**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

«На правах рукопису» «До захисту допущено»

УДК Завідувач кафедри

*Стіренко С.Г.*

(підпис) (ініціали, прізвище)

“ ” 2020 р.

**Магістерська дисертація**

зі спеціальності: *123. Комп’ютерна інженерія\_\_\_\_\_\_\_\_\_\_\_\_*

(код та назва напряму підготовки або спеціальності)

Спеціалізація: *123. Комп’ютерні системи та мережі\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

на тему: Кросплатформенний мобільний клієнт для мережі активних міських ігор

Виконав (-ла): студент (-ка) II курсу, групи ІО-з81мп\_\_\_\_\_\_

(шифр групи)

Городяненко Роман Олексійович

(прізвище, ім’я, по батькові) (підпис)

Науковий керівник професор, д.ф.-м.н., с.н.с. Гордієнко Ю. Г.

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант нормоконтроль, д.т.н., професор Кулаков Ю.О.

(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент доц. каф. АУТС, к.т.н. Писаренко А.В

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Київ – 2020 року

**ЗАВДАННЯ**

**на магістерську дисертацію студенту**

|  |
| --- |
| Городяненко Романа Олексійовича |

(прізвище, ім’я, по батькові)

1. Тема дисертації \_\_Кросплатформенний мобільний клієнт для мережі активних міських ігор \_

Науковий керівник дисертації \_професор, д.ф.-м.н., с.н.с. Гордієнко Ю. Г.

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “**12**” \_**травня**\_ **2020** року **№1092-c**

2. Строк подання студентом дисертації

3. Об’єкт дослідження \_\_\_фронтенд\_клієнти для платформи активних міських ігор, що базуються на клієнт-серверної взаємодії\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Предмет дослідження \_\_\_\_методи та підходи до кросплатформеної клієнтської розробки, та побудова додатку який демонструє таки можливості

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік завдань, які потрібно розробити: \_\_\_\_огляд існуючих рішень та додатків для взаємодії із платформою активних міських ігор, огляд ймовірних платформ для кросплатформенної розробки додатків, огляд методів для взаємодії між клієнтом та сервером, проектування та розробка крос-платформних компонентів, проектування та розробка системи для зберігання даних в оф-лайні, розробка та проектування самого додатку, Опис предметної області, аналіз існуючих рішень, аналіз технологій розробки додатків для взаємодії користувача із ігровою системою, проектування та розробка клієнтської частини додатку для взаємодії користувача із ігровою системою.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультанти розділів проекту

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| нормо контроль | д.т.н., професор Кулаков Ю.О. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання 03.02.2019

Календарний план

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів виконання  дипломного проекту | Строк виконання  етапів проекту | Примітка |
| 1 | Затвердження теми роботи | 10.12.2018 – 15.12.2018 |  |
| 2 | Вивчення та аналіз завдання | 15.12.2018 – 15.03.ц |  |
| 3 | Розробка архітектури та загальної структури систем | 15.03.2019 – 25.03.2019 |  |
| 4 | Розробка структур окремих підсистем | 25.03.2019 – 05.04.2019 |  |
| 5 | Програмна реалізація системи | 05.04.2019 – 15.04.2019 |  |
| 6 | Оформлення пояснювальної записки | 15.04.2019 – 20.05.2019 |  |
| 7 | Розробка компонентів | 03.06.2019 |  |
| 8 | Перед захист | 15.12.2019 |  |
| 9 | Захист | 25.05.2020 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_ \_Городяненко Р. О.\_\_. \_

(підпис) (ініціали, прізвище)

Керівник проекту \_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Гордієнко Ю. Г. \_\_\_\_

(підпис) (ініціали, прізвище)

**Реферат**

**Магістерська дисертація:** 83 ст.,

**Актуальність проблеми:** Зараз дуже стрімко розвиваються саме кросплатформені рішення, що допомагає розробникам писати лише один додаток для усіх ключових платформ. Також у, мною зазначеної, платформи відсутній будь якій клієнт, крім веб-версії.

**Мета і задачі дослідження:** Метою цієї роботи є розробка і створення кросплатформеного програмного забезпечення для платформи активних міських ігор <http://en.cx/>. За допомогою якого можна швидше та ефективніше взаємодіяти із ігровою платформою. Для досягнення цієї мети були поставлені наступні основні задачі: опис предметної області, аналіз існуючих рішень, аналіз технологій розробки додатків для взаємодії користувача із ігровою системою, проектування та розробка крос-платформного клієнтського програмного забезпечення для взаємодії користувача із ігровою системою.

**Об’єкт дослідження:** В роботі магістра розглядається платформа активних міських ігор http://en.cx/. Як практична сторона створено мобільний додаток призначений для більш простої взаємодії з ігровою системою. Також розглядаються фронтенд\_клієнти для платформи активних міських ігор, що базуються на клієнт-серверної взаємодії.

**Предмет дослідження:** Додаток дозволяє вводити коди, бачити завдання та підказки у режимі реального часу, також має варіант роботи в оф лайні. Програмний продукт був створений на мові JavaScript за допомоги технології React Native у візуальному середовищі JetBrains WebStorm. Також розглядаються методи та підходи до кросплатформеної клієнтської розробки, та побудова додатку який демонструє таки можливості.

**Новизна одержаних результатів:** Наукова новизна дисертації полягає у створенні способу взаємодії із платформою активних міських ігор, який базується саме на додатку для мобільної операційної системи є більш відомий для користувача. На відміну від інших існуючих рішень цей спосіб сильно спрощує користування платформою активних міських ігор. Аналогів з таким функціоналом не було знайдено у маркетах або у інших місцях.

**Практичне значення результатів роботи:** Практична цінність полягає у розробці нового крос платформного програмного забезпечення, що дозволяє, на відміну від інших існуючих рішень, більш простіше взаємодіяти із платформою активних міських ігор. Це підвищує ефективність гравців до спрощує досвід користувача.

**Ключові слова:** Крос платформа, фреймворк (платформа для розробки) React Native, мова JavaScript, система Android, система IOS, сервіс Firebase, фреймворк Flatter, бібліотека MobX, React Native повідомлення.

**Abstract**

**Master's Thesis:** 83 pg.,

**The urgency of the problem**: Cross-platform solutions are now rapidly evolving, helping developers to write just one application for all key platforms.

**Purpose and objectives of the research**: The purpose is to create a cross-platform client for the platform of active urban games http://en.cx/. It also aims to develop functionality for offline storage and development

**Object of study**: The master's thesis deals with the platform of active urban games http://en.cx/. As a practical side, the implemented mobile application is intended for easier interaction with the game system.

**Subject of Study:** The application allows you to enter codes, see tasks and tips in real time, also has the option of working offline. The software was created in JavaScript using React Native technology in the JetBrains WebStorm visual environment.

**Novelty of the results:** The way to interact with the platform of active urban games, which is based on the application for the mobile operating system is better known to the user. Also, this method greatly facilitates the use of the platform of active urban games. No analogues with this functionality have been found in marketplaces or other places.

**The practical meaning of the results of the work:** Cross-platform application is created, which makes it easier to interact with the platform of active urban games. This increases the effectiveness of players to simplify the user experience.

**Keywords**: Cross-platform, React Native, JavaScript, Android, iOS, Firebase, Flatter, MobX, React Native Notifications.

ЗМІСТ

[СПИСОК УМОВНИХ ПОЗНАЧЕНЬ 12](#_Toc27725181)

[ВСТУП 13](#_Toc27725182)

[РОЗДІЛ 1 14](#_Toc27725183)

[ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 14](#_Toc27725184)

[1.1. Кросплатформність 14](#_Toc27725185)

[1.2. Кросплатформність мов програмування 14](#_Toc27725186)

[1.3. Кросплатформність на рівні середовища виконання 15](#_Toc27725187)

[1.4. Кросплатформність на рівні середовища виконання 15](#_Toc27725188)

[1.5. Кросплатформність на апаратному рівні 16](#_Toc27725189)

[1.6. Кросплатформність на рівні додатків 16](#_Toc27725190)

[1.7. Нативні додаткі 17](#_Toc27725191)

[1.8. Кросплатформні додаткі 17](#_Toc27725192)

[1.9. Гібридні додатки 18](#_Toc27725193)

[1.10. Передумови появи кросплатформніх рішень 19](#_Toc27725194)

[1.11. Кросплатформні мобільні додаткі 20](#_Toc27725195)

[1.12. Аналіз предметної області 21](#_Toc27725196)

[1.13. Призначення додатку 21](#_Toc27725197)

[1.14. Постановка задачі 21](#_Toc27725198)

[ВИСНОВОК ДО РОЗДІЛУ 22](#_Toc27725199)

[РОЗДІЛ 2 23](#_Toc27725200)

[2. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ 23](#_Toc27725201)

[2.1. PhoneGap 23](#_Toc27725202)

[2.2. Xamarin 24](#_Toc27725203)

[2.3. React Native 25](#_Toc27725204)

[2.4. Flutter 25](#_Toc27725205)

[2.5. Ionic 27](#_Toc27725206)

[2.6. NativeScript 27](#_Toc27725207)

[2.7. Веб-інтерфейс оригінального додатку 28](#_Toc27725208)

