UNICHECK

Ім'я користувача:
приховано налаштуваннями конфіденційності

ID перевірки:
1015553664

Дата перевірки:
12.06.2023 08:55:17 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
12.06.2023 09:00:45 EEST

ID користувача:
100011372

Назва документа: **Мілюкова ЛІля ок-41**

Кількість сторінок: 23 **Кількість слів:** 3810 **Кількість символів:** 29485 **Розмір файлу:** 1.01 MB ID **файлу:** 1015205855

5.91%
Схожість

Найбільша схожість: **1.21% з джерелом з Бібліотеки** (ID **файлу:** **5945640**)

Пошук збігів з Інтернетом не проводився

5.91% Джерела з Бібліотеки **67** ..... **Сторінка** 25

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

1 ЗАГАЛЬНІ ПІДХОДИ ДО ОРГАНІЗАЦІЇ КОМП'ЮТЕРНИХ ІГОР

1.1 Типи та види комп'ютерних ігор

Існує безліч жанрів ігор, і кожен з них має свої особливості та цільову аудиторію. Ось деякі додаткові пояснення до певних жанрів:

Квести: Це ігри, в яких гравець розв'язує завдання та головоломки, часто в контексті історії або пригод. Головними аспектами таких ігор є логічне мислення та розгадування загадок.

Стрілялки: Це швидкі ігри, в яких гравець має реагувати швидко та точно стріляти ворогів або цілі. Цей жанр покладає акцент на реакцію та навички стрільби.

Рольові ігри (RPG): В цих іграх гравець приймає на себе роль персонажа, якого можна розвивати, покращувати його здібності, виконувати квести та взаємодіяти з ігровим світом. Вони часто мають елементи наративу та відкритий світ для дослідження.

Стратегії: Ці ігри зосереджені на управлінні ресурсами, будівництві, воєнних тактиках та плануванні. Гравець приймає стратегічні рішення для досягнення своїх цілей, таких як розширення території або перемога в військових конфліктах.

Симулятори: Це ігри, які імітують управління реальними об'єктами або ситуаціями, такими як автомобілі, літаки, кораблі або космічні апарати. Вони спрямовані на точність управління та відтворення реалістичного досвіду.

Логічні ігри: Ці ігри спрямовані на розвиток логічного мислення та головоломок. Гравцеві потрібно розв'язати логічні завдання, розумові головоломки або виконати послідовні роздуми для досягнення мети гри.

Азартні ігри: Цей жанр включає ігри гемблінгу, такі як покер, рулетка, слот-машини та інші, де гравець ставить гроші на випадковий результат з метою виграти більше грошей або призів.

Спортивні ігри: Ці ігри моделюють різні види спорту, такі як футбол, баскетбол, теніс, гольф та багато інших. Гравець може керувати спортивними командами або виступати як окремий гравець, відтворюючи реалістичний досвід спорту.

Мультиплеєрні ігри: Це ігри, в яких гравці можуть змагатися або співпрацювати один з одним через Інтернет або мережу. Вони можуть включати онлайн-битви, кооперативний режим гри або масові онлайн-ігри, де гравці взаємодіють у великому віртуальному світі.

Ці жанри ігор лише частина багатогранного світу геймінгу, і кожен з них має свої фанати та специфіку геймплею. Геймінг - це незрівнянний джерело розваг та можливість для гравців насолоджуватися різноманіттям ігрових вражень.

Види ігор за системою монетизації

Дійсно, система монетизації є важливою частиною геймінгової індустрії. Ось деякі типи ігор залежно від системи монетизації:

1. Платні (Paid): Гравець платить один раз за копію гри, отримуючи повний доступ до всього її вмісту. В цьому випадку гра зазвичай не містить додаткових платних елементів або реклами.
2. Умовно-безкоштовні (Freemium, Shareware): Гра доступна безкоштовно, але має додатковий платний контент або функції. Гравці можуть безкоштовно грати в основну частину гри, але для отримання додаткових можливостей або прогресування в грі вони можуть купувати платний контент.
3. З періодичною підпискою (Subscribe): Цей формат монетизації вимагає від гравця регулярної платежів, зазвичай у вигляді місячної або щорічної підписки, щоб мати постійний доступ до гри або додаткових функцій. Цей підхід дозволяє розробникам отримувати стабільний дохід згідно забезпечених послуг або контенту.
4. Безкоштовні з внутрішньо-ігровими покупками і/або рекламою (Free-to-Play): Гравець може завантажити та грати у гру безкоштовно, але гра містить внутрішньо-ігрові покупки, за які гравець може придбати додаткові

ресурси, предмети або переваги. Іноді ці ігри також включають рекламу, яка може бути показана гравцям для заробітку грошей розробниками.

