомогою візуальної, заснованої на графіках системи, замість того, щоб писати рядки традиційного коду.

Перевагою візуального програмування є доступність для новачків. Завдяки доступності, Visual Script є відмінним інструментом для тих, хто тільки починає працювати з розробкою ігор, або для дизайнерів, які хочуть додавати логіку у свої проекти без залучення програмістів.

Visual Script також може бути інтегрований з написаним кодом на C#, що забезпечує гнучкість та дозволяє використовувати обидва підходи для створення складних ігор. Також це дозволяє ініціативним дизайнерам починати власні проекти без залучення програмістів.

Рушій Unity має безліч можливостей для створення інтерактивних додатків та відео ігор і не тільки. Крім інших можливостей, Unity має:

Unity Asset Store: Це велика онлайн-платформа, де розробники можуть придбати готові ресурси для своїх проектів. Тут можна знайти багато пакетів, які допоможуть в розробці гри. Наприклад, пакети з 2D та 3D моделями персонажів, частинок, оточення та інше.

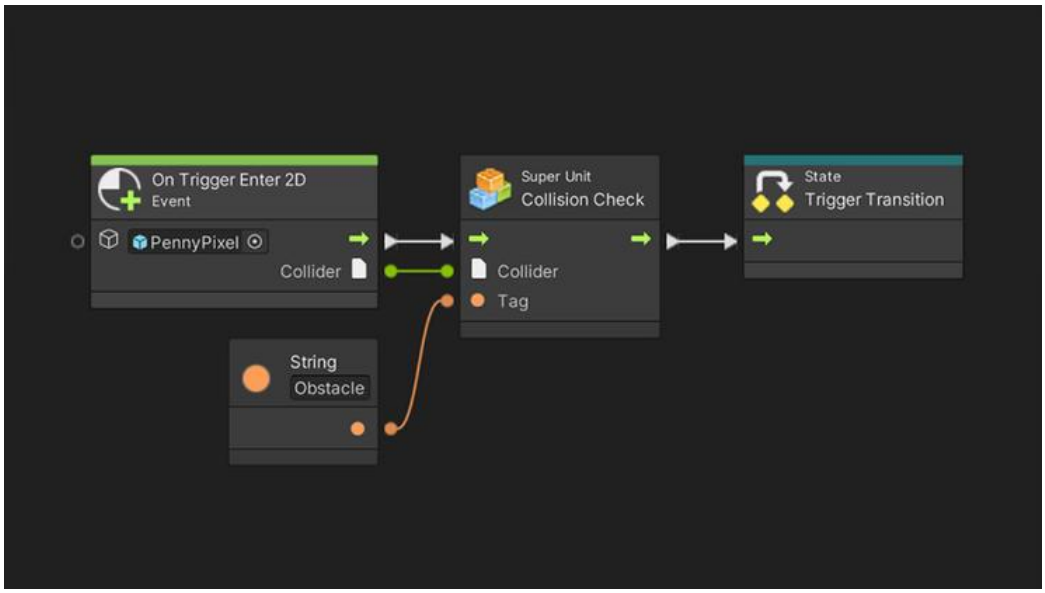

Рис.2.12 Visual Script Unity

Unity Packages: Unity має вбудовану систему пакетів, яка дозволяє додавати функціональність до проекту. Ти можеш використовувати готові пакети або створювати свої власні. Наприклад, пакети для штучного інтелекту, фізики, анімації та інше.

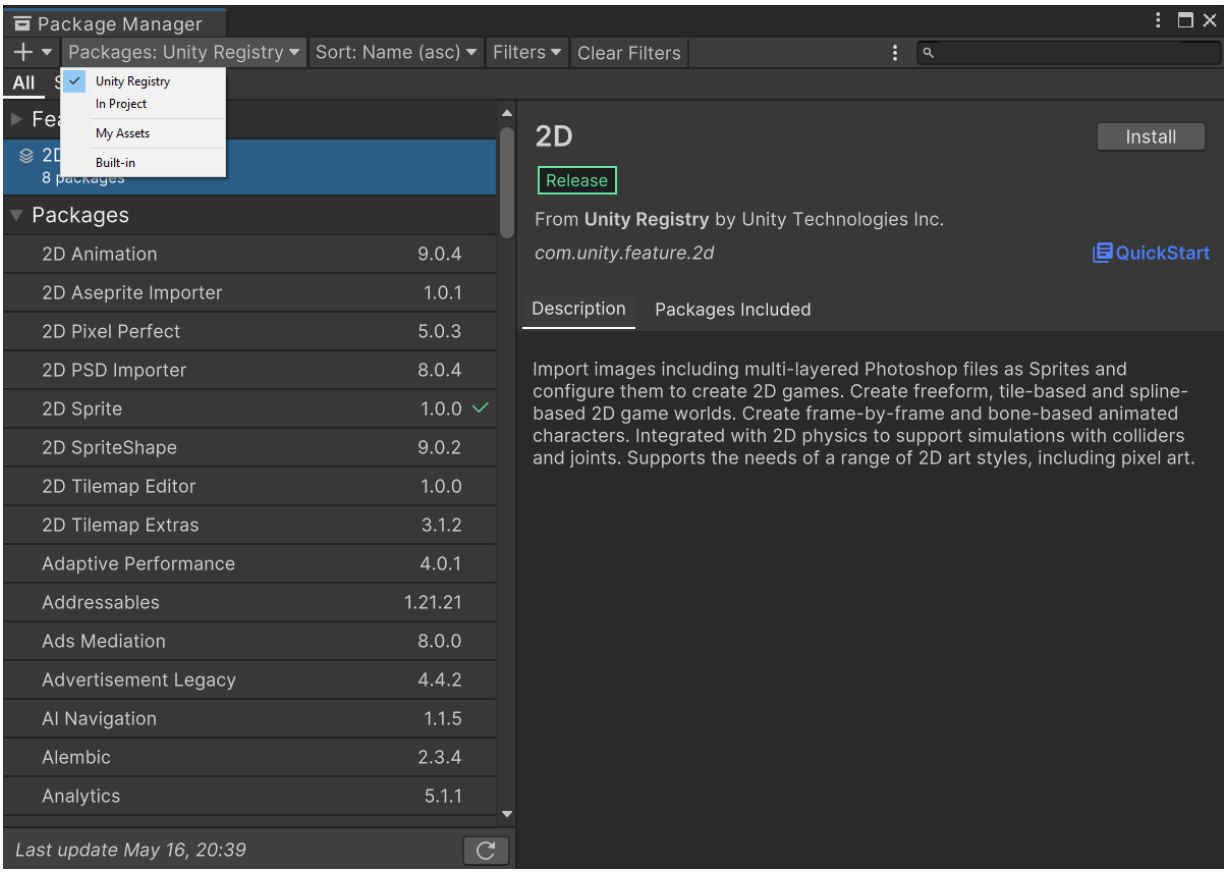

Рис. 2.13 Unity Package Manager

Огляд основних інструментів розробки Unity

Рушій Unity надає розробникам широкий набір інструментів, які дозволяють створювати високоякісні ігри та інтерактивні додатки для різних платформ. Основні інструменти включають:

Unity Editor: Головний інструмент для розробки ігор в Unity, що надає зручний інтерфейс для роботи зі сценами, об'єктами, скриптами та іншими аспектами гри. Unity Editor включає інструменти для редагування сцен, управління анімаціями, роботи з фізикою та багато іншого.

Scene View: Інструмент для візуального редагування сцен. Дозволяє розміщувати об'єкти у тривимірному просторі, налаштовувати освітлення, камери та інші елементи сцени.