[2.7.1. Описання системи 28](#_Toc27725209)

[2.7.2. Технічні характеристики 28](#_Toc27725210)

[2.7.3. Інтерфейс користувача 28](#_Toc27725211)

[РОЗДІЛ 3 31](#_Toc27725212)

[3. ПРОЕКТУВАННЯ ДОДАТКУ 31](#_Toc27725213)

[3.1. Технології 31](#_Toc27725214)

[3.1.1. Мова програмування JavaScript 31](#_Toc27725215)

[3.1.2. Бібліотека React та React Native 31](#_Toc27725216)

[3.1.3. React Native 32](#_Toc27725217)

[3.2. Інструменти розробки 33](#_Toc27725218)

[3.2.1. Середовище розробки WebStorm та Android Studio 33](#_Toc27725219)

[РОЗРОБКА ДОДАТКУ 34](#_Toc27725220)

[3.3. Проектування модулів системи 34](#_Toc27725221)

[3.3.1. Модуль аутентифікації та авторизації 34](#_Toc27725222)

[3.3.2. Модуль моделі гри 37](#_Toc27725223)

[3.4. Проектування інтерфейсу додатку 43](#_Toc27725224)

[3.4.1. Обґрунтування вибору концепції дизайну додатку 43](#_Toc27725225)

[3.4.2. Метод проектування згори донизу 45](#_Toc27725226)

[3.5. Розробка програмного забезпечення 46](#_Toc27725227)

[3.6. Розробка інтерфейсу користувача 48](#_Toc27725228)

[ВИСНОВОК ДО РОЗДІЛУ 54](#_Toc27725229)

[РОЗДІЛ 4 55](#_Toc27725230)

[4. РОЗРОБКА СТАРТАП - ПРОЕКТУ 55](#_Toc27725231)

[4.1. Опис ідеї 55](#_Toc27725232)

[4.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту 56](#_Toc27725233)

[4.3. Технологічний аудит ідеї проекту 56](#_Toc27725234)

[4.4. Технологічна здійсненність ідеї проекту 57](#_Toc27725235)

[4.5. Аналіз ринкових можливостей запуску стартап-проекту 57](#_Toc27725236)

[4.6. Попередня характеристика потенційного ринку стартап-проекту 57](#_Toc27725237)

[4.7. Характеристика потенційних клієнтів стартап-проекту 58](#_Toc27725238)

[4.8. Фактори загроз 59](#_Toc27725239)

[4.9. Фактори можливостей 60](#_Toc27725240)

[4.10. Ступеневий аналіз конкуренції на ринку 61](#_Toc27725241)

[4.11. Аналіз конкуренції в галузі за М. Портером 62](#_Toc27725242)

[4.12. Обґрунтування факторів конкурентоспроможності 62](#_Toc27725243)

[4.13. Порівняльний аналіз сильних та слабких сторін 63](#_Toc27725244)

[4.14. SWOT-аналіз стартап-проекту 63](#_Toc27725245)

[4.15. Альтернативи ринкового впровадження стартап-проекту 64](#_Toc27725246)

[ВИСНОВОК 65](#_Toc27725247)

[ПЕРЕЛІК ПОСИЛАНЬ 67](#_Toc27725248)

[ЛІСТИНГ ДОДАТКУ 70](#_Toc27725249)

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ОС – операційна система;

URL (Uniform Resource Locator) – стандартизована адреса певного ресурсу;

HTTP (HyperText Transfer Protocol) – протокол передачі даних;

UI (User Interface) – інтерфейс користувача;

XML (eXtensible Markup Language) – стандарт побудови [розмітки](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D1%80%D0%BE%D0%B7%D0%BC%D1%96%D1%82%D0%BA%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) ієрархічно структурованих [даних](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%96);

JSON (JavaScript Object Notation) – текстовий [формат](https://uk.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D1%82) обміну [даними](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%96) між [комп'ютерами](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80)

SDK (software development kit) – набір із засобів розробки, утиліт і документації;

IDE (Integrated Development Environment) –  [інтегроване середовище розробки](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BE%D0%B2%D0%B8%D1%89%D0%B5_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8);

API (application programming interface) –  набір [визначень](https://uk.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) взаємодії різнотипного [програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F);

ВСТУП

У наш час дуже багато людей знаходять собі дуже нестандартні хобі. Одне з таких хобі – активні міські ігри. Люди їздять на автомобілях, рішають різні логічні завдання та шукають надписи, які звуться кодами. Той, хто швидше всіх вирішив усі завдання – той переміг.

Але інтерфейс користувача для такої системи вже досить застарів і не відповідає останнім технологіям. Поточний інтерфейс побудовано за допомогою застарілих методів та являє собою звичайний веб-інтерфейс із звичайними проблемами веб інтерфейсу: відсутність нормальної роботи без підключення до мережі, потребує повне перевантаження сторінки при будь-якій дії користувача, що є досить повільною операцією, відсутність різних Push-повідомлень, та інші проблеми.

У сучасний час майже будь-який веб-сервіс має аналог у вигляді Android та/або iOS додатка. Нативні додатки мають багато переваг в порівнянні із звичайним веб-інтерфейсом: можливість працювати без інтернету, доступ до різних API пристрою, система хмарних повідомлень та багато іншого. Отже, метою даного дипломному проекту є розробка додатку для взаємодії користувача із ігровою системою, в основу якого будуть закладені сучасні технології, які допоможуть користувачу швидше та простіше бачити завдання, підказки, та відправляти свої відповіді на сервер самої ігрової платформи.

Актуальність даного додатку полягає у простому інтерфейсі, швидкій взаємодії із сервером та багато платформної архітектурі, що дозволяє запускати додаток на різних ОС.

РОЗДІЛ 1

ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

* 1. Кросплатформеність

Кросплатформність  — властивість програмного забезпечення працювати більш ніж на одній програмній (в тому числі — операційній системі) або апаратній платформі; технології, що дозволяють досягти такої властивості. Кросплатформність дозволяє суттєво скоротити витрати на розробку нового або адаптацію існуючого програмного забезпечення.

* 1. Кросплатформність мов програмування

Кросплатформність на рівні мов програмування досягається шляхом забезпечення незалежності програмного коду від платформи. Багатоплатформними є більшість сучасних високорівневих мов програмування, для яких реалізовані транслятори, що можуть виконуватись на різних платформах. Наприклад, C, С++ і Pascal — кросплатформні мови на рівні компіляції, тобто для цих мов є компілятори під різні платформи. Кросплатформність на рівні редакторів зв'язків досягається реалізацією для різних платформ кросплатформних бібліотек, які реалізують незалежний від платформи інтерфейс, в тому числі — стандартизованих бібліотек. Зокрема, стандартизовані багато бібліотек мови Сі (див. POSIX). Існує також велика кількість нестандартних кросплатформних бібліотек: Qt, GTK+, FLTK, STL, Boost, OpenGL, SDL, OpenAL, OpenCL.

* 1. Кросплатформність на рівні середовища виконання

Кросплатформність на рівні середовищ виконання забезпечується реалізацією в цих середовищах можливостей, необхідних програмам незалежно від платформи. Декларований набір таких можливостей прийнято називати «контрактом» — обов'язком який покладається на середовище, щоб забезпечити виконання програми. Ці обов'язки реалізуються через інтерпретатор, файлові потоки, системні виклики, протоколи, віртуальну машину тощо. Java і C# — кросплатформні мови на рівні виконання, тобто їх виконавчі файли можна запускати на різних платформах без попередньої перекомпіляції. PHP, ActionScript, Perl, Python, Tcl і Ruby — кросплатформні інтерпретовані мови, їх інтерпретатори існують для багатьох платформ.

* 1. Кросплатформність на рівні середовища виконання

Кросплатформність на рівні середовищ виконання забезпечується реалізацією в цих середовищах можливостей, необхідних програмам незалежно від платформи. Декларований набір таких можливостей прийнято називати «контрактом» — обов'язком який покладається на середовище, щоб забезпечити виконання програми. Ці обов'язки реалізуються через інтерпретатор, файлові потоки, системні виклики, протоколи, віртуальну машину тощо. Java і C# — кросплатформні мови на рівні виконання, тобто їх виконавчі файли можна запускати на різних платформах без попередньої перекомпіляції. PHP, ActionScript, Perl, Python, Tcl і Ruby — кросплатформні інтерпретовані мови, їх інтерпретатори існують для багатьох платформ.

* 1. Кросплатформність на апаратному рівні

Кросплатформність на апаратному рівні досягається реалізацією однакових машинних команд та форматом їх представлення, систем переривань, механізмів адресації пам'яті, регістрів тощо. Може досягатись шляхом віртуалізації відповідних ресурсів та механізмів.

* 1. Кросплатформність на рівні додатків

Перед тим як почати міркування на тему крос-платформних рішень, потрібно впевнитися, що ми чітко розуміємо різницю між крос-платформними фреймворками і нативной розробкою. Для простоти сприйняття, ми будемо проектувати процес розробки крос-платформних додатків і порівнювати його з розробкою нативних додатків під iOS. Отже, який же сенс криється за цими термінами.