Кожен з цих типів монетизації має свої переваги та недоліки, і це дозволяє гравцям знайти ігри, які відповідають їх

1.2 Комп'ютерні ігри як важливи. засіб формування життєвих навичок

Справді, комп'ютерні ігри сьогодні відіграють важливу роль у розвитку та навчанні. Вони стали більш ніж просто розважальними засобами і набули значного значення у вихованні, навчанні та розвитку різних навичок та вмінь.

Комп'ютерні програми та ігри спрямовані на розвиток психічних функцій дітей допомагають у розвитку їх візуального та слухового сприйняття, уваги, пам'яті, словесних та логічних міркувань. Вони можуть бути особливо корисними для дітей дошкільного та молодшого шкільного віку. Ці ігри сприяють розвитку інтелекту, логічного мислення та уяви, а також удосконалюють увагу, швидкість реакції та сприйняття.

Грати в комп'ютерні ігри може також розвивати різноманітні навички та вміння, які можуть бути корисними у майбутньому. Наприклад, командна гра може сприяти розвитку спілкування та співпраці, стратегічні ігри сприяють розвитку аналітичних та планувальних навичок.

Враховуючи всі ці фактори, можна зробити висновок, що комп'ютерні ігри мають значний потенціал у навчанні та розвитку. Важливо використовувати їх з розумінням і збалансовано, обираючи ігри, які сприяють розвитку певних навичок та цілей, а також враховуючи вікові особливості гравців.

тестування додатку проводиться перевірка його функціональності, стабільності та взаємодії з користувачем. Тестування може включати в себе ручне тестування, автоматизовані тести, тестування на різних пристроях та операційних системах для забезпечення сумісності.

Якщо під час тестування виявляються помилки чи недоліки, проводяться виправлення та покращення. Ітеративний підхід дозволяє вносити зміни та вдосконалення під час розробки на основі зворотного зв'язку та тестування. Це дозволяє покращити якість та функціональність додатку.

Після успішного завершення тестування ітерації розробки, можна переходити до випуску готового мобільного додатку. Це включає підготовку додатку до розповсюдження, наприклад, публікацію в мобільних маркетплейсах, як App Store або Google Play.

Процес розробки ітеративний, що означає, що після випуску першої версії додатку можуть проводитися подальші ітерації та оновлення з метою вдосконалення функціоналу, виправлення помилок та відгуків користувачів. Це дозволяє підтримувати додаток актуальним і покращувати його з часом.

В цьому розділі детальне описання процесу розробки та ітерацій дозволить читачу отримати уявлення про те, як саме ви працювали над розвиваючим мобільним додатком, які кроки були виконані та які методи та підходи використовувалися для забезпечення якості та ефективності розробки.

1.3 Огляд способів розробки мобільних додатків

У поняття розробки мобільних додатків відноситься написання програмного коду для різних програм. Ці програми і додатки можуть встановлюватися на мобільні телефони, смартфони та мобільні пристрої. В процесі розробки мобільного додатку розробники мають розуміти, наскільки увага користувачів обмежена розміром екрану, як зменшити функціонал натискання клавіш, і як вмістити в додаток необхідний набір функцій.

Послуги з розробки мобільних додатків можна відокремити в залежності від платформи, для якої буде створюватися додаток. В основні послуги входить:

- Розробка додатків для iOS
- Розробка додатків для Android
- Розробка VR, AR, MR додатків

Багато компанії пропонують повне створення проекту під ключ.