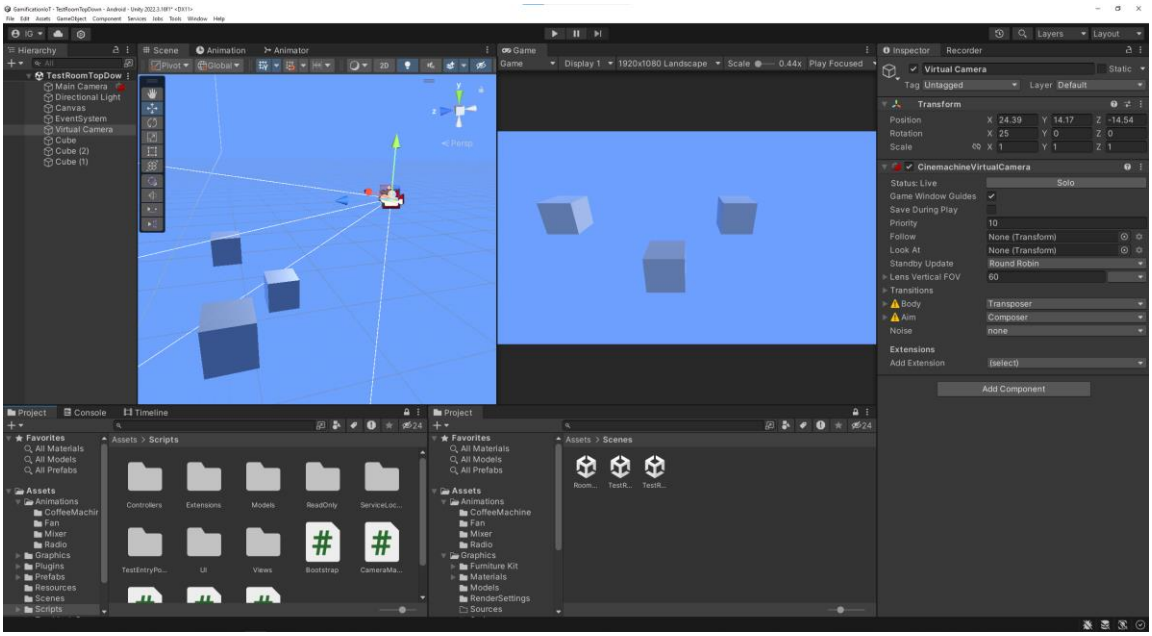

Рис.2.14 Інтерфейс Unity Editor

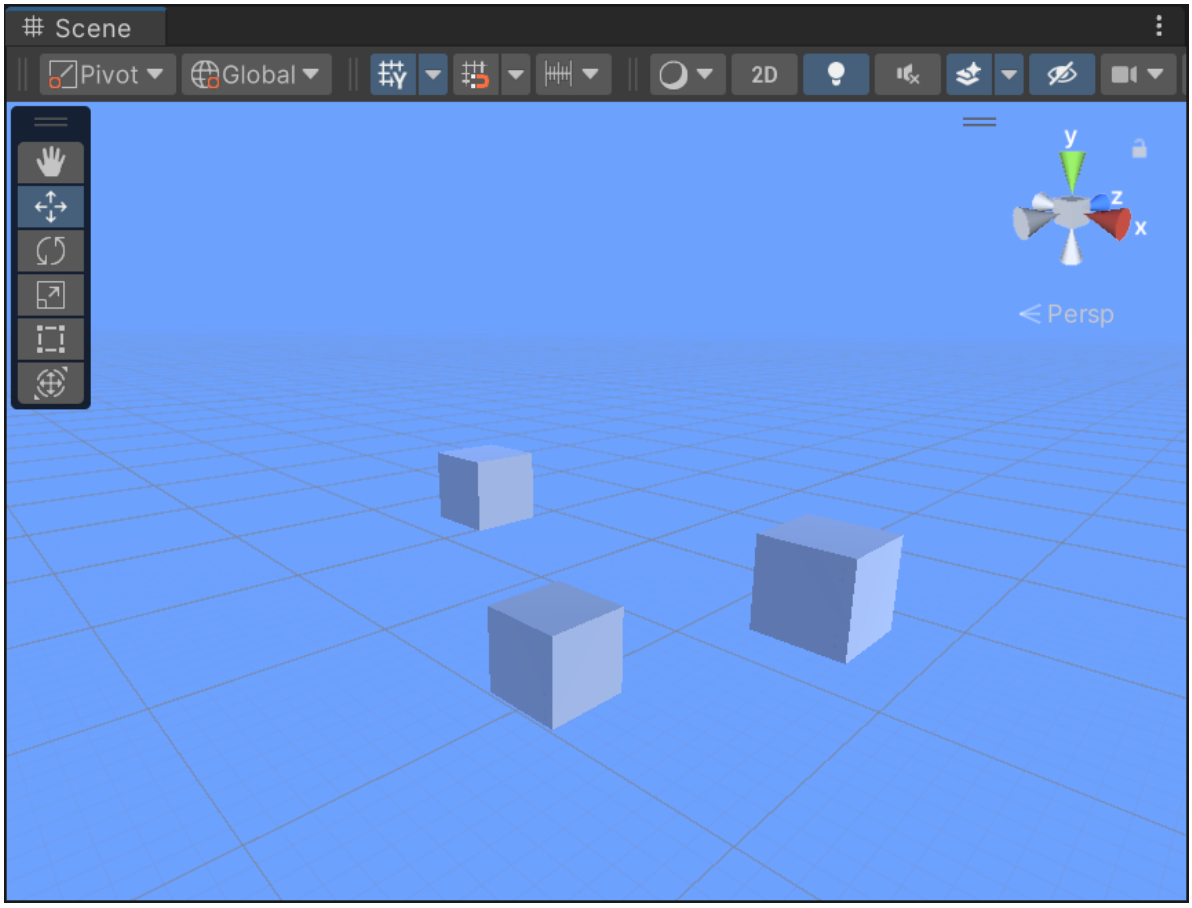

Рис.2.15 Вікно Scene

Game View: Інструмент для перегляду та тестування гри в реальному часі. Дозволяє розробникам бачити, як їх гра буде виглядати на кінцевих пристроях.

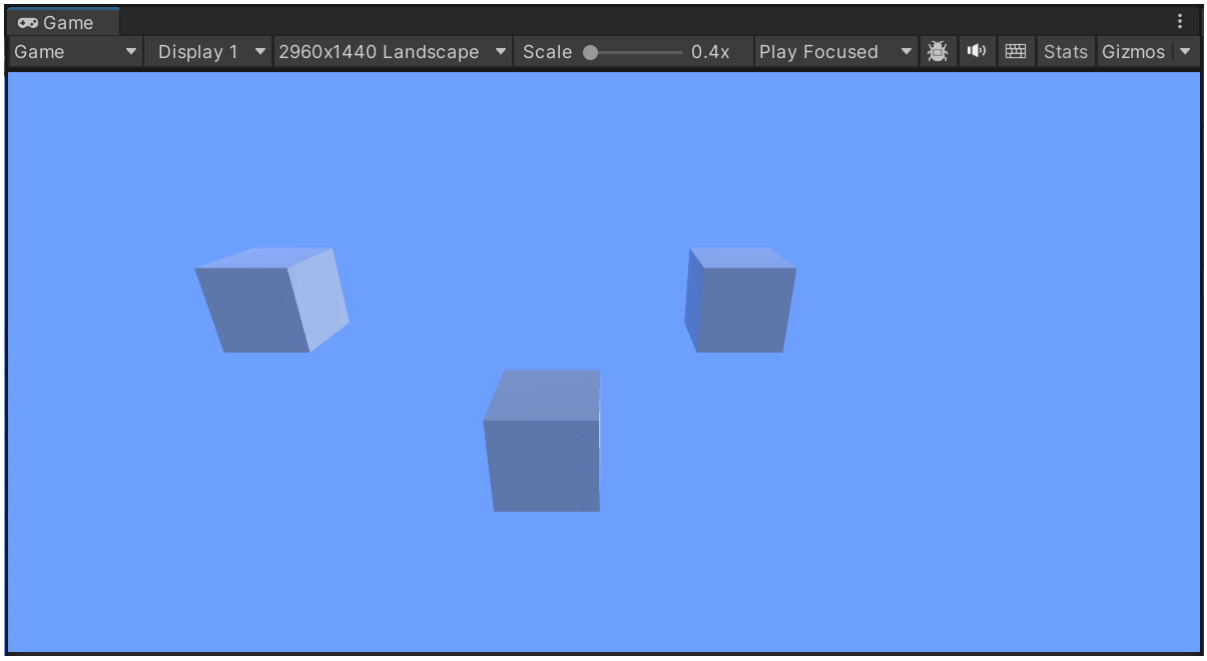

Рис.2.16 Вікно Game

Inspector: Панель для перегляду та редагування властивостей об'єктів. Дозволяє налаштовувати параметри об'єктів, таких як позиція, обертання, масштаб, а також властивості інших компонентів.

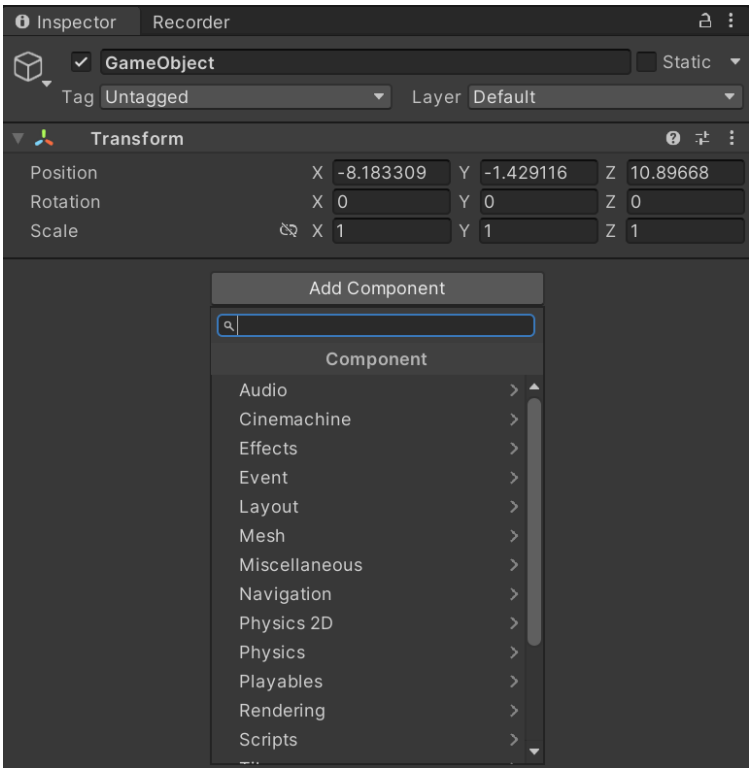

Рис.2.17 Вікно Inspector

Project: Панель для управління файлами проекту. Відображає всі ресурси проекту, включаючи скрипти, моделі, текстури, звуки та інші файли.

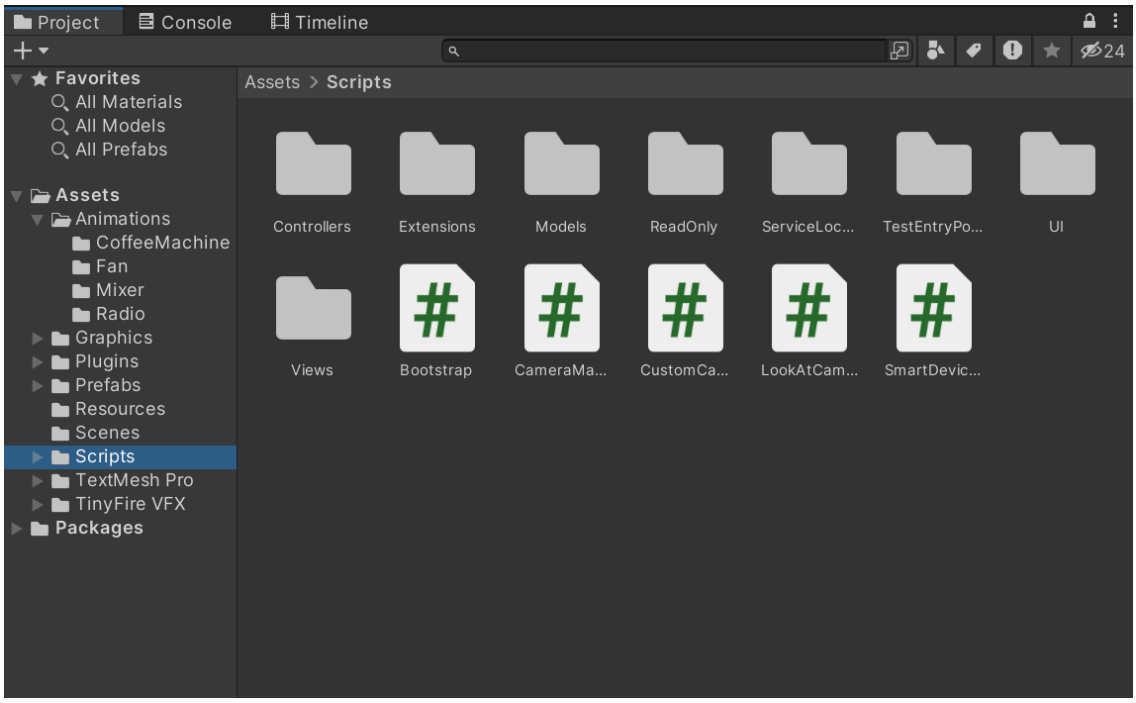

Рис.2.18 Вікно Project(аналог провідника в Unity)

Console: Панель для виводу повідомлень, попереджень та помилок. Використовується для налагодження коду та відстеження проблем у грі.

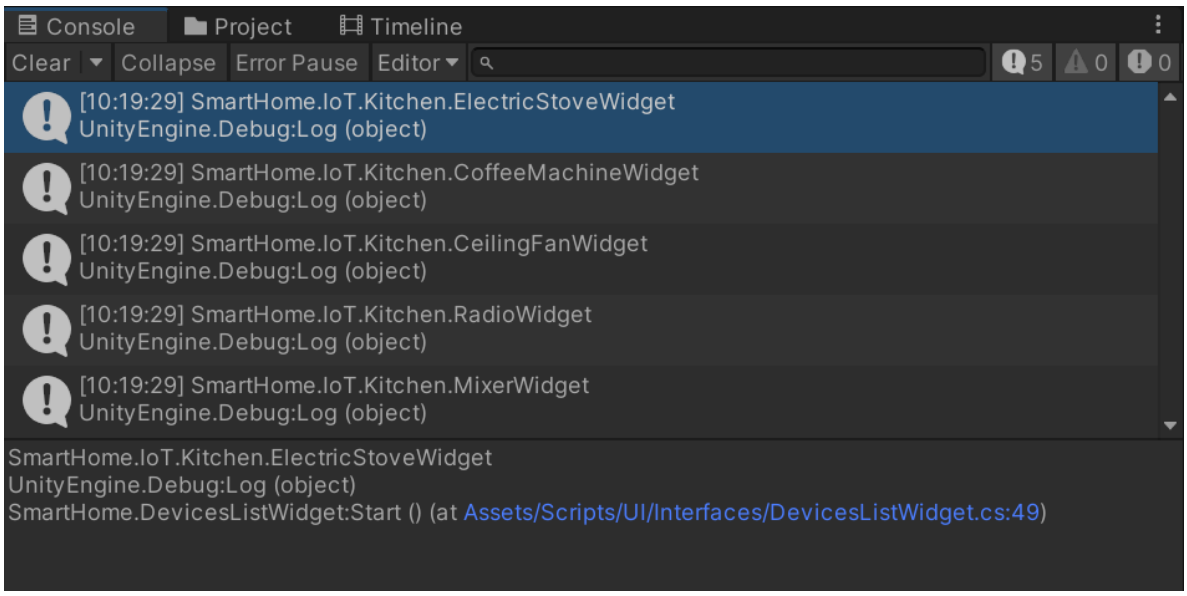

Рис.2.19 Вікно Console

Переваги використання рушія Unity:

1. Крос-платформність:

Unity підтримує розробку для численних платформ, включаючи Windows, macOS, Linux, iOS, Android, консолі (PS4, Xbox One, Nintendo Switch) та VR/AR пристрої.