* 1. Нативні додатки

Якщо говорити про розробку мобільних додатків під платформу Apple iOS, то нативним додатком буде називатися рішення, розроблене на мові програмування Objective-c або Swift (мова програмування Apple для розробки додатків під iOS і OS X, представлений на WWDC 2014 року), в рамках інтегрованого середовища розробки Xcode. З точки зору користувача, нативний додаток нічим не відрізняється від якісно виконаного крос-платформного додатки, але все ж варто відзначити, що нативні додатки під конкретну мобільну операційну систему, зазвичай, мають зрозумілий користувачеві (нативний) інтерфейс. Зручність полягає в тому, що при розробці нативних додатків зв'язка UI / UX фахівця і програміста найбільш ефективна: дизайнер точно знає які UI підходи звичні користувачам конкретної операційної системи, наприклад кнопка «Back» і нижній Tab Bar в iOS, а iOS розробник точно знає як реалізувати ті чи інші UI фішки саме для мобільних пристроїв виробництва Apple. Як результат, користувач відкриває нативний додаток і інтуїтивно взаємодіє з новим інтерфейсом.

* 1. Кросплатформні додатки

Важливо розуміти чому додаток називається крос-платформних. Справа в тому, що, насправді, сам виконуючий файл, скомпільований під одну з мобільних платформ, не може бути запущений в інший ОС. Тобто якщо було розроблено крос-платформне рішення і зібрано під Android, це не означає, що можна взяти файл з розширенням «\* .apk» і запустити його на iPad'e. Ідея крос-платформних рішень не в зручності для користувача, а в оптимізації процесів розробки мобільного застосування. Отже, крос-платформний додаток - це рішення розроблене таким чином, щоб мати можливість, з мінімальними зусиллями, скомпілювати вихідний код для виконання на кількох мобільних платформах, але результатом кожної окремої компіляції будуть окремі виконувані файли. Наприклад, під iOS виконуваний файл має розширення - «\* .ipa», під Android - «\* .apk» і т.д.

* 1. Гібридні додатки

Найчастіше це додатки, при розробці яких використовуються веб-технології, а не розробка в нативному середовищі (пр. Зв'язка Xcode і Objective-c в разі розробки під iOS). В результаті виходить веб-додаток, що запускається в «обгортці» і подається не як веб-сторінка, а як окремий додаток, що вимагає установки і має окрему іконку. Гібридні рішення досить популярні серед розробників крос-платформних движків, обумовлюється це тим, що з функцією веб-браузера може однаково впоратися практично будь-яка мобільна ОС, а значить якщо додаток уже працює під будь-якої мобільної OS, то запустити її на інший не складе труднощів . Звичайно, є свої нюанси, про які ми поговоримо в окремому огляді активно що просувається гібридного фреймворка, ім'я якому Phone Gap.

* 1. Передумови появи кросплатформних рішень

Які завдання покликані вирішити крос-платформні движки, чому хтось вирішив, що в цьому є сенс? Якщо ви запитаєте у Java-розробника під Android або Objective-c розробника під iOS про те, що вони думають з приводу крос-платформних фреймворків під мобільні платформи, то в більшості випадків отримаєте однозначну негативну характеристику і ще кілограм епітетів.

Основна причина такої реакції полягає в тому, що хороші розробники, як і будь-який інший професіонал, дуже трепетно ​​ставляться до самої концепції розробки, ефективності, продуктивності, у кожного з них своє бачення того «що таке добре, а що таке погано?», Але зазвичай все крутиться навколо технології якою вони володіють, пестять і плекають. Важливо розуміти, що далеко не кожен розробник може пожертвувати програмістської романтикою і дати об'єктивну оцінку тому, що насправді вигідно для бізнесу в конкретний момент. Загалом сталося так, що вирішує всі, як зазвичай, бюджет, тому основною передумовою для появи крос-платформних рішень стає саме користь для бізнесу, яку можна деталізувати наступним чином:

* при розробці мобільного додатку під 2-3-х мобільних OS на крос-платформному фреймворку велика частина коду може бути написана одним фахівцем. Тобто відпадає необхідність в full-time розробника під кожну окрему OS;
* відсутність на ринку робочої сили вузьких фахівців під уже досить велика кількість мобільних платформ;
* відносна простота в контексті контролю відповідності між версіями програми під різними мобільними ОС.
  1. Кросплатформні мобільні додатки

Важливо розуміти, що існує безліч причин, за якими один розробник вибере одну платформу, а інший іншу. Наприклад, розробник на мові Ruby не буде дивитись в бік Xamarin, тому що у нього, швидше за все, відсутній досвід розробки на C #. Але якщо Xamarin виявиться в руках досвідченого C # розробника, то в результаті клієнт отримає додатки під Android та iOS, які нічим не поступаються нативним.

З іншого боку, незважаючи на те, що в команді є 2-3 відмінних C # розробника, Xamarin може виявитися дорогим в реалізації, поступившись дешевому PhoneGap, якщо проект не вимагає ніякої особливої ​​роботи з графікою і апаратними засобами пристроїв, а планується лише як MVP прототип для перевірки бізнес моделі та повинен бути реалізований в рамках вкрай обмеженого бюджету.

Ще одним досить суттєвим параметром є ціна за доступ до середовища розробки на конкретному фреймворку. Якщо PhoneGap повністю безкоштовний інструмент, то Vеrivo хоче $ 60,000 / рік, що створить подив у стартаперів, та й не тільки у них.

* 1. Аналіз предметної області

Дослідивши поточний веб-інтерфейс платформи було помічено, що він використовує дуже старі технології, наприклад бібліотеку jQuery, зовсім немає AJAX-запитів та взагалі інтерфейс працює досить повільно.

* 1. Призначення додатку

Додаток створено для того, щоб будь-який користувач ігрової платформи зміг зайти на GooglePlay або AppStore та завантажити собі сам додаток, щоб простіше та швидше взаємодіяти із ігровою системою.

* 1. Постановка задачі

Додаток для взаємодії із ігровою платформою, що розробляється у даній магістерській дисертації, націлений на покращення та спрощення процесу самої гри. При створенні цього додатку, на базі аналізу предметної області були поставлені наступні задачі:

* Реалізувати якомога зручніший та швидкий процес отримання даних гри використовуючи API;
* Можливість відправлення кодів на сервер;
* Забезпечити можливість роботи без підключення до інтернету;
* Зробити роботу додатку стабільною навіть на «слабких» за своїми технічними характеристиками пристроях;

При розробці мають бути використані сучасні технології.

ВИСНОВОК ДО РОЗДІЛУ

В даному розділі було проаналізовано наявні аналоги та їх недоліки. Ознайомлення з предметною областю дало можливість більш детально сформулювати призначення розроблюваного додатку.

Були поставлені основні задачі, що мають бути реалізовані у фінальній версії додатку.

Метою даного дипломного проекту є втілення нових інструментів для комунікації користувачами та ігровою системою. Усі задачі були поставлені на основі аналізу існуючих рішень та потреб користувачів.

РОЗДІЛ 2

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ
   1. PhoneGap

PhoneGap - це основа для розробки мобільних додатків, вироблених Nitobi, а згодом придбаних Adobe Systems.3 4 Головним чином, PhoneGap дозволяє програмістам розробляти додатки для мобільних пристроїв, використовуючи загальні інструменти, такі як JavaScript, HTML5 та CSS3. Отримані в результаті програми є гібридними, тобто вони насправді не є власними програмами пристрою (оскільки візуалізація здійснюється через веб-перегляди, а не з певними графічними інтерфейсами кожної системи), але вони також не є веб-додатками (враховуючи, що це програми, упаковані таким чином, що їх можна розгорнути на пристрої, навіть працюючи з нативної API API). Третя версія PhoneGap включає використання командного інтерфейсу через консоль, нову архітектуру децентралізованих доповнень та можливість використання уніфікованого веб-коду для створення декількох проектів. PhoneGap обробляє API, які дозволяють отримати доступ до таких елементів, як акселерометр, камера, контакти на пристрої, мережа, зберігання, сповіщення тощо. Ці API підключаються до операційної системи за допомогою власного коду хост-системи через зовнішній функціональний інтерфейс у Javascript. PhoneGap дозволяє розробляти або запуск програм у нашому веб-браузері, не використовуючи тренажер, присвячений цій задачі, і пропонує можливість підтримувати функції в таких рамках, як Sencha Touch або JQuery Mobile. PhoneGap - це розповсюдження Apache Cordova.5 Програма спочатку називалася "PhoneGap", а пізніше "Apache Callback". Обидві системи мають майже однакові функції, головна відмінність Apache Cordova від Phonegap полягає в тому, що друга має доступ до послуг з компіляції хмар, що надаються Adobe Creative Cloud. Apache Cordova - це програмне забезпечення з відкритим кодом, і це і PhoneGap можна вільно використовувати в будь-якій програмі без необхідності атрибуції чи будь-яких ліцензій.

* 1. Xamarin

Xamarin - компанія, що належить Microsoft у Сан-Франциско, заснована в травні 2011 року інженерами, які створили Mono, Xamarin.Android (раніше Mono для Android) і Xamarin.iOS (раніше MonoTouch), які є впровадженням загальних Мовно інфраструктура (CLI) та загальні специфікації мови (часто їх називають Microsoft .NET).