В пакет послуг з розробки мобільного додатку входять:

- Аналіз завдання, повне занурення в проект і пошук можливих існуючих IT-рішень.
- Розробка дизайнерського рішення (UI / UX / IA-проектування), ТЗ (технічного завдання для розробників).
- Розробка мобільного функціоналу.
- Проведення тестування.
- Реалізація програми (технічне обслуговування і підтримка).

Огляд способів розробки мобільних додатків включає розгляд різних аспектів і підходів до процесу розробки. Основні способи розробки мобільних додатків включають розробку для платформи iOS, розробку для платформи Android та розробку VR (віртуальна реальність), AR (доповнена реальність) та MR (змішана реальність) додатків. Деякі компанії пропонують повний цикл розробки під ключ, що включає аналіз завдання, розробку дизайну, розробку функціоналу, тестування та підтримку.

Аналіз завдання є першим кроком у процесі розробки мобільного додатку. Він включає вивчення вимог та цілей проєкту, розробку технічного завдання і пошук можливих рішень. Цей етап допомагає зрозуміти потреби користувачів та визначити ключові функціональні вимоги до додатку.

Розробка дизайну є важливою складовою частиною процесу розробки мобільного додатку. Вона включає створення дизайну інтерфейсу користувача (UI), дослідження користувацького досвіду (UX) та проектування інформаційної архітектури (IA). Метою цього етапу є створення зручного та привабливого інтерфейсу, який забезпечує зручність використання додатку та задоволення користувачів.

Після розробки дизайну відбувається розробка мобільного функціоналу. Цей етап включає написання програмного коду та реалізацію функцій, які відповідають вимогам і специфікаціям проєкту. Розробники використовують різні програмні мов розробки, такі як Swift для розробки додатків для iOS або Java/Kotlin для розробки додатків для Android. Важливо враховувати особливості кожної платформи та найкращі практики розробки, щоб забезпечити ефективну та оптимізовану роботу додатку.

Після розробки функціоналу важливо провести тестування додатку. Тестування допомагає виявити та виправити помилки, а також переконатися, що додаток працює належним чином і задовольняє вимоги користувачів. Тестування може включати функціональне тестування, тестування взаємодії, тестування продуктивності та інші види тестування, які допомагають забезпечити якість додатку перед його випуском.

Нарешті, після успішного тестування можна переходити до реалізації програми. Це означає підготовку додатку до розповсюдження, включаючи випуск додатку в магазині додатків (наприклад, App Store для iOS або Google Play для Android), налаштування інфраструктури для розгортання додатку та підтримку користувачів.

Після випуску додатку важливо забезпечити його технічне обслуговування та підтримку. Це може включати виправлення помилок, оновлення функціоналу, підтримку нових версій операційних систем або пристроїв та надання відповіді на запити користувачів.

Огляд способів розробки мобільних додатків надає загальну уяву про процес розробки, проте важливо враховувати, що кожен проєкт може мати свої унікальні особливості та вимоги. Індивідуальний підхід до розробки мобільних додатків може включати інші методи і підходи, залежно від конкретної потреби та контексту проекту. Наприклад, існують хайбридні підходи, які поєднують використання веб-технологій (HTML, CSS, JavaScript) з можливостями нативних додатків. Це дозволяє розробляти додатки, які працюють на різних платформах з використанням спільного коду.

Також варто зазначити, що наявні різні інструменти та фреймворки, які полегшують розробку мобільних додатків. Наприклад, для розробки додатків для iOS можна використовувати фреймворк SwiftUI, який надає зручний спосіб створення інтерфейсу користувача. Для розробки додатків для Android можна використовувати фреймворк Android Jetpack, який надає набір інструментів та компонентів для швидкої розробки додатків.

Також варто враховувати зростання популярності хмарних сервісів та мобільних бекенд-платформ. Використання хмарних сервісів дозволяє забезпечити масштабованість, безпеку та доступ до різних функціональних можливостей для мобільних додатків. Мобільні бекенд-платформи надають готові рішення для роботи з базами даних, автентифікації користувачів, сповіщень та інших аспектів розробки мобільних додатків.