2. Інтуїтивний інтерфейс:

Unity Editor має зручний та інтуїтивно зрозумілий інтерфейс, який полегшує процес розробки навіть для новачків.

3. Потужні інструменти та компоненти:

Unity надає широкий набір інструментів для роботи з графікою, анімацією, фізикою, звуком та іншими аспектами розробки ігор.

4. Активна спільнота та підтримка:

Велика і активна спільнота розробників, численні форуми, блоги, відеоуроки та документація допомагають швидко вирішувати проблеми та отримувати нові знання.

5. Asset Store:

Unity Asset Store пропонує безліч ресурсів, таких як моделі, текстури, звуки, скрипти та інші інструменти, що прискорюють процес розробки.

6. Підтримка C#:

Використання C# як основної мови програмування забезпечує потужні можливості для розробки та об'єктно-орієнтованого програмування.

Недоліки рушія Unity

1. Високі вимоги до апаратного забезпечення:

Для комфортної роботи з великими проектами Unity потребує потужного , сучасного апаратного забезпечення, особливо для великих проектів.

2. Великий розмір кінцевих пакетів встановлення:

Готові проекти ігор можуть мати великий розмір через необхідність включення всіх необхідних бібліотек та ресурсів, високу якість графіки та велику кількість об’єктів.

3. Обмеження продуктивності для мобільних платформ:

На мобільних пристроях можуть виникати проблеми з продуктивністю через обмежені ресурси. Тому доведеться оптимізувати додатки, оптимізовувати 3D-моделі та налаштовувати якість графіки відповідно до можливостей пристрою.

4. Вартість ліцензії:

Повна версія Unity (Unity Pro) є досить дорогою, що може бути проблемою для інді-розробників. Щоб відтермінувати сплату за повну версію можна скористатися безкоштовною версією, Unity Personal, для початкових етапів розробки.

Інструменти прототипування інтерфейсу користувача

Figma є потужним інструментом для дизайну інтерфейсів та спільної роботи, який широко використовується в різних галузях, включаючи розробку ігор. Його популярність пояснюється інтуїтивно зрозумілим інтерфейсом, багатофункціональними можливостями та підтримкою роботи в реальному часі.

Основні можливості Figma:

- Спільна робота в реальному часі: Figma дозволяє кільком користувачам одночасно працювати над одним проектом, що значно полегшує командну роботу. Це особливо корисно для великих команд, де дизайнери, розробники та інші учасники можуть бачити зміни в реальному часі та залишати коментарі.
- Хмарне зберігання: Усі проекти зберігаються в хмарі, що забезпечує доступ до них з будь-якого пристрою та запобігає втраті даних.
- Прототипування: Figma дозволяє створювати інтерактивні прототипи, що дає змогу тестувати інтерфейс без необхідності написання коду.
- Плагіни та інтеграції: Інструмент підтримує безліч плагінів та інтеграцій з іншими сервісами, що розширює його функціональність і дозволяє інтегрувати його в існуючі робочі процеси.

Figma часто застосовують в контексті розробки ігор тому, що це інструмент є дуже зручним та максимально простим, має великий набір стандартних функцій, що у подальшому можна збільшити за допомогою розширень.

Одним з ключових аспектів гри є її інтерфейс. Figma дозволяє створювати високоякісні макети UI, які можна легко адаптувати до різних розмірів екранів.

Наприклад, дизайнери можуть створювати меню, панелі інструментів, екрани завантаження та інші елементи інтерфейсу, а потім експортувати їх для використання в грі.

У процесі розробки гри Figma служить мостом між дизайнерами та розробниками. Дизайнери можуть створювати макети і передавати їх розробникам, які інтегрують ці елементи в ігровий движок, такий як Unity. Використання Figma зменшує кількість помилок і забезпечує більш гладкий робочий процес завдяки можливості швидкої перевірки та зворотного зв'язку.

Figma дозволяє створювати інтерактивні прототипи, що дає змогу розробникам ігрового процесу тестувати UI/UX без необхідності написання коду. Це особливо корисно на ранніх стадіях розробки, коли важливо швидко оцінити ідеї та отримати зворотний зв'язок від тестувальників.

Існують плагіни та інструменти, які дозволяють безпосередньо інтегрувати макети з Figma в Unity. Наприклад, розробники можуть використовувати плагіни, які автоматично конвертують дизайн з Figma у формат, сумісний з Unity, що значно скорочує час на імпорт графічних елементів та зменшує ймовірність помилок при їхній інтеграції.

Figma є незамінним інструментом для дизайну та спільної роботи, особливо в контексті розробки ігор. Його можливості для створення UI, прототипування, та інтеграції з ігровими движками, такими як Unity, роблять його важливим елементом сучасного робочого процесу розробки. Використання Figma допомагає командам працювати більш ефективно, зменшувати кількість помилок та швидше втілювати ідеї в життя.

3 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ СИСТЕМИ УПРАВЛІННЯ РОЗУМНОГО БУДИНКУ З ВИКОРИСТАННЯМ ЕЛЕМЕНТІВ ГЕЙМІФІКАЦІЇ

Після огляду засобів для розробки та визначення поняття гейміфікаці, наступним кроком буде підготовка та розробка додатку управління розумного будинку з використанням елементів гейміфікації.

У цьому розділі будуть встановлені функціональні вимоги додатку, проведений відбір інструментів для проекту на основі його вимог та почнеться процес розробки, починаючи архітектурою і закінчуючи кінцевою реалізацією деяких його функцій.

3.1 Аналіз функціональних вимог до мобільного додатку

У сучасному представленні, додатки для керування розумними пристроями мають лише текстовий вигляд та не дають користувачу оглянути окомірно всі пристрої у кімнаті одночасно. Також, проблемою сучасних додатків є те, що вони є застарілими та сприяють залученню користувача до їх використання, тому у додатку який розроблятиметься будуть застосовані елементи гейміфікації.

Для опису функцій буде застосовано UML(Unified Modeling Language) для побудови діаграм класів, потоків та діяльності.

Так як основною ціллю додатку є управління розумним будинком слід почати опис функцій додатку з цього боку. Серед усіх можливих функцій, виділимо найголовніші:

1. Дистанційна взаємодія з пристроями: Користувач повинен мати можливість дистанційно вмикати або вимикати пристрої та мати можливість взаємодіяти з пристроями так, наче користувач стоїть перед ними. Наприклад: користувач бажає приготувати собі кави, він знає, що його чашка вже стоїть у потрібному місці на кавоварці, тоді користувач, має натиснути лише декілька

кнопок у додатку(обрати пристрій, обрати каву та запустити приготування кави). У результаті, пристрій має почати готувати обраний варіант напою.

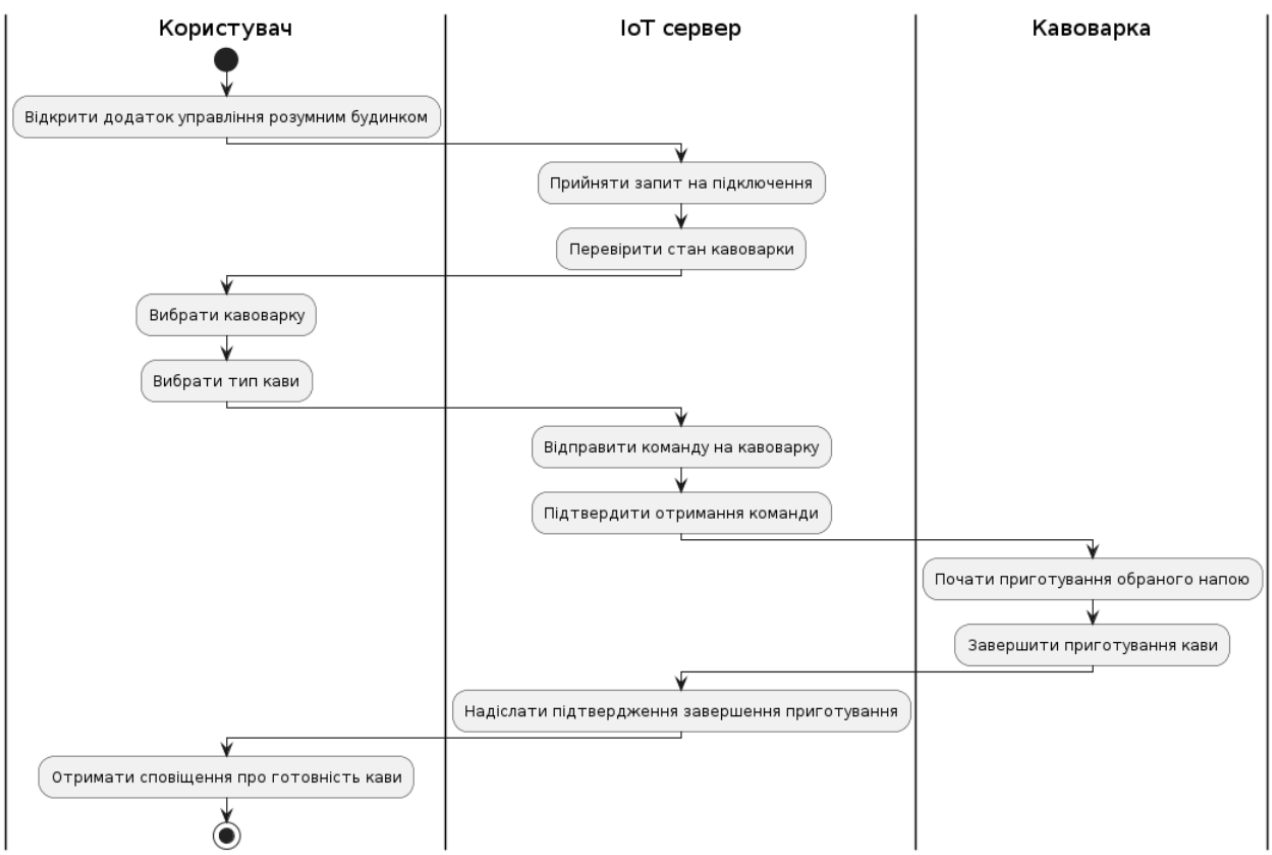

Рис.3.1 Діаграма потоку з описом процесу приготування кави з використання додатку

2. Налаштування сценаріїв: Застосунок повинен дозволяти створювати та керувати сценаріями автоматизації (наприклад, "Ранковий режим", "Вечірній режим" та інші).