За допомогою кодової бази С #, розробники можуть використовувати інструменти Xamarin для запису нативних програм для Android, iOS та Windows з власним інтерфейсом користувача та обміну кодом на багатьох платформах, включаючи Windows, macOS та Linux. За даними Xamarin, понад 1,4 мільйона розробників використовували продукцію Xamarin у 120 країнах світу станом на квітень 2017 року.

24 лютого 2016 року Microsoft оголосила, що підписала остаточну угоду на придбання Xamarin.

* 1. React Native

Принципи роботи React Native практично ідентичні React, за винятком того, що React Native не маніпулює DOM за допомогою Virtual DOM. Він запускається у фоновому процесі (який інтерпретує JavaScript, написаний розробниками) безпосередньо на кінцевому пристрої та спілкується з нативною платформою через серіалізацію, асинхронний та пакетний міст.

Реактивні компоненти обертають наявний кодовий код та взаємодіють із нативними API через декларативну парадигму інтерфейсу користувача та JavaScript. Це дає змогу розробляти нативну програму для цілих нових команд розробників, а також дозволяти існуючим власним командам працювати набагато швидше.

React Native не використовує HTML. Натомість повідомлення з потоку JavaScript використовуються для маніпулювання нативними View.

* 1. Flutter

Flutter - це набір для розробки програмного забезпечення з відкритим кодом, створений Google. Він використовується для розробки програм для Android, iOS, Windows, Mac, Linux, Google Fuchsia та Інтернету[10].

Перша версія Flutter була відома під кодовою назвою "Sky" і працювала в операційній системі Android. Він був оприлюднений на саміті розробників Dart 2015 року, із заявленим наміром мати можливість послідовно малювати зі швидкістю 120 кадрів в секунду. Під час основного виступу днів розробників Google у Шанхаї Google оголосив Flutter Release Preview 2 - останній великий реліз перед Flutter 1.0. 4 грудня 2018 року на заході Flutter Live вийшов Flutter 1.0, що позначає першу "стабільну" версію Framework. 11 грудня 2019 року на заході Flutter Interactive було випущено Flutter 1.12, було оголошено, що Flutter - це перша платформа інтерфейсу, призначена для обчислення навколишнього середовища.

Програми Flutter написані мовою Dart і використовують багато вдосконалених можливостей мови. У Windows, macOS та Linux за допомогою офіційного проекту Flutter Desktop Embedding Flutter працює у віртуальній машині Dart, яка має функціональний механізм виконання. Під час написання та налагодження програми Flutter використовує компіляцію Just In Time, що забезпечує «гаряче перезавантаження», за допомогою якого модифікації вихідних файлів можна вводити в запущену програму. Flutter поширює це на підтримку сталого гарячого перезавантаження, де в більшості випадків зміни вихідного коду можуть бути відображені негайно в запущеному додатку, не вимагаючи перезавантаження або втрати стану. Ця функція, реалізована у Flutter, отримала широку оцінку [потрібна цитата]. Версії випуску додатків Flutter збираються заздалегідь (AOT) на Android і iOS, завдяки чому можлива висока продуктивність Flutter на мобільних пристроях.

* 1. Ionic

Ionic - це повний SDK з відкритим кодом для розробки гібридних мобільних додатків, створений Максом Лінчем, Бен Сперрі та Адамом Бредлі з Drifty Co. в 2013 році. Оригінальна версія була випущена в 2013 році та побудована на вершині AngularJS та Apache Cordova. Однак остання версія була відновлена як набір веб-компонентів, що дозволяє користувачеві вибирати будь-які рамки інтерфейсу користувача, такі як Angular, React або Vue.js. Він також дозволяє використовувати компоненти Ionic без фреймворку користувача. Ionic надає інструменти та послуги для розробки гібридних мобільних, настільних та прогресивних веб-додатків на основі сучасних технологій та практик веб-розробок, використовуючи такі веб-технології, як CSS, HTML5 та Sass. Зокрема, мобільні додатки можуть бути побудовані за допомогою цих веб-технологій, а потім розповсюджуватися через нативні магазини додатків, які встановлюватимуться на пристрої, використовуючи Cordova або Capacitor.

* 1. NativeScript

NativeScript - це програма з відкритим кодом для розробки додатків на платформах Apple iOS та Android. Спочатку він був задуманий і розроблений компанією Progress. Програми NativeScript створені за допомогою JavaScript або за допомогою будь-якої мови, яка перекладається на JavaScript, наприклад, TypeScript. NativeScript підтримує рамки Angular та Vue JavaScript. Мобільні програми, побудовані за допомогою NativeScript, призводять до повністю нативних програм, які використовують ті самі API, що і в Xcode або Android Studio. Крім того, розробники програмного забезпечення можуть призначати сторонні бібліотеки з CocoaPods, Android Arsenal, Maven та npm.js у своїх мобільних додатках без необхідності обгортки.

* 1. Веб-інтерфейс оригінального додатку
     1. Описання системи

Звичайний веб-інтерфейс є офіційним рішенням у цій галузі. Зроблено у вигляді звичайної HTML-сторінки. Підтримує як мобільні, так і звичайні екрани. Дозволяє бачити завдання, сектори, підказки, та вводити відповіді.

* + 1. Технічні характеристики

Розроблено на мові JavaScript та використовує таку бібліотеку, як jQuery. Працює через звичайні HTTP-форми

* + 1. Інтерфейс користувача

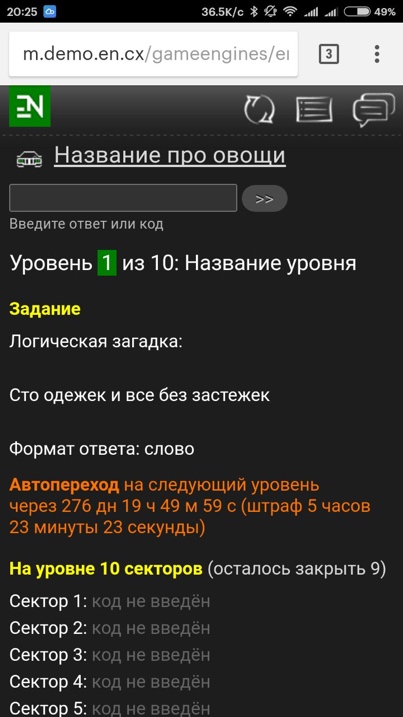


Рис. 2.1 Інтерфейс веб-додатку у браузері «Google Chrome»

ВИСНОВОК ДО РОЗДІЛУ

В даному розділі було проаналізовано наявні аналоги та їх недоліки. Ознайомлення з предметною областю дало можливість більш детально сформулювати призначення розроблюваного додатку.

Були поставлені основні задачі, що мають бути реалізовані у фінальній версії додатку.

Метою даного дипломного проекту є втілення нових інструментів для комунікації користувачами та ігровою системою. Усі задачі були поставлені на основі аналізу існуючих рішень та потреб користувачів.

РОЗДІЛ 3

1. ПРОЕКТУВАННЯ ДОДАТКУ
   1. Технології
      1. Мова програмування JavaScript

JavaScript (JS) — динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. Мова JavaScript також використовується для програмування на стороні сервера (подібно до таких мов програмування, як Java і C#), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), всередині PDF-документів тощо.

* + 1. Бібліотека React та React Native

React.js, здебільшого називають React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці одно сторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.[3] React дозволяє розробникам створювати великі веб- застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид- контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. В даний час React використовують Khan Academy, Netflix, Yahoo, Airbnb, Sony, Atlassian та інші.

* + 1. React Native

Принципи роботи React Native практично ідентичні React, за винятком того, що React Native не маніпулює DOM за допомогою Virtual DOM. Він запускається у фоновому процесі (який інтерпретує JavaScript, написаний розробниками) безпосередньо на кінцевому пристрої та спілкується з нативною платформою через серіалізацію, асинхронний та пакетний міст.

Реактивні компоненти обертають наявний кодовий код та взаємодіють із нативними API через декларативну парадигму інтерфейсу користувача та JavaScript. Це дає змогу розробляти нативну програму для цілих нових команд розробників, а також дозволяти існуючим власним командам працювати набагато швидше.

React Native не використовує HTML. Натомість повідомлення з потоку JavaScript використовуються для маніпулювання нативними View.

* 1. Інструменти розробки
     1. Середовище розробки WebStorm та Android Studio

В якості IDE для розробки додатку було обрано WebStorm. На це є декілька причин:

* Android Studio розробляється тією ж корпорацією, що і ОС Android, а саме Google Inc. Це означає більш зручну інтеграцію усіх необхідних функцій для розробки під найсучасніші версії Android.
* WebStorm включає в себе зручне управління усіма процесами які потрібні під час розробки.
* Ще одним важливим фактором є наявність зручного емулятора Android пристроїв. Це дозволяє тестувати додаток під час розробки прямо на ПК, на якому ведеться розробка, а також провести тестування на різних пристроях з різними технічними характеристиками.
* Зручне вистежування помилок. Крім звичних лагів можна розділити відстеження для різних процесів, потоків і додатків, що вкрай необхідно робити, тому що найчастіше додатки будуть використовувати створення нових потоків, а в окремих випадках ще й взаємодіяти з іншими додатками.
* Підтримка системи автоматичної зборки проектів Webpack.