Вибір конкретного способу розробки залежить від багатьох факторів, таких як потреби проєкту, технічні вимоги, доступні ресурси та експертиза розробників. Важливо ретельно проаналізувати ці фактори та обрати підходящий розробки мобільних додатків може включати інші методи і підходи, залежно від конкретної потреби та контексту проекту. Наприклад, існують хайбридні підходи, які поєднують використання веб-технологій (HTML, CSS, JavaScript) з можливостями нативних додатків. Це дозволяє розробляти додатки, які працюють на різних платформах з використанням спільного коду.

Також варто зазначити, що наявні різні інструменти та фреймворки, які полегшують розробку мобільних додатків. Наприклад, для розробки додатків для iOS можна використовувати фреймворк SwiftUI, який надає зручний спосіб створення інтерфейсу користувача. Для розробки додатків для Android можна використовувати фреймворк Android Jetpack, який надає набір інструментів та компонентів для швидкої розробки додатків.

Також варто враховувати зростання популярності хмарних сервісів та мобільних бекенд-платформ. Використання хмарних сервісів дозволяє забезпечити масштабованість, безпеку та доступ до різних функціональних

можливостей для мобільних додатків. Мобільні бекенд-платформи надають готові рішення для роботи з базами даних, автентифікації користувачів, сповіщень та інших аспектів розробки мобільних додатків.

Вибір конкретного способу розробки залежить від багатьох факторів, таких як потреби проєкту, технічні вимоги, доступні ресурси та експертиза розробників. Важливо ретельно проаналізувати ці фактори та обрати підходящий спосіб розробки, який найкраще відповідає вашим потребам. Ось кілька додаткових способів розробки мобільних додатків, які варто розглянути:

1. Кросплатформні фреймворки: Кросплатформні фреймворки, такі як React Native, Flutter та Xamarin, дозволяють розробляти додатки, які працюють на різних платформах, використовуючи спільний код. Це забезпечує ефективність в розробці та можливість швидкого випуску додатків на різних платформах.

2. Прогресивні веб-додатки: Прогресивні веб-додатки (Progressive Web Apps, PWA) є додатками, які використовують веб-технології, але мають можливості, подібні до нативних додатків. Вони можуть працювати в офлайн-режимі, мати доступ до пристроєвих функцій і навіть бути встановленими на домашній екран пристрою. PWA можуть бути розроблені з використанням популярних фреймворків, таких як Angular або React.

3. Нативна розробка: Нативна розробка передбачає написання додатків окремо для кожної платформи, використовуючи їхні специфічні мови програмування та інструменти. Це надає найвищий рівень доступу до функцій пристрою та оптимізацію для конкретної платформи. Однак, цей підхід може вимагати більше ресурсів та часу, оскільки потрібно розробляти окремі версії додатку для кожної платформи.

4. Мобільні інтегровані середовища розробки (Mobile Integrated Development Environments, MIDE): Деякі MIDE, такі як Android Studio для розробки додатків для Android або Xcode для розробки додатків для iOS, надають повноцінні набори інструментів та середовища для розробки мобільних

додатків. Вони надають зручний інтерфейс для розробки, тестування та налагодження додатків, а також можливості для розгортання та публікації додатків у відповідних магазинах додатків.

5. Інтеграція з хмарними платформами: Деякі мобільні додатки потребують зберігання даних, аутентифікації користувачів або інших хмарних сервісів. Використання хмарних платформ, таких як Firebase, Amazon Web Services (AWS) або Microsoft Azure, дозволяє швидко та зручно інтегрувати такі сервіси у ваш додаток.

6. Розробка додатків розширеного досвіду: VR (віртуальна реальність), AR (розширена реальність) та MR (змішана реальність) додатки стають все більш популярними. Розробка цих додатків вимагає спеціалізованих знань та інструментів, таких як Unity або Unreal Engine.

Кожен з цих способів має свої переваги та обмеження, тому вибір залежить від ваших конкретних потреб, вимог проєкту, ресурсів та експертизи розробників. Іноді комбінація різних способів розробки може бути оптимальним рішенням для досягнення бажаних результатів.

Важливо також враховувати тренди та зміни в галузі розробки мобільних додатків, так як технології та підходи постійно розвиваються.