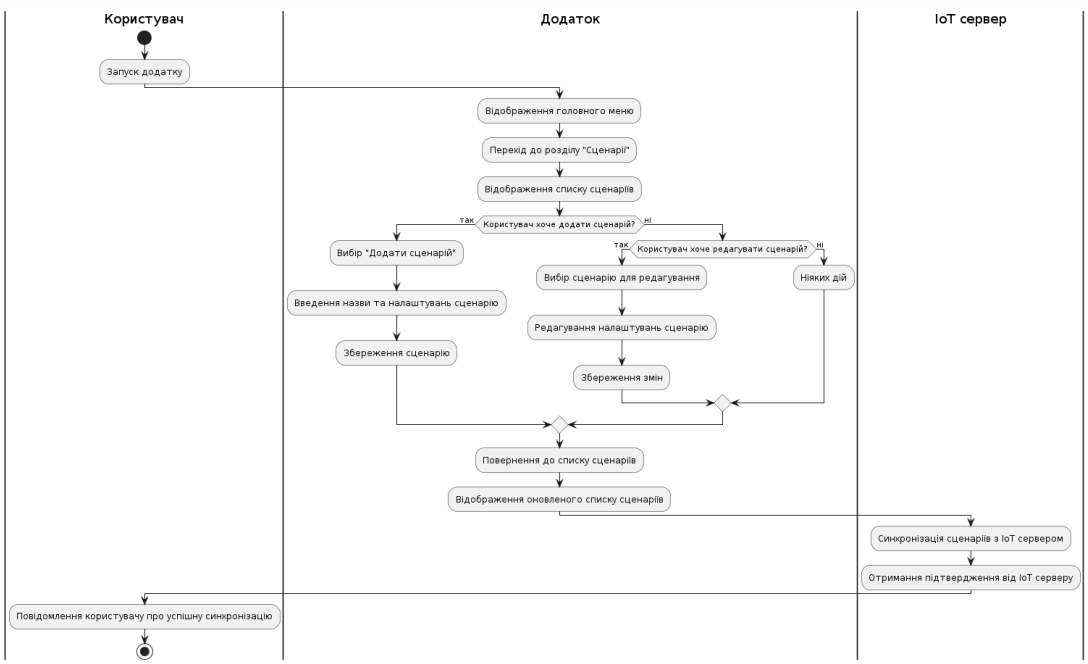

Рис.3.2 Діаграма потоку з описом процесу створення сценарію для роботи пристроїв через додатку

3. Моніторинг стану пристроїв: Відображення поточного стану кожного пристрою (ввімкнений/вимкнений, рівень заряду батареї тощо). Цей процес має відбуватися автоматично, без необхідності чекати поки користувач відкриє опис конкретного пристрою.

Рис.3.3 Діаграма потоку з описом процесу оновлення стану пристроїв та синхронізації з додатком

Особливістю проекту є використання у ньому елементів гейміфікації, серед таких у проекті можна реалізувати:

1. Система нагород і досягнень: Впровадження балів, значків або медалей за певні дії користувача (наприклад, за економію енергії, регулярне використання додатку).

2. Завдання та виклики: Пропозиція користувачам виконувати завдання або приймати виклики (наприклад, "знизити споживання енергії на 10% за тиждень").

3. Таблиця лідерів: Відображення рейтингу користувачів, які досягли найбільших успіхів у використанні додатку.

4. 3D представлення кімнати: Створення та заповнення кімнати 3D моделями для забезпечення можливості окомірного огляду всіх пристрої в будинку та переходу до управління пристроєм за допомогою натискання на 3D модель конкретного пристрою.

Окрім основний функцій, проект має збирати та зберігати використання додатку та пристроїв. В першу чергу для надання розробникам інформації про використання застосунку користувачами, а також для реалізації додаткових функцій, наприклад:

1. Статистика використання: Відображення інформації про використання пристроїв (час роботи, споживання енергії тощо).

2. Звіти про економію енергії: Генерація звітів, що демонструють економію ресурсів завдяки використанню розумного будинку.

3.2 Вибір оптимальних інструментів для конкретного проекту

Середовище розробки застосунку має відповідати певним вимогам, що дадуть змогу реалізувати усі функції додатку. Середовище має відповідати таким вимогам:

1. Кросплатформенність: Платформа на які буде розроблятися проект має надавати можливість створювати застосунок для різних операційних систем (iOS, Android).

2. Інтеграція з розумними пристроями: Мова програмування що застосовуватиметься у середовищі розробки має бути сучасною, мати можливість використання SDK та API від виробників розумних пристроїв для забезпечення сумісності.

3. Інструменти для реалізації гейміфікації: Підтримка гейміфікаційних елементів або, хоча б, надавати можливість створити їх самостійно та включатиме підтримку анімацій.

4. Аналітичні інструменти: Інтеграція з аналітичними сервісами для відстеження прогресу користувачів та ефективності гейміфікаційних елементів.

5. Бази даних та серверні рішення: Використання хмарних платформ для зберігання та обробки даних.

6. Інтерфейс користувача: Підтримка UI/UX та наявність дизайн інструментів для проектування та тестування користувацького інтерфейсу. Прикладом інструменту проектування інтерфейсу користувача може бути, наприклад Figma.

7. Бібліотеки компонентів: Використання бібліотек з готовими компонентами для прискорення розробки та забезпечення єдиного стилю.

8. Тестування та відлагодження: Використання інструментів для автоматизованого тестування функціоналу та безпеки додатку. Інтеграція з системами моніторингу для виявлення та усунення помилок. Наявність інструментів для самостійного тестування у середовищі розробки.

Незважаючи на великий список вимог, серед розглянутих у другому розділі інструментів знайшовся той, що відповідає більшості з них.

Unity є потужною платформою для розробки застосунків, яка відповідає багатьом сучасним вимогам розробників. Давайте розглянемо, як Unity відповідає зазначеним критеріям.

1. Кросплатформенність: Unity підтримує розробку для різних операційних систем, таких як iOS, Android, Windows, macOS та інші. Це дозволяє створювати один код і використовувати його на різних платформах, що суттєво економить час і ресурси.

2. Інтеграція з розумними пристроями: Unity використовує мову програмування C#, яка є сучасною та широко підтримуваною. Unity також має можливість інтеграції з різними SDK та API від виробників розумних пристроїв, забезпечуючи їхню сумісність з вашим додатком.

3. Інструменти для реалізації гейміфікації: Unity надає потужні можливості для створення гейміфікованих елементів, включаючи анімацію та інтерактивні елементи. За допомогою Unity можна легко реалізувати механіки гри, які роблять застосунок більш привабливим для користувачів.

4. Аналітичні інструменти: Unity має інтеграцію з різними аналітичними сервісами, такими як Unity Analytics, Google Analytics та іншими. Це дозволяє відстежувати прогрес користувачів, аналізувати ефективність гейміфікаційних елементів і загальну продуктивність додатку.

5. Бази даних та серверні рішення: Unity може інтегруватися з різними хмарними платформами, що дозволяє зберігати та обробляти дані в хмарі, забезпечуючи масштабованість та надійність.

6. Інтерфейс користувача: Unity підтримує сучасні UI/UX практики та має потужні інструменти для проектування інтерфейсів. Наприклад, ви можете використовувати Unity UI для створення інтуїтивно зрозумілих інтерфейсів.

7. Бібліотеки компонентів: Unity має багатий набір бібліотек і готових компонентів, що прискорює розробку і забезпечує єдиний стиль. Unity Asset Store надає доступ до тисяч готових рішень, які можна використовувати у своєму проекті.

8. Тестування та відлагодження: Unity пропонує широкий спектр інструментів для автоматизованого тестування та відлагодження додатків. Ви можете використовувати Unity Test Framework для написання та запуску тестів, а також інтегрувати системи моніторингу для виявлення та усунення помилок. Крім

того, Unity надає зручне середовище для самостійного тестування додатків під час розробки.

Отже, Unity є потужною та універсальною платформою, яка підходить для розробки застосунків з урахуванням сучасних вимог. Вона надає всі необхідні інструменти та можливості для створення якісних та конкурентоспроможних продуктів.

3.3 Проектування та архітектура мобільного додатку

Важливою складовою розробки додатку є його архітектура. Сучасні додатки мають розроблятися за архітектурними шаблонами щоб зменшити поріг входу для нових розробників. Одним з таких шаблонів є MVC(Model-View-Controller).

MVC (Model-View-Controller) - це архітектурний шаблон, що використовується для розробки програмного забезпечення з розділенням логіки додатка на три пов'язані компоненти: Модель(Model), Вид(View) та Контролер(Controller). У контексті використовуваної середи розробки, це буде означати, що лише View буде наслідуватися від основного класу в Unity – MonoBehaviour.

- Модель (Model): Представляє дані та бізнес-логіку додатку. Модель може бути представлена як база даних або будь-яка інша структура даних, яка зберігає інформацію та відповідає за обробку цих даних.
- Вид (View): Відповідає за відображення даних користувачу та взаємодію з ним. Це може бути інтерфейс користувача, веб-сторінка або будь-який інший спосіб представлення інформації користувачу.
- Контролер (Controller): Керує взаємодією між Моделлю та Видом. Він обробляє вхідні дані від користувача, оновлює модель відповідно до цих даних і відображає відповідний вигляд користувачу.

Перевагами використання MVC є:

1. Розділення логіки додатку на компоненти, що спрощує розробку та підтримку коду.

2. Забезпечує легку змінюваність та розширюваність програмного забезпечення. Додавати нові розумні пристрої у додаток стане простіше.

3. Дозволяє розділити розробку між різними командами або розробниками.

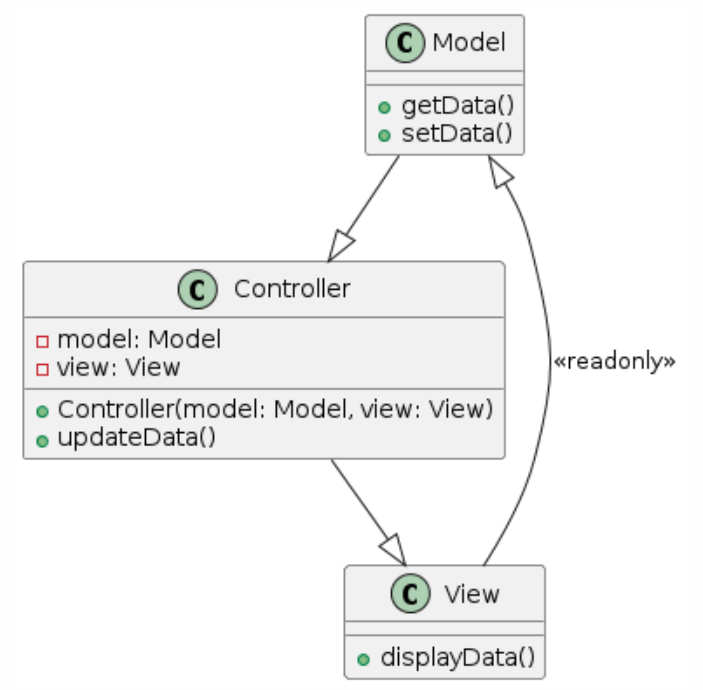

Рис.3.4 MVC у вигляді діаграми класів

Стандартна архітектура буде складною у реалізації, тому було прийнято рішення адаптувати її для роботи з Unity.

Через відсутність у проекті окремого введення від гравця, йому прийшов на заміну клас Widget.