РОЗРОБКА ДОДАТКУ

* 1. Проектування модулів системи

Даний підрозділ описує структуру модулів, що складають дану систему, їх властивості та взаємозв’язки один з одним. З визначення можна прийняти модуль як поняття з чітко окресленими межами, що має сенс у контексті прикладної проблеми, що розглядається.

Основною метою розробки модульної моделі є визначення і опис об’єктів, що становитимуть у сукупності проектований додаток.

* + 1. Модуль автентифікації та авторизації

Так як автентифікація користувачів відбувається на серверу самої ігрової платформи Encounter, для відправки запитів та отримання відповідей від сервера потрібно мати дійсний Cookie, який видається автоматично при авторизації у системі. В процесі автентифікації сервер перевіряє валідність логіна та паролю та генерує Cookie маркер сеансу (session token) який буде міститися у кожному запиті до Encounter API. Маркер сеансу посилається у заголовку HTTP-запиту під назвою Cookie.

Авторизація на сайті Encounter здійснюється шляхом відправки наступного запиту на сервер

POST <ім'я сайту> / login / signin?json = 1

------------------------------------

POST параметри:

<Login> = ім'я користувача

<Password> = пароль

Відповіддю сервера буде JSON об'єкт, який має наступну структуру:

{

"Error": 0, // Код помилки

"Message": "", // Локалізоване повідомлення про помилку

"IpUnblockUrl": null, // URL сторінки розблокування IP адреси

"BruteForceUnblockUrl": null, // URL сторінки сторінки розблокування при підозрі на брутфорс

"ConfirmEmailUrl": null, // URL сторінки підтвердження E-Mail

"CaptchaUrl": null, // URL до зображення захисного малюнку

"AdminWhoCanActivate": null // Адміністратори які можуть активувати обліковий запис

}

Також, у своєму HTTP-відповіді сервер поверне колекцію Cookies, в якій

слід звернути увагу на:

GUID = 29e408d5%2D0061%2D4d1c%2Da145%2D2a07e6993c0f

stoken = cwxi0ylkexg2mv4i1vx30hvd

atoken = uid%3d40481%26iss%3d0%26iscd%3d1%26tkn%3d1d

Для утримання сесії (авторизації) і цілей безпеки, необхідно забезпечити

передачу цих значень назад на сервер, в кожному наступному запиті, якому

необхідно виконання в контексті користувача.

Успішна авторизація.

У разі, якщо авторизація пройшла успішно, властивість JSON-об'єкта

Error дорівнюватиме 0, на цьому авторизація закінчена і більше ніяких

дій робити не потрібно.

Помилка авторизації

Якщо в процесі авторизації сталася помилка, то властивість Error буде

набувати наступних значень:

1 - Ви перевищили кількість неправильних спроб авторизації

властивість CaptchaUrl буде містити посилання на картинку, для

завантаження картинки захисту (спосіб захисту від роботів) необхідно передати header параметр

Referer = / login, для повторної авторизації на сервері до POST запиту

потрібно додати параметр MagicNumbers c цифрами отриманими з

картинки захисного малюнку.

2 - Неправильний логін або пароль

3 - Користувач або в чорному списку, або на домені не можна з інших доменів

4 - У користувача в профілі включена блокіровка по IP, поточний IP не входить в список дозволених

властивість IpUnblockUrl ініціалізується посиланням на сторінку розблокування обмеження по IP.

5 - У процесі авторизації сталася помилка на сервері

6 - Не використовується в JSON запитах

7 - Користувач заблокований адміністратором

8 - Останній зареєстрований учасник не активований

властивість AdminWhoCanActivate буде створити масивом адміністраторів, які можуть активувати обліковий запис.

9 - Вирішення розцінені як брутфорс

властивість BruteForceUnblockUrl створюється посиланням на сторінку розблокування.

10 - Користувач не підтвердив E-Mail

властивість ConfirmEmailUrl створюється посиланням на сторінку підтвердження E-Mail.

* + 1. Модуль моделі гри

Даний модуль відповідає за усе, що пов’язано з моделлю самої гри.

Отримання тексту завдання, підказок, бонусів та відправлення відповідей на

сервер ігрової платформи. Для відправлення запитів використовується

протокол HTTP та бібліотека axios[5].

Початок роботи з API

Для початку роботи з API необхідно виконати наступні кроки:

• Авторизація на сайті.

• Здійснення запиту на отримання JSON об'єкта GameEngineModel

• Оцінити поточний стан гри

Здійснення запиту для отримання GameEngineModel

|  |  |
| --- | --- |
| HTTPVERB | GET |
| URL | /GameEngines/Encounter/Play/{ID Гри}?json=1 |
| Приклад | /GameEngines/Encounter/Play/2020?json=1 |

Реакцією сервера на даний запит буде один з наступних відповідей:

• JSON об'єкт (GameEngineModel), що представляє собою поточний стан гри

• Повернення на сторінку логіна (користувач не на сайті або зруйнована сесія)

• Повернення на сторінку захисту від DDOS (якщо надсилається забагато запитів від IP)

• Повернення на сторінку помилки (якщо сталася помилка на сервері)

Оцінка поточного стану гри

Після відправки вищевказаного запиту, якщо не відбулося помилок, то у відповідь буде повернутий JSON об'єкт:

|  |  |
| --- | --- |
| Event (int) | Відображає в якому стані знаходиться гра |
| GameId (int) | ID гри |
| GameNumber (int) | номер гри |
| GameTitle (string) | назва гри |
| LevelSequence (int) | тип послідовності: 0 - лінійна, 1 -вказана, 2 - випадкова, 3 - штурмова, 4 - динам. випадкова |
| UserId (int) | ID гравця |
| TeamId (int) | ID команди гравця |
| EngineAction (object) | інформація про результат останнього запиту гравця |
| Level (object) | інформація про поточний рівень |
| Levels (object array) | список всіх рівнів |

Після кожного запиту необхідно перевіряти яке значення містить властивість Event, яке може набувати таких значень:

|  |  |
| --- | --- |
| **Код** | **Опис** |
| 0 | **Гра в нормальному стані** |
| 2 | Гра з зазначеним ID не існує |
| 3 | Гра не відповідає запрошенням Engine |
| 4 | Гравець не на сайті |
| 5 | Гра не почалася |
| 6 | Гра скінчилась |
| 7 | Не подана заявка (гравцем) |
| 8 | Не подана заявка (командою) |
| 9 | Гравець ще не прийнятий в гру |
| 10 | У гравця немає команди (в командній грі) |
| **Код** | **Опис** |
| 11 | Гравець не активний в команді (в командній грі) |
| 12 | У грі немає рівнів |
| 13 | Перевищено кількість учасників в команді (в командній грі) |
| 16 | Рівень знятий |
| 17 | Гра закінчена |
| 18 | Рівень знятий |
| 19 | Рівень пройдений автопереходом |
| 20 | Всі сектора відгадані |
| 21 | Рівень знятий |
| 22 | Тайм-аут рівня |

Варіанти дій в залежності від поточного значення властивості Event

|  |  |
| --- | --- |
| **Код** | **Необхідна дія** |
| 0 | Гра в нормальному стані, можна продовжувати посилати запити |
| 16, 18, 19, 20, 21, 22 | Поточний рівень змінився, необхідно надіслати запит на сервер ще раз |
| всі інші коди | Вивести повідомлення гравцеві відповідне коду помилки |

Оцінка поточного стану рівня

У разі якщо гра активна, і допускає участь гравця (model.Event = 0), перед відправкою відповідей гравця на сервер, необхідно упевнитися, що введення відповідей на даному рівні дозволений, для цього потрібно перевірити об'єкт model.Level, який містить інформацію про поточний рівні і має наступні властивості:

|  |  |
| --- | --- |
| LevelId | ID Рівня |
| Name | ім'я рівня |
| Number | номер рівня |
| Timeout | час (в секундах) спрацьовування автопереходу, 0 - якщо немає |
| TimeoutAward | штраф за автоперехід (в секундах), 0 - якщо немає |
| TimeoutSecondsRemain | залишилося часу до спрацьовування автопереходу (в секундах) |
| IsPassed | рівень пройдений |
| StartTime | час початку рівня для гравця |
| HasAnswerBlockRule | чи є на рівні блокування відповідей |
| BlockDuration | залишилося секунд блокування; 0 - не активна |
| BlockTargetId | блокування встановлена для: 0,1 - для гравця; 2 - для команди |
| AttemtsNumber | кількість спроб дозволених в рамках AttemtsPeriod |
| AttemtsPeriod | період спрацьовування блокування (в секундах) |
| RequiredSectorsCount | кількість секторів, які необхідно відгадати |
| PassedSectorsCount | кількість вгаданих секторів |
| SectorsLeftToClose | кількість невгаданих секторів |
| MixedActions | історія введених відповідей |
| Messages | повідомлення адміністратора |
| Task | текст завдання |
| Sectors | сектора |
| Helps | підказки |
| Bonuses | додаткові завдання |

Перед відправкою відповідей перевіряємо:

• model.Level.IsPassed == false рівень не пройдений

• model.Level.Dismissed == false рівень не знято

• model.Level.HasAnswerBlockRule == false || model.Level.BlockDuration <= 0 - активна блокування вибрати номер відповідей на завдання (не діє для додаткових відповідей)

Якщо перераховані вище умови справедливі, то можна посилати відповіді на перевірку.