2 ВИБІР ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ ПРОЄКТУ

2.1 Вибір мобільної платформи

При розробці мобільних додатків вибір операційної системи відбувається між iOS і Android, а не "IOS і Android". iOS є операційною системою, розробленою Apple, тоді як Android є операційною системою, розробленою Google.

Популярність смартфонів і планшетів привела до появи двох основних операційних систем для мобільних пристроїв - iOS і Android. Linux є операційною системою для комп'ютерів і серверів, а Windows - операційною системою для персональних комп'ютерів.

З точки зору зручності, додатки, розроблені для iOS, будуть працювати на всіх пристроях iPhone та iPad, в той час як додатки для Android можуть бути сумісними з різними пристроями, що працюють на операційній системі Android. Вибір операційної системи залежить від цільової аудиторії та її пристроїв.

Щодо безпеки, iOS відомий своєю строгою системою контролю якості та захисту додатків у своєму AppStore. Android має відкритий характер і дозволяє розробникам більшу свободу, але це також може призвести до більшої кількості потенційних загроз безпеці. Проте, обидві платформи постійно вдосконалюють свої механізми безпеки.

Щодо технічних характеристик, не можна стверджувати, що "велика частина гаджетів на iOS набагато потужніші і багатофункціональні". Якщо порівнювати окремі моделі пристроїв, то можуть бути розбіжності в їх потужності, функціональності та характеристиках.

Щодо цільової аудиторії, дійсно, пристрої iOS, такі як iPhone та iPad, мають вищу середню вартість порівняно з багатьма пристроями на Android. Тому, розробляючи мобільний додаток, важливо враховувати фінансові

можливості цільової аудиторії та розповсюдження пристроїв на обраній платформі.

Загалом, вибір між iOS і Android для розробки мобільного додатку залежить від багатьох факторів, таких як цільова аудиторія, функціональність додатку, безпека, доступні ресурси та інші. Важливо зробити детальний аналіз і вибрати операційну систему, яка найкраще відповідає вашим потребам та цілям.

2.2 Вибір середовища розробки

Xcode — середовище розробки виробництва Apple. Дозволяє створювати програмне забезпечення.Є єдиним засобом написання «універсальних» прикладних програм для Mac OS X. На рисунку 2.1 зображено іконку середовище розробки Xcode.

Рисунок 2.1 – Іконка середовища розробки Xcode

Xcode включає в себе більшу частину документації розробника від Apple та InterfaceBuilder — застосунок, який використовується для створення графічних інтерфейсів.

Xcode підтримує розробку програмного забезпечення для різних платформ Apple, включаючи macOS, iOS, watchOS та tvOS. Воно надає широкий набір функціональності для розробки, таких як інтелектуальне завершення коду, налагодження крок за кроком, інструменти для аналізу продуктивності, інструменти для інтерфейсу користувача та багато іншого.

Xcode також забезпечує інтеграцію з іншими інструментами та сервісами, такими як Git для контролю версій, TestFlight для бета-тестування додатків, Instruments для профілювання та оптимізації продуктивності, а також засоби для публікації програм у магазині App Store.

Загалом, Xcode є потужним інструментом для розробки програмного забезпечення для платформ Apple. Він надає розробникам всі необхідні засоби для створення високоякісних додатків та ігор, тестування та розгортання їх на різних пристроях Apple.

Якщо ви зацікавлені в детальнішій інформації про Xcode та середовище розробки для платформ Apple, я можу додатково розповісти про наступні особливості:

1. Інтерфейс та навігація: Xcode має інтуїтивно зрозумілий інтерфейс користувача, який дозволяє зручно працювати з різними елементами розробки, такими як проекти, файли, код, інструменти та інше. Він також надає можливість швидкого доступу до документації, пошуку помилок та інших корисних ресурсів.

2. Редактор коду: Xcode має потужний редактор коду з підсвічуванням синтаксису, автодоповненням, перевіркою помилок, рефакторингом та багатьма іншими функціями, що полегшують написання та редагування коду.

3. Інструменти для інтерфейсу користувача: Xcode включає в себе Interface Builder, інструмент для створення графічних інтерфейсів за допомогою перетягування та розміщення елементів. Ви можете візуально розробляти інтерфейс своїх додатків, налаштовувати зв'язки між елементами та програмувати їх поведінку.