Віджет(Widget – це елемент графічного інтерфейсу користувача, що відображає інформацію і/або надає можливість взаємодіяти з програмою. Реалізацію можна зустріти на будь якому мобільному пристрої, затиснувши на пустому місці головного екрану після чого користувачу відкриється меню з графічними елементами. Серед таких елементів буть:

1. Годинник

2. Погода

3. Музичний програвач та інші

Після невеликої модифікації, MVC архітектура проекту буде виглядати як на рисунку 3.5

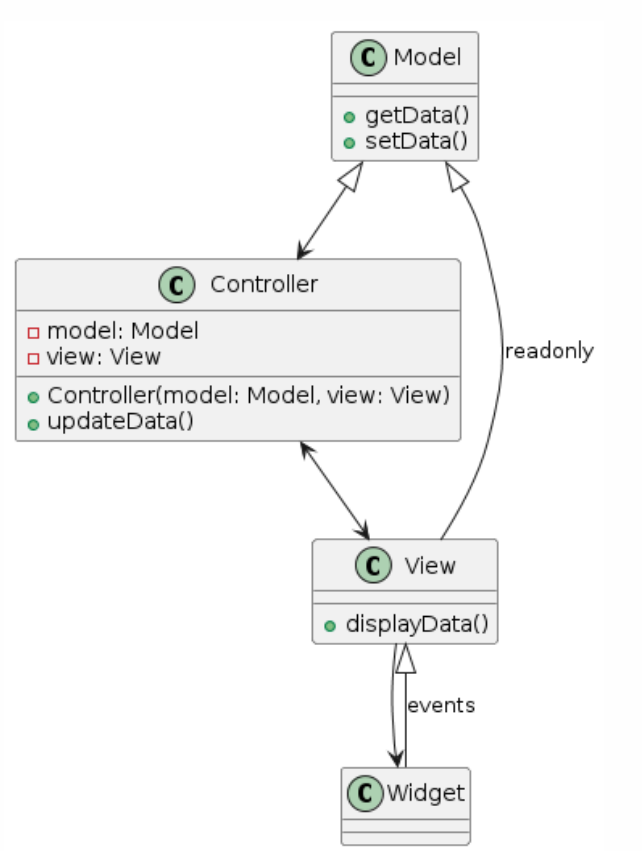

Рис.3.5 Шаблон архітектури розумних пристроїв у додатку

Що стосується проектування інтерфейсу користувача, для цього було використано web-інструмент Figma.

У застосунку були створені прототипи для декількох розумних пристроїв:

1. Вентилятор

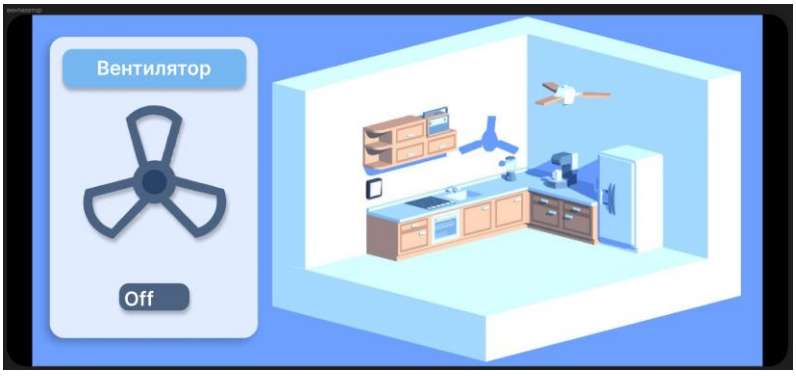

Рис.3.6 Прототип інтерфейсу вентилятора

2. Радіо

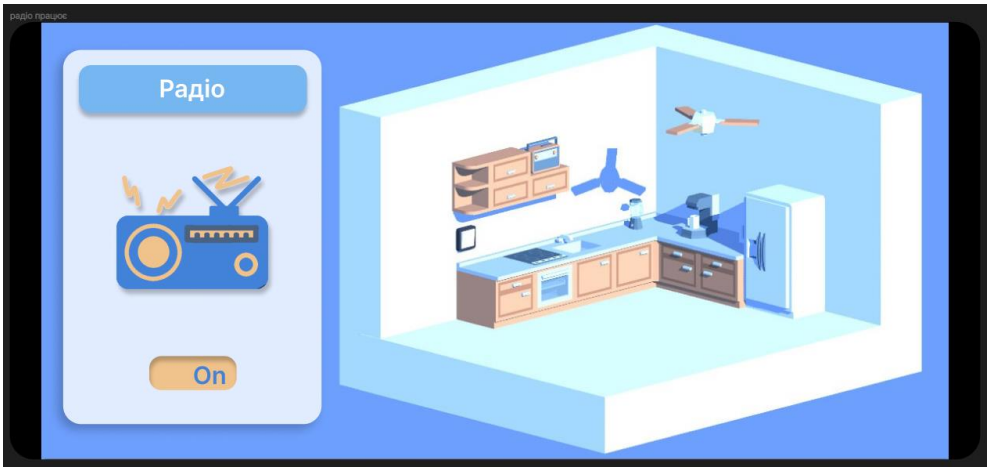

Рис.3.7 Прототип інтерфейсу радіо

3. Міксер

Рис.3.8 Прототип інтерфейсу міксера

4. Кавоварка

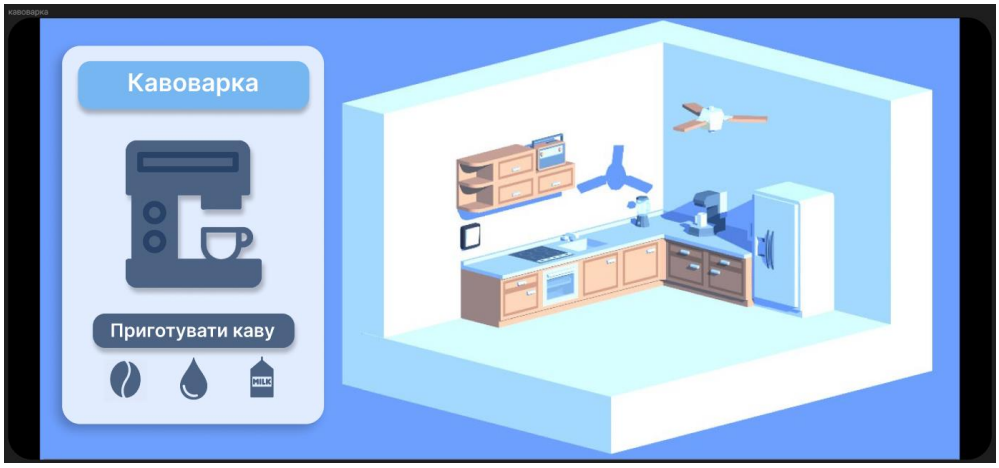

Рис.3.9 Прототип інтерфейсу кавоварки

5. Електроплита

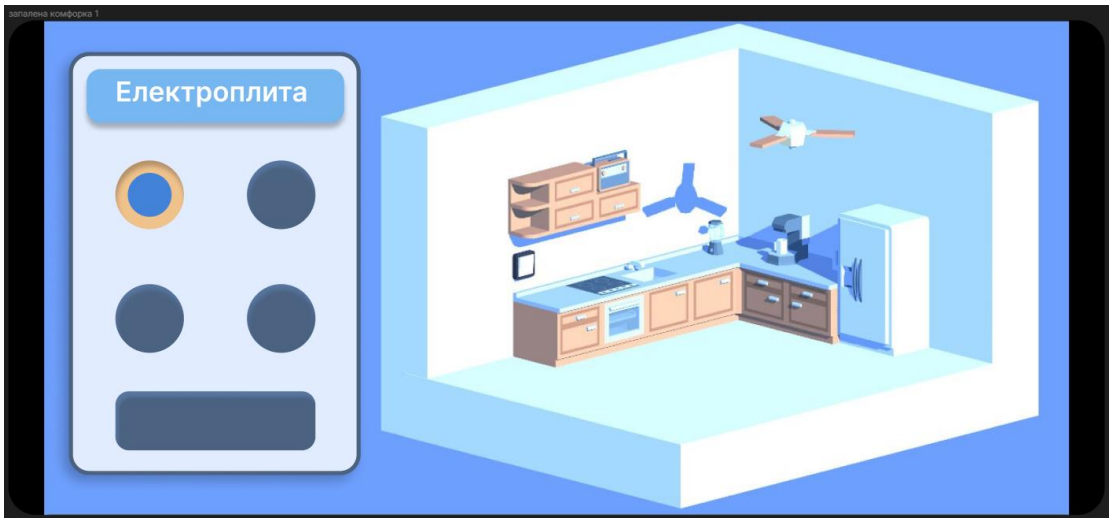

Рис.3.10 Прототип інтерфейсу електроплити

6. Холодильник

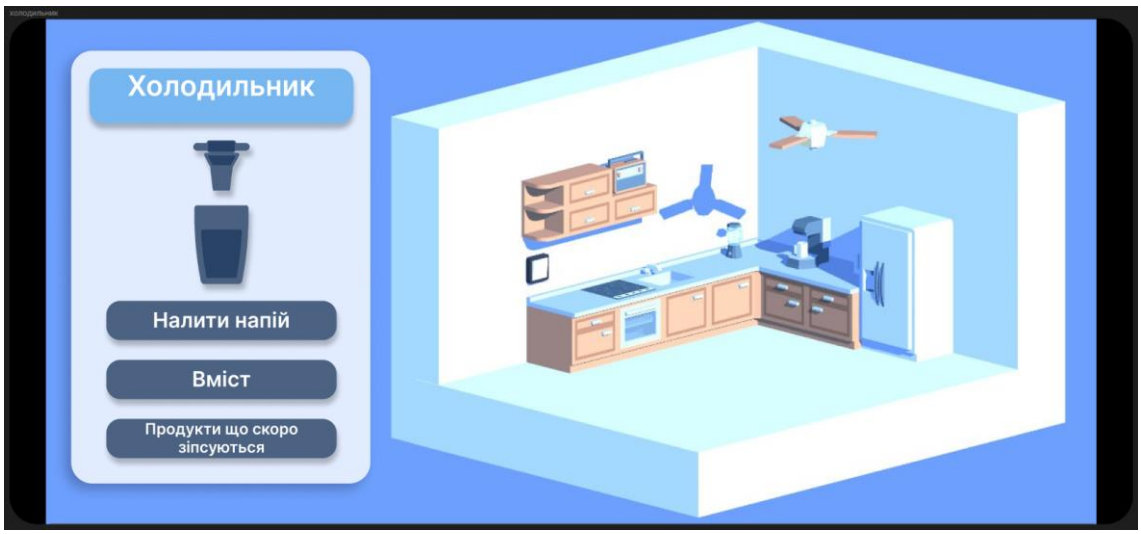

Рис.3.11 Прототип інтерфейсу холодильник

3.4 Реалізація функціональності мобільного додатку

Робота над реалізацією проекту почалася після налаштування проекту Unity.

Для зручної роботи, базовий проект Unity необхідно додатково налаштувати встановивши декілька розширень. Варто звернути увагу на декілька з них.

Cinemachine. Потужний інструмент для управління камерами, інтегрований у Unity. Він значно спрощує процес створення динамічних і кінематографічних

сцен, надаючи розробникам безліч можливостей для налаштування і анімації камер без необхідності написання складного коду.