Відправлення відповідей на завдання рівня і додаткові завдання

|  |  |
| --- | --- |
| HTTPVERB | POST |
| URL | /GameEngines/Encounter/Play/{ID Гри}?json=1 |
| Params | LevelId = {Id рівня} (беремо з model.Level.LevelID)  LevelNumber = {Номер рівня} (беремо з model.Level.Number)  LevelAction.Answer {Текст відповіді} |

Результат останньої дії гравця зберігається в об'єкті model.EngineAction, який має такі властивості.

|  |  |
| --- | --- |
| GameId (int) | ID гри |
| LevelId (int) | ID Рівня |
| LevelNumber(int) | Номер рівня на який був введена відповідь |
| LevelAction (object) | інформація про результат відправки відповіді на рівень і бонус |
| BonusAction (object) | інформація про результат відправки відповіді на бонус |

Для того що б дізнатися, чи був введена відповідь правильною, необхідно перевірити властивість model.EngineAction.LevelAction:

|  |  |
| --- | --- |
| Answer (string) | Введена відповідь |
| IsCorrectAnswer (bool) | null - відповіді не було, false - неправильна відповідь; true - правильна відповідь; |

Таким чином, спочатку перевіряємо, чи був введена відповідь model.EngineAction.LevelAction.IsCorrect! = Null і тільки потім дивимося інші властивості.

* 1. Проектування інтерфейсу додатку
     1. Обґрунтування вибору концепції дизайну додатку

В останні декілька років прослідковується популяризація плаского дизайну в користувацьких інтерфейсах. Нові розробки компаній-гігантів яскраво це демонструють. До недавнього часу такі компанії як Apple, Microsoft, Google використовували філософію скевоморфізму в розробці дизайну пристроїв та інтерфейсів. Скевоморфізм – це практика імітації та наближення сприйняття віртуального об'єкта до його аналогу з матеріального світу.

25 червня 2014 року компанія Google на конференції «2014 Google I/O» анонсувала своє бачення плаского дизайну – Material Design. Більшість з нас взаємодіють з цифровим світом більшу частину життя. Для нас вже не настільки критично, щоб віртуальні кнопки були схожі на їх фізичні аналоги. І чим швидше до цього звикне більшість, тим швидше digital design зможе рухатися вперед.

Виходячи з цього, Material design вже не настільки схильний до впливу скевоморфізму і, скоріше, він став новим щаблем у розвитку мови візуальних образів і імітації реальності. На стику скевоморфізму і еволюції користувацького досвіду і з'явився Material design.

Поверхні і краї елементів в даному напрямку дизайну створюють візуальні образи і сигнали, які передають підказки і допомагають інтуїтивно орієнтуватися, як якщо б це відбувалося в реальному світі.

Використання знайомих тактильних характеристик і реалістичне освітлення допомагають користувачеві візуально відокремити головні об'єкти від другорядних, зрозуміти ставлення об'єкта до його оточенню і визначити його призначення.

Material design ґрунтується і на принципах друкованого дизайну. І не тільки для краси, а й для розстановки акцентів і фокусування уваги користувача на потрібному елементі, для спрощення навігації серед ієрархії конструкцій інтерфейсу, для інтуїтивної передачі їх сенсу.

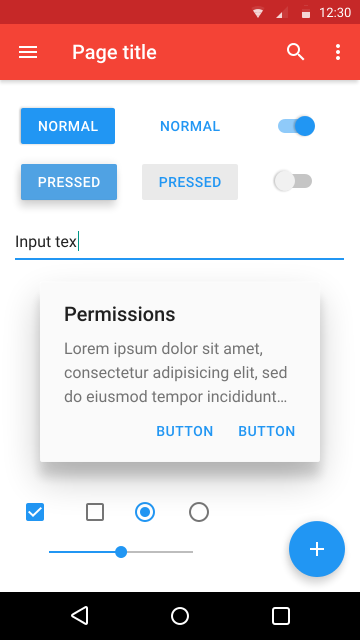


Рис. 4.1 Основні компоненти Material Design

Дослідження, зроблене дизайнерами Google, змінює підхід до інтерфейсу і робить його реальнішим.

* Material Design за допомогою системи управління тінями створює візуальну ілюзію простору між застосунком та екраном пристрою.
* Була збільшена реалістичність анімації за допомогою прискорення та пригальмовування рухів, пружного стрибання об'єктів.
* З'явилися інтерактивні іконки.
* З'явилася об'єктна хореографія, при якій елементи здатні впливати на сусідні елементи.
  + 1. Метод проектування згори донизу

Без сумніву, важлива умова успішного створення програм полягає в застосуванні надійних методів проектування. Широке поширення отримали наступні три методи: спадний (зверху вниз), висхідний (знизу вгору) і спеціальний (на даний конкретний випадок). У разі спадного методу ви починаєте творчий процес з програми високого рівня і спускаєтеся до підпрограм низького рівня. Висхідний метод працює в зворотному напрямку: ви починаєте з окремих спеціальних підпрограм, поступово будуєте на їх основі більш складні конструкції і закінчуєте самим верхнім рівнем програми. Спеціальний підхід не має заздалегідь встановленого методу.

Метод проектування системи зверху донизу є складовою частиною структурної розробки програмного продукту і допомагає детально спроектувати систему. Як і для будь-якої загальної схеми, при застосуванні спадного методу починають з загального опису програми, а потім переходять до опрацювання її конкретних елементів. На практиці при розробці програми найкраще спочатку точно визначити, що програма буде робити на найвищому рівні, і тільки після цього занурюватись в подробиці, що стосуються кожної дії.

Спочатку необхідно реалізувати основний інтерфейс, в якому, замість кожного зв'язкового логічного фрагмента тексту, вставляється виклик підпрограми, яка буде виконувати цей фрагмент. Для початку замість справжніх підпрограм, в програму вставляються фіктивні частини – заглушки. Заглушки представляють собою JSON-файли, що є аналогами відповідей сервера на запити. Потім заглушки замінюються або доопрацьовуються до справжніх повнофункціональних фрагментів із використанням API. На кожній стадії процесу реалізації вже створена програма повинна правильно працювати по відношенню до більш низького рівня.

* 1. Розробка програмного забезпечення

Систему розроблену у даній магістерській дисертації проекті було реалізовано у вигляді додатку для мобільних пристроїв на базі ОС Android та iOS.

Розроблено модулі для зв’язку з сервером за допомогою API, реалізовано інтерфейс в стилі Material Design.

Основними функціями цього рівня є відображення інформації та взаємодія з користувачем, а саме надання йому доступу до тексту завдання, секторів, бонусів, підказок та інших частин ігрового рівня

Мобільний додаток зв’язується з хмарним сервером і використовує бібліотеку axios, тому обмін між сервером та клієнтом здійснюється за допомогою окремого прошарку, реалізованого у вигляді API функцій, який є другим рівнем додатку. Цей рівень реалізований за допомогою технологій:

* async/await;
* axios;
* Promises

Основними функціями цього рівня є забезпечення обміну даними між клієнтом та сервером, обробка запитів, які викликає клієнт та опрацьовування результатів у форматі JSON.

* 1. Розробка інтерфейсу користувача

На початку роботи з додатком потрібно авторизуватись. Після авторизації користувач потрапляє на сторінку гри, де бачить усі основні складові ігрового рівня.

Елементи навігації:

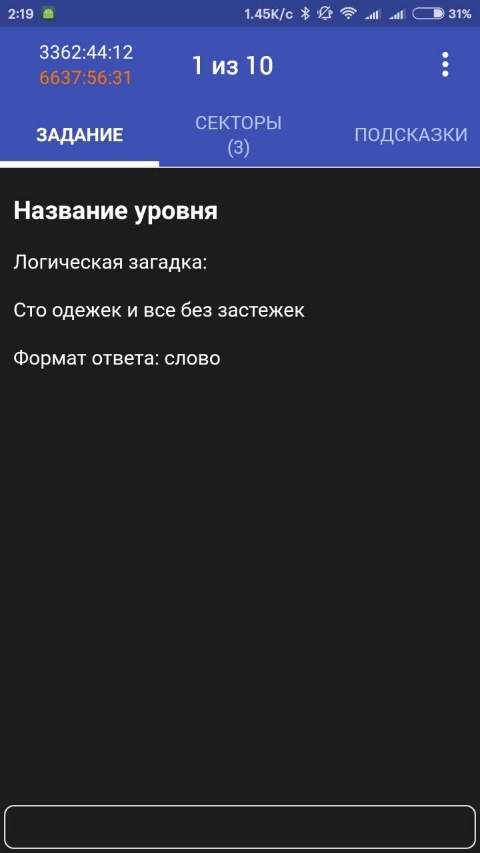


Рис. 5.1 Завдання рівня

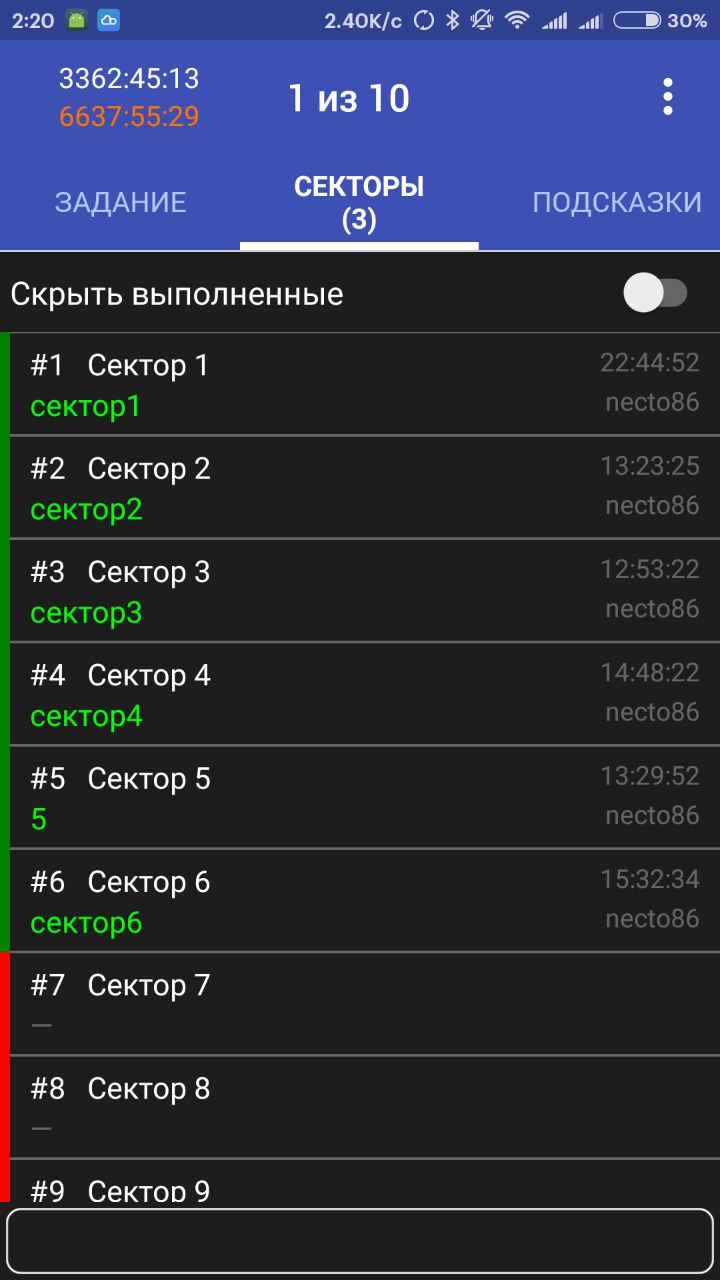
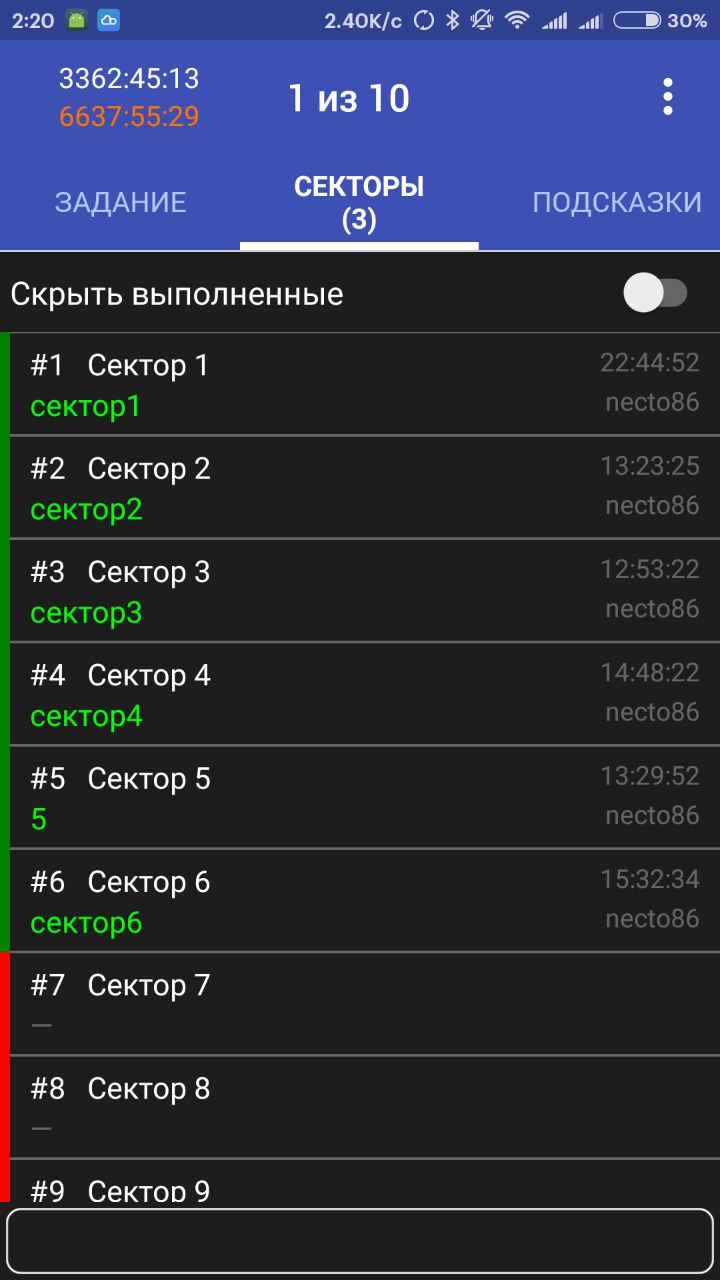


Рис. 5.2 Сектори рівня

У додатку реалізована функція приховування виконаних секторів.



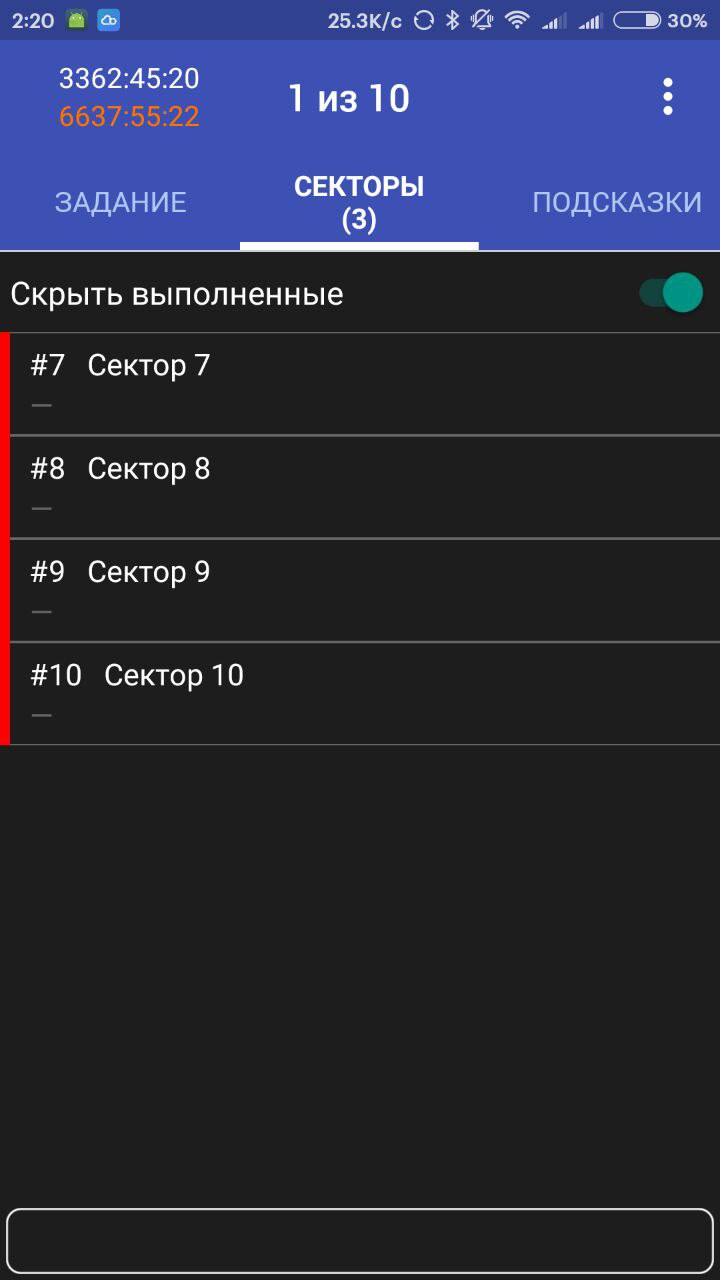


Рис. 5.3 Приховані та видимі виконані сектори

При натисканні на спеціальний чек бокс виконані сектори можуть бути як видимі, так і приховані.