4. Тестування та налагодження: Xcode надає інструменти для написання й автоматизації тестів, а також для налагодження програмного забезпечення. Ви можете використовувати різні типи тестів (функціональні, юніт-тести, тестування продуктивності тощо) для перевірки роботи своїх додатків.

5. Інструменти для розгортання: Xcode має можливості для розгортання програмного забезпечення на різних пристроях Apple. Ви можете створювати виконувані файли, архіви для розпов

2.3 Вибір мови програмування

Swift є мовою програмування, розробленою компанією Apple, і була представлена у вересні 2014 року. Вона стала альтернативою для розробки програмного забезпечення для платформ Apple, таких як macOS, iOS, tvOS і watchOS.

Swift була створена з метою полегшити процес розробки програмного забезпечення та забезпечити більш безпечне, швидке і ефективне програмування. Вона поєднує в собі кращі аспекти інших мов програмування і пропонує нові можливості, такі як замикання (closures), узагальнене програмування (generics) і елементи функціонального програмування.

Основним застосуванням Swift є розробка користувацьких застосунків для платформ Apple. Вона інтегрується з фреймворками Cocoa і CocoaTouch, що дозволяє розробникам створювати різноманітні програми, включаючи мобільні додатки для iPhone і iPad, додатки для Mac, програми для AppleWatch та телевізійні програми для Apple TV.

Swift отримала популярність серед розробників завдяки своїй простоті, експресивності та безпеці. Вона продовжує розвиватися і оновлюватися, забезпечуючи нові можливості для створення інноваційного програмного

забезпечення для платформ Apple. На рисунку 2.2 зображена іконка мови Swift.

Рисунок 2.2 – Вигляд іконки Swift

Swift надає об'єктну модель, сумісну з Objective-C. Swift щільно інтегровано до власницького середовища розробки Xcode, проте може бути викликано з терміналу, що уможливлює її використання на операційний системах, відмінних від macOS, наприклад, на Linux.

Swift від компанії Apple не треба плутати з давно розробленою скриптовою мовою Swift, поставленого під вільною ліцензією Apache.

Swift є мовою програмування загального призначення, що підтримує статичну типізацію. Вона надає широкий спектр можливостей для розробки різноманітних програмних застосунків.

Swift має сучасний синтаксис, що спрощує розробку коду і покращує читабельність. Вона підтримує багато функцій, які полегшують роботу зі стрічками, колекціями, роботою з файлами, мережевими операціями та багато іншого.

Окрім цього, Swift надає безпеку типів, що допомагає уникнути багатьох помилок програмування та сприяє створенню надійного програмного забезпечення. Мова також має вбудовану підтримку висновку типів, що полегшує написання коду, адже розробникам не потрібно явно вказувати тип змінних чи констант у багатьох випадках.

Також варто зазначити, що Swift активно розвивається спільнотою розробників, і вона має відкритий вихідний код. Це сприяє появі нових функцій, виправленню помилок та забезпечує активну підтримку спільнотою.

У загальному, Swift є потужним і сучасним інструментом для розробки програмного забезпечення для платформ Apple. Вона має багато переваг, які роблять її привабливим вибором для розробників.

Продовжуючи розмову про мову програмування Swift, важливо зазначити, що вона має широку підтримку від Apple та активну спільноту розробників. Тут декілька додаткових характеристик Swift:

1. Висока продуктивність: Swift була оптимізована для роботи зі сучасними процесорами та операційними системами Apple, що забезпечує швидкість виконання програм. Вона використовує передові оптимізації компілятора, що сприяють швидкості виконання коду.

2. Широкі можливості: Swift підтримує різноманітність функцій, таких як функціональне програмування, об'єктно-орієнтоване програмування, узагальнене програмування, паралельне програмування та багато іншого. Це дозволяє розробникам вибирати підхід, який найкраще відповідає їхнім потребам.

3. Легка інтеграція з Objective-C: Swift сумісна з Objective-C, що дозволяє розробникам поступово переходити до Swift, використовуючи наявний код на Objective-C. Це зберігає сумісність з існуючими бібліотеками та фреймворками, що спрощує процес міграції.