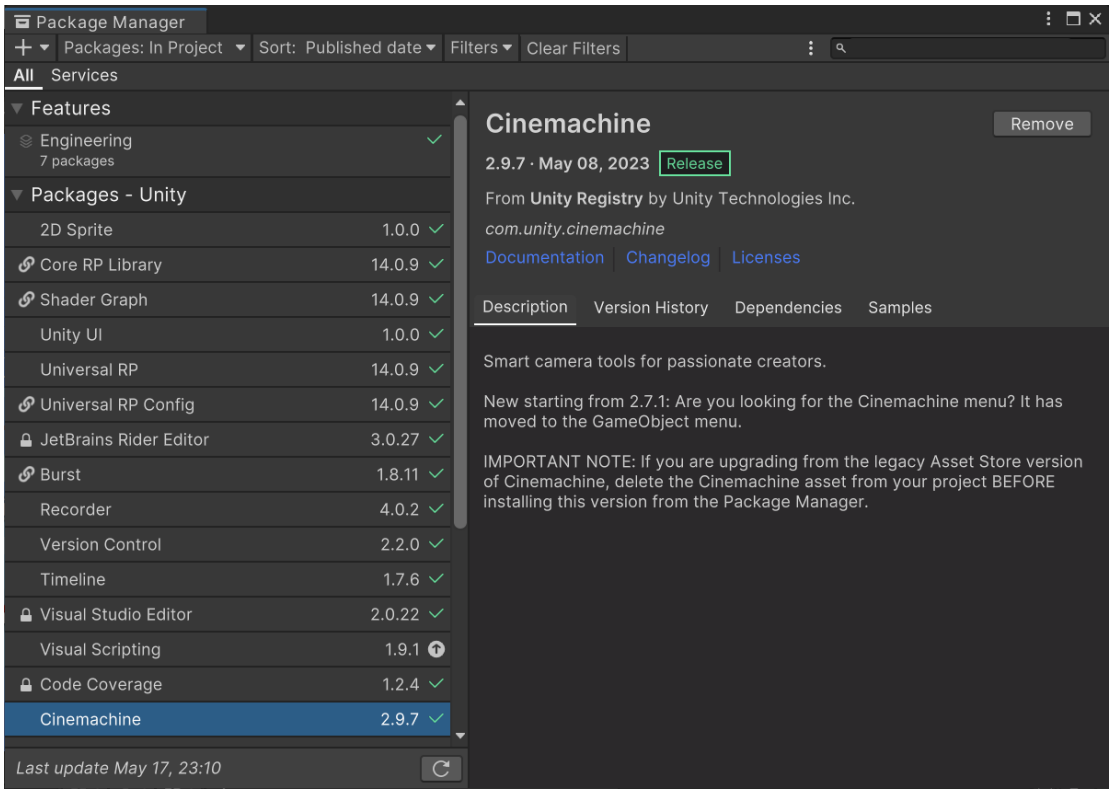

Рис.3.12 Меню Package Manager із встановленими у проекті розширеннями

Universal RP. Один з ключових компонентів екосистеми Unity, який надає розробникам гнучкі та потужні засоби для створення високоякісної графіки в іграх. URP розроблено для забезпечення оптимального балансу між якістю візуалізації та продуктивністю, що робить його ідеальним вибором для широкого спектру платформ, від мобільних пристроїв до консолей і ПК.

Наступним кроком у процесі розробки був процес перенесення прототипу інтерфейсу користувача у проект Unity. Для переносу не було застосовано додаткових модифікацій інтерфейсу, за допомогою ручного засобу було відтворено весь інтерфейс користувача у Unity.

Щоб мати можливість керувати візуальними ефектами користувацького інтерфейсу та у подальшому отримувати введення даних від користувача було створено базовий клас Widget та усі інші нащадки для кожного пристрою та меню окремо.

```
public class Widget : MonoBehaviour
{
    public event Action<bool> VisibilityUpdate;

    public string Name => WidgetName;
    [SerializeField] private string WidgetName = "Not initialized";

    public bool Visible
    {
        get => _visibility;
        set
        {
            _visibility = value;
            VisibilityUpdate?.Invoke(_visibility);
            VisibilityUpdated();
        }
    }

    private bool _visibility;

    protected virtual void VisibilityUpdated()
    {
        gameObject.SetActive(_visibility);
    }
}
```

Рис.3.13 Код базового класу Widget

У класі Widget було реалізовано ключові властивості та методи для будь якого Widget, такою властивістю є перемикання стану графічного елементу. Надається зручна можливість для швидкого увімкнення або вимкнення інтерфейсу. Була створена подія, на яку можна підписати ззовні класу, щоб не створювати оберненої залежності класів. Також, базовий клас графічного елементу зберігає у собі його назву.

Завдяки Cinemachine, у проекті вдалось реалізувати перехід камери від головної до націленої на конкретний пристрій. Цього вдалось досягти завдяки використанню віртуальної камери.

На сцені розміщується лише одна камера(головна). Коли розробник за допомогою написаного функціоналу вимикає головну віртуальну камеру, ту що націлена на показ всієї кімнати, і вмикає камеру що цілеспрямовано, з близької позиції знімає розумний пристрій, головна камера починає плавно рухатися зі своєї початкової позиції у позицію де знаходить віртуальна камера пристрою і намагається повторити всі її параметри.

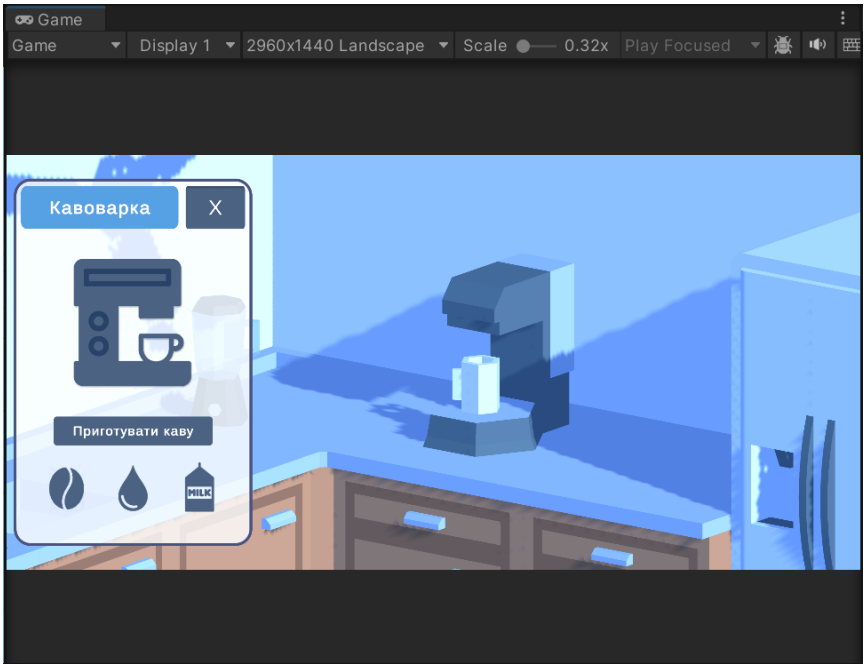

Рис.3.14 Вид на пристрій після переходу камери від положення з оглядом кімнати у положення огляду пристрою

Можливість вмикати і вимикати необхідні камери надає клас CameraManager.

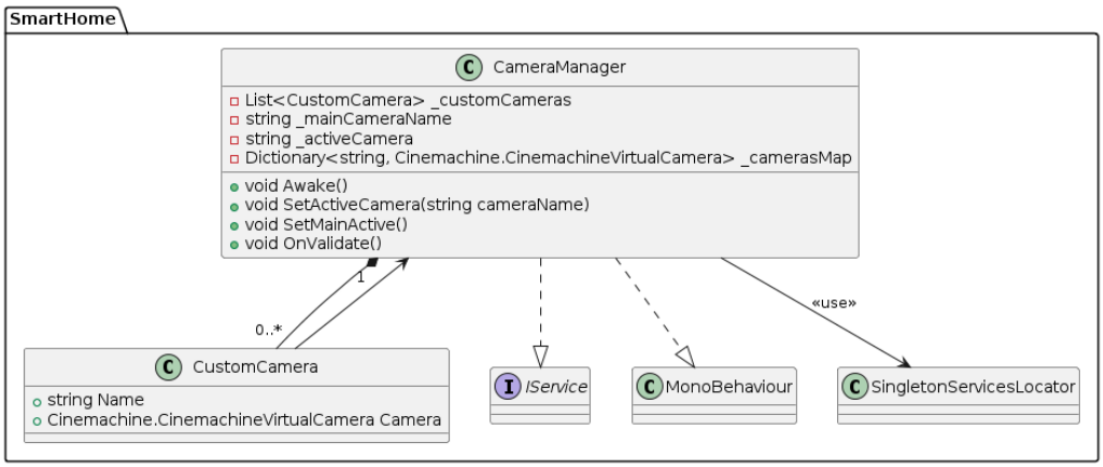

Рис.3.15 Діаграма класів для класу CameraManager

```
public class CameraManager : MonoBehaviour, IService
{
    [SerializeField] private List<CustomCamera> _customCameras;
    [SerializeField] private string _mainCameraName;
    private string _activeCamera;
    private Dictionary<string, CinemachineVirtualCamera> _camerasMap;

    private void Awake()
    {
        _camerasMap = new Dictionary<string, CinemachineVirtualCamera>();
        _activeCamera = _mainCameraName;
        foreach (var customCamera in _customCameras)
        {
            _camerasMap[customCamera.Name] = customCamera.Camera;
        }

        SingletonServicesLocator.Instance.Register(service: this);
    }

    public void SetActiveCamera(string cameraName)
    {
        if(!string.IsNullOrEmpty(_activeCamera))
            _camerasMap[_activeCamera].gameObject.SetActive(false);

        _activeCamera = cameraName;
        _camerasMap[cameraName].gameObject.SetActive(true);
    }

    public void SetMainActive()
    {
        SetActiveCamera(_mainCameraName);
    }

    private void OnValidate()
    {
        _customCameras.Clear();

        _customCameras.AddRange(collection: GetComponentsInChildren<CustomCamera>(includeInactive: true));
    }
}
```

Рис.3.16 Код класу CameraManager

Завдяки тому, що кожен Widget має власну назву, так само як і кожна віртуальна камера для огляду пристрою, їх можна зберігати у зручній структурі даних, що носить назву Dictionary. Словник (Dictionary) зберігає у середині пари ключ та значення. Перевагою використання такої структури даних є швидкість отримання доступу до значення за ключем та порівняно не висока швидкість запису даних у структуру.

Створений клас реалізує, просту функцію, перемикання камер пристроїв. За допомогою того, що клас зберігає назву поточної активної камери, наступна камера гарантовано спрацює за планом, спочатку вимкнеться активна камера, після чого активується нова камера, а її ім’я збережеться у стані об’єкту.

Розглянемо реалізацію одразу декількох пристроїв, таких як: вентилятор, міксер та радіо. Перелічені пристрої мають майже однакову архітектуру, через простоту їх реалізації на цьому етапі проекту, тому розглянемо детально лише один пристрій.

Як вже було визначено, структура розумного пристрою складається з:

- Моделі
- Контролеру
- Виду
- Графічним елементом (Widget)

У зв’язку з цим, реалізація вентилятору складається з декількох класів, розглянемо їх у зазначеному порядку.

FanModel – модель даних розумного вентилятору. Містить у собі лише одне поле логічного типу, що зберігає стан пристрою(ввімкнутий або вимкнутий).

```
[Serializable]
public class FanModel
{
    public bool Enabled;
}
```

Рис.3.17 Код класу FanModel

FanController – контролер пристрою, зберігається у об’єкті, що реалізує шаблон класу “Service Locator”. Контролеру для такої простої моделі даних достатньо реалізовувати лише функції Enable та Disable, що повністю забезпечує роботу класу і пристрою вцілому.

```
public class FanController : IReadOnlyFan
{
    public bool Enabled => _model.Enabled;

    private readonly FanModel _model;

    public FanController(FanModel model)
    {
        _model = model;
    }

    public void Enable()
    {
        _model.Enabled = true;
    }

    public void Disable()
    {
        _model.Enabled = false;
    }
}
```

Рис. 3.18 Код класу FanController

FanView – клас, що є контролером для візуальної частини всього пристрою. Містить у собі поля Animator(відповідає за компонент анімації), FanController, CeilingFanWidget(графічний елемент пристрою). Використовує базові функції Awake та Start для початкового налаштування пристрою у яких підписується на подію у Widget для зчитування введення від гравця(натиск на кнопку), після чого кожне спрацювання події буде викликати метод OnStateChanged класу FanView.

Метод OnStateChange опрацьовує стан пристрою, передає ввід користувача далі, до контролеру, та за допомогою методу Animate оновлює дані для контролера анімацій вентилятору.

```
[RequireComponent(typeof(Animator))]
public class FanView : MonoBehaviour
{
    private CeilingFanWidget _widget;
    private FanController _controller;
    private Animator _animator;

    [SerializeField] private string _spinningAnimation = "Spinning";
    private int _spinningKey;

    private void Awake()
    {
        _animator = GetComponent<Animator>();

        _spinningKey = Animator.StringToHash(_spinningAnimation);
    }

    private void Start()
    {
        var widgetService = SingletonServicesLocator.Instance.Get<WidgetsServiceLocator>();
        _widget = widgetService.Get<CeilingFanWidget>();

        var controllersService = SingletonServicesLocator.Instance.Get<ControllersServiceLocator>();
        _controller = controllersService.Get<FanController>();

        _widget.OnStateChanged += OnStateChanged;
    }

    private void OnStateChanged(bool state)
    {
        if (state)
            _controller.Enable();
        else
            _controller.Disable();

        Animate();
    }

    private void Animate()
    {
        _animator.SetBool(_spinningKey, _controller.Enabled);
    }
}
```

Рис. 3.19 Код класу FanView

Останнім елементом цього ланцюгу є клас CeilingFanWidget. Функція цього класу – забезпечення роботи інтерактивного інтерфейсу користувача, тому серед полів цього класу можна побачити такі класи як:

- Image –пов’язує поле з об’єктом картинки на сцені, всередині інтерфейсу користувача.
- Sprite – зберігає у поле посилання на файл картинки з вікна Project.

- Button – пов’язує поле з об’єктом кнопки на сцені, всередині інтерфейсу користувача.
- GameObject – пов’язує будь який об’єкт на сцені з полем класу.

Також, клас містить перевизначені методи On та Off, а також метод Switch, що виконує роль перемикача, реагуючи на натиск кнопки.

```
public class CeilingFanWidget : SwitchWidget
{
    [SerializeField] private Image _deviceIcon;

    [SerializeField] private Sprite _onSprite;
    [SerializeField] private Sprite _offSprite;

    [SerializeField] private Button _button;

    [SerializeField] private GameObject _onView;
    [SerializeField] private GameObject _offView;

    private void Awake()
    {
        Off();

        _button.onClick.AddListener(Switch);
    }

    protected override void On()
    {
        State = true;

        _deviceIcon.sprite = _onSprite;
        _offView.SetActive(false);
        _onView.SetActive(true);

        InvokeStateChanged();
    }

    protected override void Off()
    {
        State = false;

        _deviceIcon.sprite = _offSprite;

        _onView.SetActive(false);
        _offView.SetActive(true);

        InvokeStateChanged();
    }
}
```

Рис. 3.20 Код класу CeilingFanWidget

За таким самим принципом реалізовуються і всі інші пристрої, починаючи з моделі і закінчуючи графічним.

Після реалізації всіх пристроїв, проект набув виду як на рисунку. Кнопки у меню зліва допомагають користувачу переміщатися між пристроями та взаємодіяти з інтерфейсом кожного пристрою окремо.

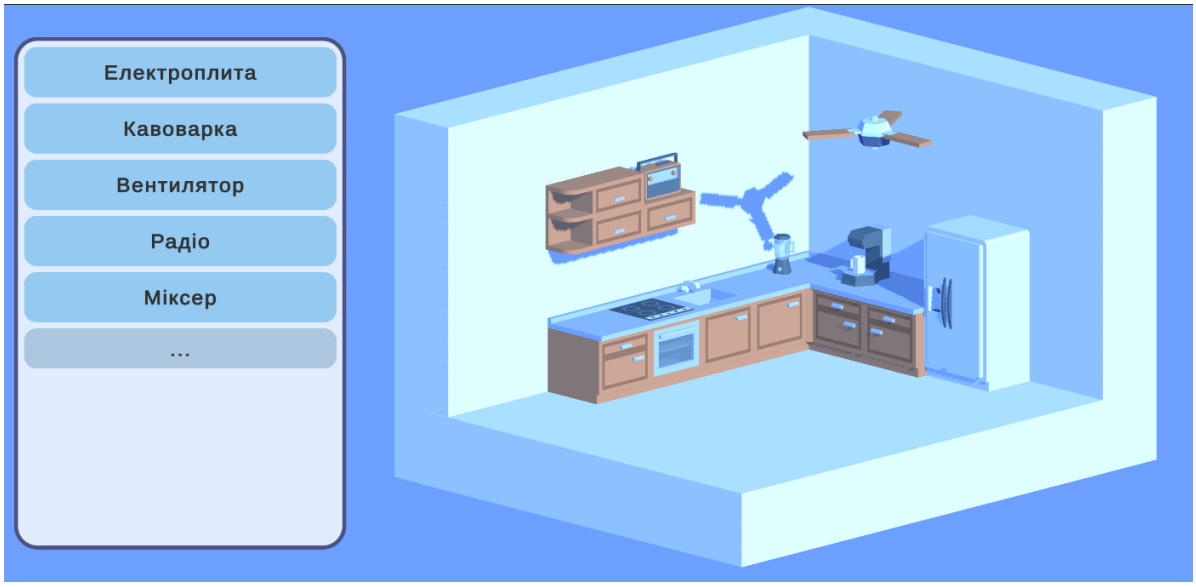

Рис. 3.21 Огляд з головної камери додатку

Розглянемо, що відбудеться, якщо користувач натисне на кнопку з написом “Електроплита”. Після натискання, камера поступово, на певному проміжку часу, наближаться до обраного пристрою та набуває вигляду як на рисунку.

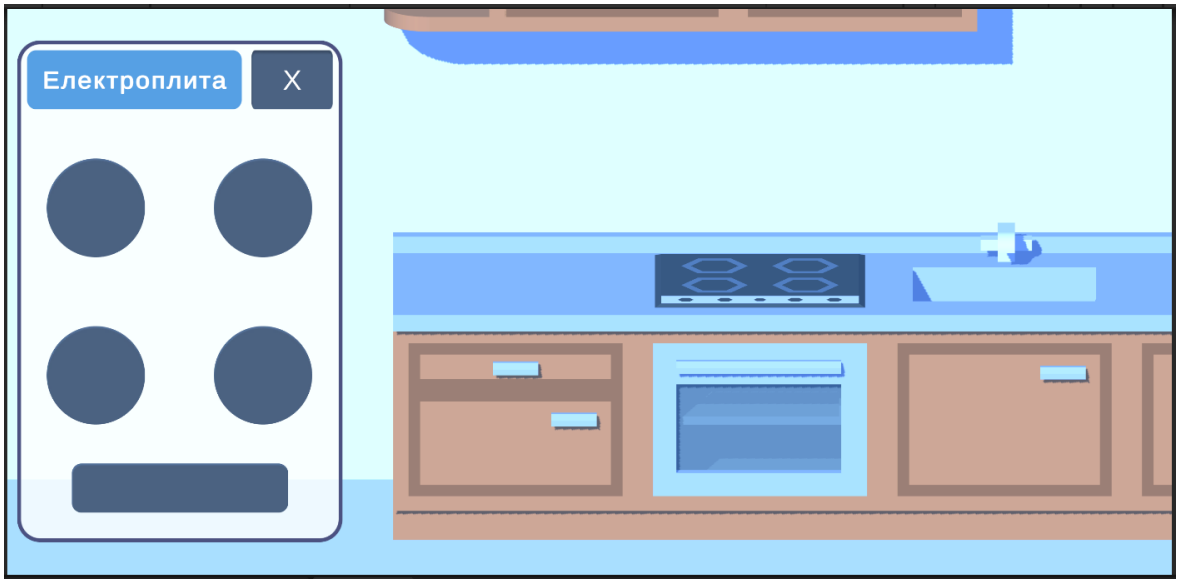

Рис.3.22 Огляд електроплити у застосунку

3.5 Перспективи розвитку мобільного додатку управління розумним будинком

На цьому етапі розробки проект має декілька потенційних напрямків розвитку, до прикладу:

- Швидкість додавання нових пристроїв. Через потребу створити чотири класи для реалізації одного розумного пристрою швидкість є одним з недоліків проекту. Проте цей процес можна буде пришвидшити у майбутньому, створивши зручну компонентну систему, що дозволить окремим спеціалістам, таким як дизайнери, додавати власні пристрої без необхідності залучення у цей процес розробників.
- Повні зміна стилю інтерфейсу. Інтерфейс проекту є не найкращою реалізацією такого. Проблемою такого інтерфейсу швидке перевантаження. Якщо у майбутньому, в кімнаті будинку буде більше ніж п’ятнадцять пристроїв то такий інтерфейс швидко стане не зручним у використанні. Рішенням може стати повні зміна підходу до відображення пристроїв та процесу взаємодії з ними. Замість ручного керування користувачу буде запропонований функціонал, за допомогою якого можна буде створювати безліч сценаріїв роботи пристроїв, що дасть змогу позбутися від безкінечного списку пристроїв.

Такий перехід, забезпечить інший підхід до управління пристроями, наддасть користувачу можливість автоматизувати розумні девайси під власні потреби.

ВИСНОВКИ

На основі досліджень і розробки застосунку можна зробити деякі висновки.

Розробка мобільних додатків для управління розумним будинком є нагальним і багатообіцяючим завданням. Використання елементів гейміфікації в таких додатках може підвищити інтерес користувача і його взаємодію з системою, що, в свою чергу, підвищує загальну ефективність використання розумного будинку.

Серед проаналізованих інструментів розробки інтерактивних додатків було зроблено вибір найбільш підходящих інструментів і технологій для реалізації програми. Використовуючи UML (уніфіковану мову моделювання) для створення діаграм класів, потоків та проектування застосунку вдалося створити чітку архітектуру, яка відповідає останнім вимогам розробки програмного забезпечення.

У застосунку було випробувано використання елементів гейміфікації, таких яких як 3D представлення кімнати з розумними пристроями, що має перспективу позитивно вплинути на залучення користувачів. Ці елементи допомагають підвищити мотивацію користувачів регулярно використовувати додаток і досягати своїх цілей.

Розроблений додаток дозволяє користувачам віддалено взаємодіяти зі своїми розумними домашніми пристроями, що значно покращує зручність та зручність їх використання. Це підтверджує доцільність і ефективність застосування таких рішень в повсякденному житті.

Виконана робота відкриває перспективи для подальшого вивчення та вдосконалення програми. Одним з можливих напрямків є розширення функціональності програми за рахунок інтеграції з новими типами інтелектуальних пристроїв і систем.

В цілому, робота досягла своєї мети, підтвердивши гіпотезу і продемонструвавши успішну реалізацію мобільних додатків для управління розумними будинками з використанням елементів гейміфікації.

ПЕРЕЛІК ПОСИЛАНЬ

1. Android Studio Overview. [Електронний ресурс]. – Режим доступу: [https://developer.android.com/studio/intro]. – Дата доступу: 23.05.2024.

2. Unity User Manual. [Електронний ресурс]. – Режим доступу: [https://docs.unity3d.com/Manual/index.html]. – Дата доступу: 23.05.2024.

3. Figma Design. [Електронний ресурс]. – Режим доступу: [https://www.figma.com/design/]. – Дата доступу: 23.05.2024.

4. Zichermann, G., Cunningham, C. Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps. – Sebastopol, CA: O'Reilly Media, 2011. – 208 p.

5. Burke, B. Gamify: How Gamification Motivates People to Do Extraordinary Things. – Brookline, MA: Bibliomotion, Inc., 2014. – 206 p.