4. Розширена безпека: Swift була розроблена з метою забезпечення безпеки програмування. Вона включає в себе ряд функцій, таких як безпечний доступ до пам'яті, опціональні типи, контроль неперевищення масивів та багато інших, що сприяють уникненню помилок та забезпеченню стабільної роботи програм.

4. Широкі можливості розробки: Swift має багатий екосистему інструментів, включаючи Xcode, інтегроване середовище розробки (IDE) від Apple.

3 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

3.1 Структура додатку

Моделі (Models): Моделі представляють дані, бізнес-логіку і бізнес-правила в додатку. Вони відповідають за збереження та обробку даних, а також виконання операцій, пов'язаних з бізнес-логікою. Моделі можуть мати методи для отримання, зміни або видалення даних.

Представлення (Views): Представлення відповідають за відображення даних моделей користувачеві. Вони можуть бути графічним інтерфейсом користувача, сторінками веб-додатка або іншими способами відображення інформації. Представлення отримують дані з моделей і відображають їх користувачу у зручному форматі.

Контролери (Controllers): Контролери служать посередниками між моделями і представленнями. Вони приймають вхідні дані від користувача (наприклад, натискання кнопки) і перетворюють їх на команди для моделей та представлень. Контролери обробляють вхідні дані, виконують необхідні дії з моделями і оновлюють представлення залежно від результатів.

Додатки (Applications): Додатки включають доступні об'єкти, які координують різні компоненти додатка і забезпечують їх взаємодію для обробки запитів. Вони можуть включати роутери, маршрутизатори або інші об'єкти, які керують потоком даних і дій у додатку.

Компоненти додатку (ApplicationComponents): Компоненти додатку є зареєстрованими об'єктами в додатку, які надають різні можливості для обробки запитів. Вони можуть включати сервіси, менеджери, фабрики або інші компоненти, які виконують специфічні завдання в рамках додатку.

Модулі (Modules): Модулі організовані за допомогою декількох модулів і дозволяють логічно групувати пов'язані компоненти, моделі, представлення та контролери разом. Модулі допомагають забезпечити чистоту і структуру додатку.

Фільтри (Filters): Фільтри представляють собою код, який виконується до і після обробки запиту контролерами. Вони можуть виконувати певні дії або операції, такі як автентифікація, перевірка доступу або логування, для підготовки даних або обробки результатів.

Віджети (Widgets): Віджети є об'єктами, які можуть бути вбудовані у представлення. Вони надають додаткову функціональність або можливості відображення, такі як кнопки, текстові поля, списки або графіки.

Ці компоненти допомагають організувати додаток за архітектурним шаблоном MVC, забезпечуючи розділення відповідальностей, модульність і покращену керованість коду.

Структура мобільного зображена на рисунку 3.1.

Рисунок 3.1 – Структура мобільного додатку

Діаграма взаємодії між компонентами зображена на рисунку 3.2.

Рисунок 3.2 – Діаграма взаємодії між компонентами

3.2 Розробка інтерфе.су мобільного додатку

Інтерфейс є надзвичайно важливим аспектом будь-якого додатку, оскільки він визначає спосіб взаємодії користувача з програмним продуктом. Гарно продуманий і зручний інтерфейс допомагає полегшити сприйняття візуальної інформації, забезпечує зручну навігацію та використання функціоналу, а також створює позитивне враження від використання додатку.

Ефективний інтерфейс повинен бути інтуїтивно зрозумілим, зручним у використанні та відповідати потребам і очікуванням цільової аудиторії. Для цього важливо добре вивчити потреби та поведінку користувачів, провести тестування та отримати зворотний зв'язок, щоб врахувати їхні пропозиції та побажання.

Сучасні тенденції в дизайні інтерфейсу включають простоту, чистоту та консистентність, використання іконок, піктограм, зручних макетів, кольорів і шрифтів. Також важливо розробити зручну інформаційну архітектуру, щоб забезпечити логічну структуру та організацію інформації в додатку.

Успішне поєднання функціональності, естетики і зручності в інтерфейсі допоможе залучити користувачів, збільшити їхню задоволеність від використання додатку і підвищити загальний рівень його прийняття.