6. Kapp, K. M. The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education. – San Francisco, CA: Pfeiffer, 2012. – 336 p.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

Державний університет інформаційно-комунікаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Розробка мобільного додатку системи управління розумного будинку з використанням елементів гейміфікації»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
освітньо -професійної програми Інформаційні системи та технології

Виконав: Гойна І.С., ІСД -41
Науковий керівник роботи
Полоневич О.В.

Київ - 2024

- ***Актуальність теми:***
- Відсутність комплексних рішень на ринку мобільних додатків для управління розумним будинком.
- Інноваційність використання гейміфікації для підвищення зацікавленості користувачів.
- Важливість підвищення ефективності використання домашніх ресурсів.
- ***Наукова новизна:*** Розробка нового підходу до інтеграції елементів гейміфікації в мобільний додаток для управління розумним будинком.
- ***Об`єкт дослідження:*** управління системою «розумного будинку».
- ***Предмет дослідження:*** елементи гейміфікації у застосунку управління «Розумного будинку».
- ***Мета дослідження:*** покращення дизайну застосунків для управління розумного будинку за рахунок впровадження в його функціональну архітектуру елементів гейміфікації.
- ***Завдання дослідження:***
  - **1.** Аналіз існуючих рішень та технологій для розробки мобільних додатків управління розумним будинком.
  - **2.** Проектування архітектури мобільного додатку з використанням моделей штучного інтелекту та елементів гейміфікації.
  - **3.** Реалізація та тестування мобільного додатку на базі розробленої архітектури.

Гейміфікація

Елементи гейміфікації

Рис. 1 Приклад таблиці лідерів

Рис. 2 Приклад системи рівнів у іграх

Інструменти та прийоми розробки мобільних додатків

Інтегровані середовища розробки

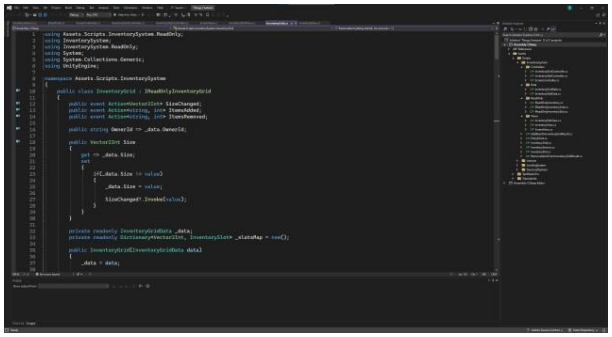

Рис. 3 Інтерфейс інтегрованого Середовища розробки Visual Studio

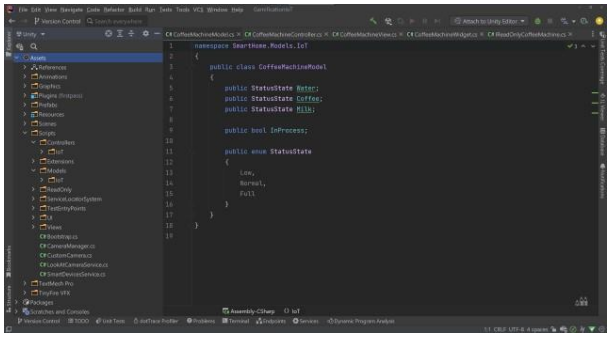

Рис. 4 Інтерфейс інтегрованого середовища розробки Rider

Інструменти та прийоми розробки мобільних додатків

Середовища та рушії розробки

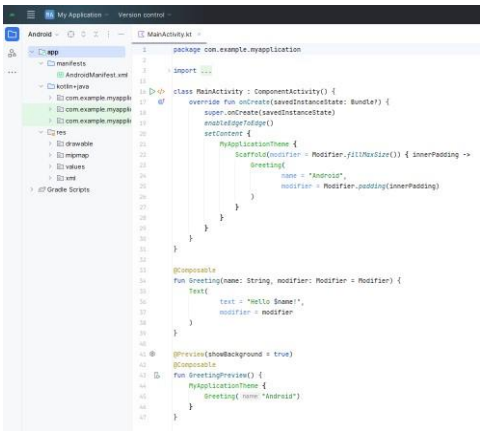

Рис. 5 Середовище розробки Android Studio

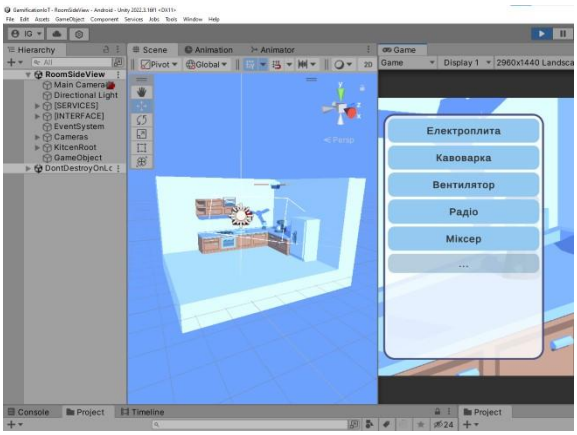

Рис. 6 Рушій розробки додатків Unity 3D

5

Розробка мобільного додатку

Архітектура

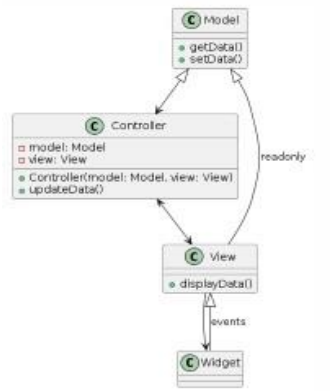

Рис. 7 Діаграма класів архітектури розумних пристроїв

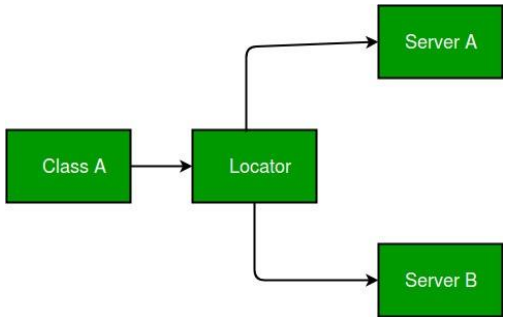

Рис. 8 Діаграма структури сховища сервісів (ServiceLocator)

6

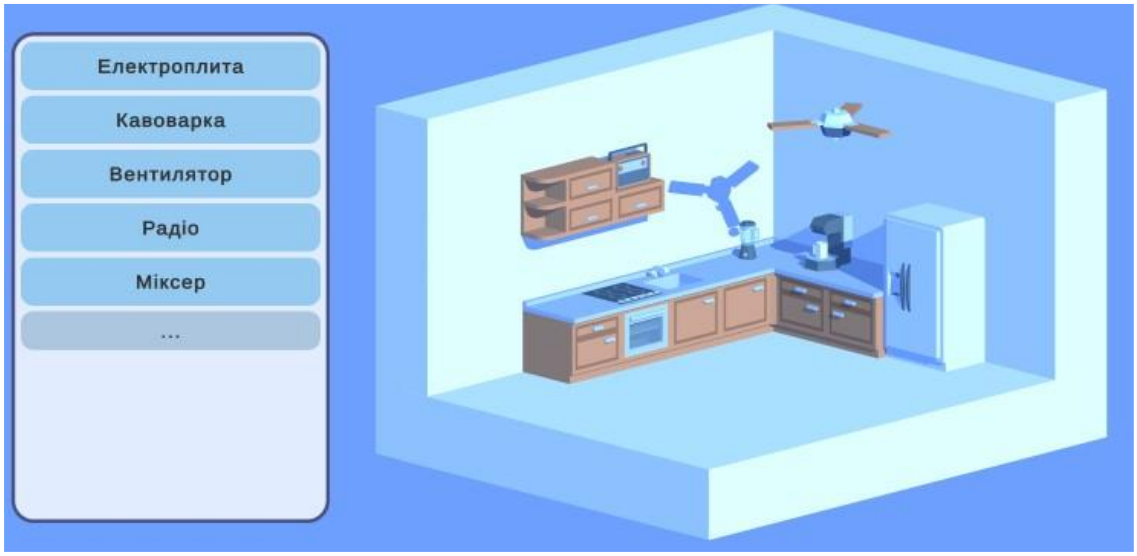

Рис. 9 Головне меню додатку управління розумним будинком

7

Додаток управління розумним будинком у роботі

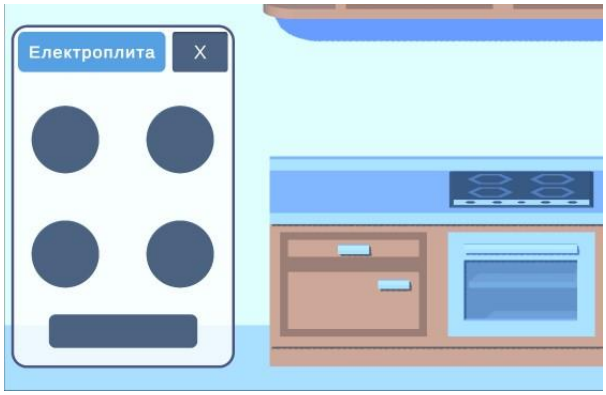

Рис. 10 Приклад інтерфейсу електроплити(вимкнута)

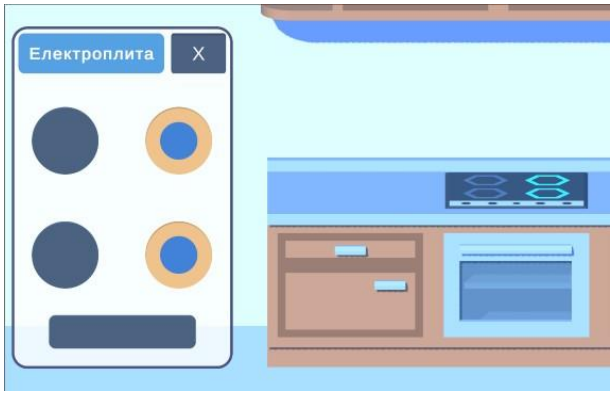

Рис. 11 Приклад інтерфейсу електроплити(увімкнута)

8

Перспективи

- **Швидкість додавання нових пристроїв.**

Необхідність реалізувати:
- Клас моделі даних
- Клас контролер
- Клас контролер представлення
- Клас графічного елементу пристрою

Необхідність додати:
- 3D модель пристрою
- Об’єкт на сцену(налаштувати)

- **Повна зміна стилю інтерфейсу.**

Інтерфейс додатку має орієнтацію на керування конкретним пристроєм, що у подальшому, при збільшенні, кількості пристроїв обов’язково зробить інтерфейс не зручним та переповнений списками.

Рішенням може стати створення системи активностей, що дасть змогу переорієнтувати інтерфейс на можливість планувати дії за часом(створювати сценарії), надасть можливість обирати дію яку хоче виконати користувач у будинок.

Висновки

- Під час роботи було досліджено сучасні рішення у сфері мобільних додатків для управління розумним будинком.
- Наочно продемонстровано переваги використання гейміфікації у системі управління "Розумного будинку", що покращує взаємодію користувачів із системою.
- Окреслено основні технології, використані при розробці додатку, та виокремлено ключові елементи гейміфікації і їх вплив на користувачів.
- Визначено архітектурні рішення для реалізації додатку і обґрунтовано необхідність впровадження нових функцій для підвищення ефективності.

- **Апробація**
- Всеукраїнській науково-технічна конференція «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і Світу». ДУІКТ, 2023.
- V Міжнародній науково-технічній конференції «Сучасний стан та перспективи розвитку IoT». ДУІКТ, 2024.