Я намагалася розробити користувацький інтерфейсякнайбільш зручним. Для цього, в більшості випадків, використовувалися стандартні елементи інтерфейсу, які є перевіреними часом, гармонійно вписуються в загальний стиль iOS 15.

При запуску додатку користувач потрапляє на головний екран, на якому є дві області. Найголовнішим компонентом є робоча область додатку (поле з картками), яка розташовується по центру екрану. Вона дозволяє користувачеві взаємодіяти з нею за допомогою натискання на картки та додаткових кнопок,які розміщені у нижній частині екрану. Варто зазначити, що кожен елемент має своє додаткове вікно UIView,в якому розташований весь йоговласний функціонал. Зображення головної сторінки додатку зображене на рисунку 3.3.

Рисунок 3.3 – Зображення головної сторінки додатку

При натисканні на картки вони відкриються, що ви і можете побачити на рисунку 3.4.

Рисунок 3.4 – Зображення другої сторінки екрану

Якщо користувачу вдасться знайти дві однакові карточки, то вони почнуть крутитися і зникнуть об'єднавшись, що зображено на рисунках 3.5 та 3.6 відповідно.

Рисунок 3.5 – Зображення однакових карток, які об'єдналися

Рисунок 3.6 – Зникнення карток, які об'єдналися

Кнопка RESTART перезапускає гру, а кнопка SHUFFLE перемішує картки між собою.

Схожість

Джерела з Бібліотеки 67

1	Студентська робота	ID файлу: 5945640	Навчальний заклад: Lviv Polytechnic National University	39 Джерело	1.21%
2	Студентська робота	ID файлу: 2001392	Навчальний заклад: Lviv Polytechnic National University		1%
3	Студентська робота	ID файлу: 1005790922	Навчальний заклад: Ternopil Volodymyr Hnatiuk National Pedagog...		0.84%
4	Студентська робота	ID файлу: 1000760239	Навчальний заклад: National Technical University of Ukraine "Kyi...		0.58%
5	Студентська робота	ID файлу: 1003650291	Навчальний заклад: National Technical University of Ukr	4 Джерело	0.5%
6	Студентська робота	ID файлу: 1005785597	Навчальний заклад: National Technical University of Ukraine "Kyi...		0.47%
7	Студентська робота	ID файлу: 1014851242	Навчальний заклад: Vinnytsia State Pedagogical University		0.34%
8	Студентська робота	ID файлу: 1015191474	Навчальний заклад: Lviv Polytechnic National University	3 Джерело	0.26%
9	Студентська робота	ID файлу: 1015178548	Навчальний заклад: National Aviation University		0.24%
10	Студентська робота	ID файлу: 1015184887	Навчальний заклад: National Aviation University		0.24%
11	Студентська робота	ID файлу: 1015088688	Навчальний заклад: Vasyl Stus Donetsk National University		0.24%
12	Студентська робота	ID файлу: 1014853707	Навчальний заклад: Yuriy Fedkovych Chernivtsi National	3 Джерело	0.24%
13	Студентська робота	ID файлу: 1013066798	Навчальний заклад: Taras Shevchenko National University of Kyiv		0.21%
14	Студентська робота	ID файлу: 1011348357	Навчальний заклад: Izmail State University of Humanities		0.21%
15	Студентська робота	ID файлу: 1015057643	Навчальний заклад: Taras Shevchenko National University of Kyiv		0.21%
16	Студентська робота	ID файлу: 1015058482	Навчальний заклад: National Technical University of Ukraine "Kyi...		0.21%
17	Студентська робота	ID файлу: 1015079876	Навчальний заклад: National Technical University of Ukraine "Kyi...		0.21%
18	Студентська робота	ID файлу: 1014971508	Навчальний заклад: Lviv Polytechnic National University	2 Джерело	0.21%
19	Студентська робота	ID файлу: 1009607025	Навчальний заклад: National Technical University of Ukraine "Kyi...		0.21%
20	Студентська робота	ID файлу: 5975713	Навчальний заклад: National Technical University of Ukraine "Kyiv Po...		0.21%

21	**Студентська робота**	ID **файлу**: **1015173353**	**Навчальний заклад**: **National University of Life and Environmenta...**	**0.21%**