Елементи навігації та керування:

* Секція “Завдання” (рис 5.1);
* Секція “Сектори” (рис 5.2);
* Секція “Підказки” (рис 5.4);
* Секція “Бонуси” (рис 5.5);

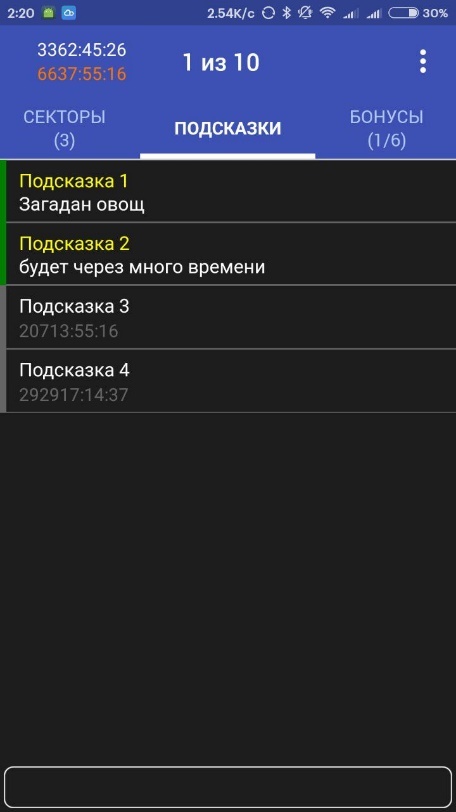


Рис. 5.4

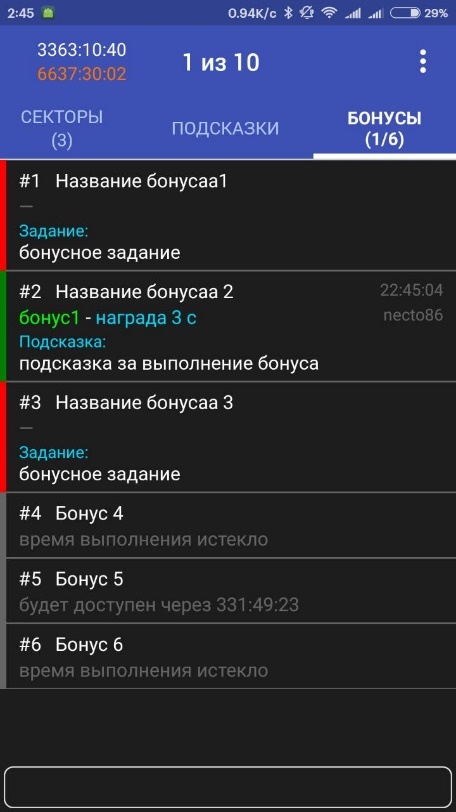


Рис. 5.5

Також у додатку є можливість введення відповідей. Додаток відрізняє коли введена вірна відповідь (рис 5.7), а коли – невірна (рис 5.6).

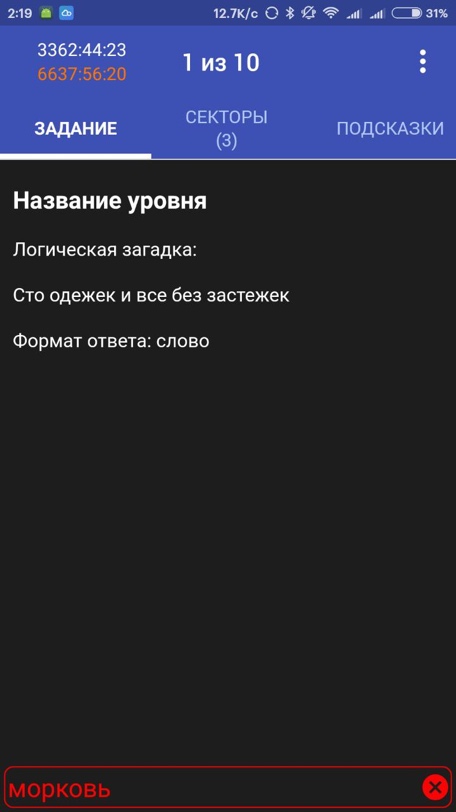


Рис. 5.6

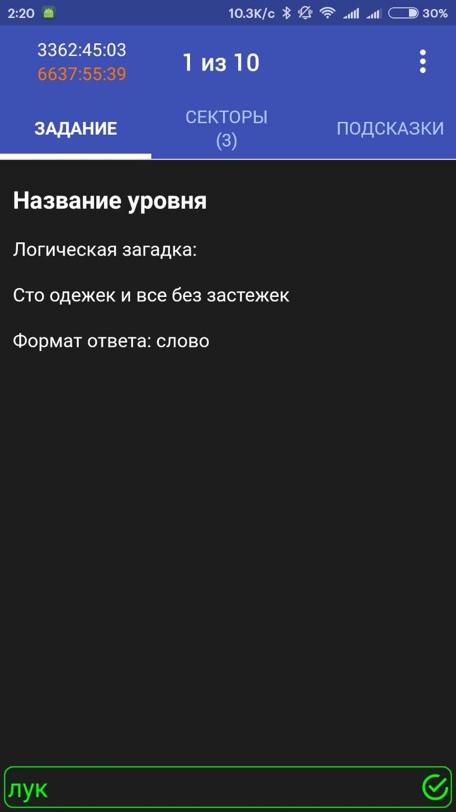


Рис. 5.7

ВИСНОВОК ДО РОЗДІЛУ

В даному розділі було розроблену модульну модель додатку для взаємодії користувача із ігровою платформою. До неї увійшли основні модулі, що пов’язані з прикладною проблемою:

* Модуль автентифікації та авторизації, який керує сесіями додатку та користувачів.
* Модуль самої гри, який відповідає за усе, що пов’язано з моделлю гри та взаємодією користувача.

Також в даному розділі було обрано концепцію побудови інтерфейсу додатку, що включає в себе застосування парадигми Material Design та методу проектування згори до низу.

Виконана розробка мобільного додатку з використанням усіх заявлених в ТЗ технологій.

Також в даному розділі було описано та проаналізовано технології та інструменти, що використовувались для розробки даного магістерської дисертації. Було обрано мову програмування для розробки додатку, потрібні фреймворки та технології. Описані їхні переваги та принцип роботи.

Також було обрано середовище розробки WebStorm через вагомі переваги перед аналогами.

РОЗДІЛ 4

1. РОЗРОБКА СТАРТАП - ПРОЕКТУ

У даному розділі буде проведено аналіз та розробка стартап-проекту відповідно до дисертації.

* 1. Опис ідеї

У таблиці наведено зміст, напрямки та вигоди для користувача даного додатку.

|  |  |  |
| --- | --- | --- |
| *Зміст ідеї* | *Напрямки застосування* | *Вигоди до користувача* |
| Створення крос-платформного клієнта для платформи активних міських ігор | Сфера розваг, логічні ігри | Більш зручніша взаємодія із ігровою системою |

Визначення слабких та сильних сторін додатку.

У даному розділі буде виконано аналіз та розробку стартап-проекту відповідно до теми дисертації.

Наступним кроком стане визначення та аналіз техніко-економічних можливостей та характеристик ідеї. Так як конкурентів знайдено не було, то буде представлений лише перелік техніко-економічних характеристик даної ідеї (табл 4.2).

* 1. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

|  |  |  |
| --- | --- | --- |
| *№* | *Техніко-економічні характеристики ідеї* | *Даний*  *проект* |
| 1 | Крос-платформний додаток | + |
| 2 | Не потребує багато ресурсів системи | + |
| 3 | Безкоштовне застосування | + |
| 4 | Можливість запустити на багатьох різноманітних платформах | - |

* 1. Технологічний аудит ідеї проекту

Необхідно провести аудит технологій, за допомогою яких буде реалізовано додаток. Результати цього аудиту наведені нижче.

* 1. Технологічна здійсненність ідеї проекту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Ідея проекту* | *Технології* | *Наявність*  *технологій* | *Доступність*  *технологій* |
| 1 | Створення  Крос-платформного  Додатку  для платформи  активних  міських ігор | JavaScript | Наявна | У відкритому |
|  |  |  | доступі |
| 2 | React Native | Наявна | У відкритому |
|  |  |  | доступі |
|  |  |  |  |
| 3 | Material UI | Наявна | У відкритому |
|  |  |  | доступі |
|  |  |  |  |
| 4 | MobX | Наявна | У відкритому |
|  |  |  | доступі |
|  |  |  |  |
| 5 | Axios | Наявна | У відкритому |
|  |  |  | доступі |
|  |  |  |  |

Як видно з результатів, усі технології для реалізації є в OpenSource доступі .

* 1. Аналіз ринкових можливостей запуску стартап-проекту

Наступним кроком стане аналіз ринку

* 1. Попередня характеристика потенційного ринку стартап-проекту

|  |  |  |
| --- | --- | --- |
| *№* | *Показники стану ринку* | *Характеристика* |
| 1 | Кількість головних гравців, од | 0-1 |
| 2 | Загальний обсяг продаж, грн/ум.од | Середній дохід від реклами в додатку, після досягання 1 тисячі завантажень складає 1,25 тисяч грн.  / $50 |
| 3 | Динаміка ринку | Постійно зростає |
| 4 | Наявність обмежень для входу | Ніяких |

У таблиці нижче наведено потенційні групи клієнтів, їх потреби та вимоги до продукту.

* 1. Характеристика потенційних клієнтів стартап-проекту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Потреба, що формує ринок* | *Цільова аудиторія* | *Відмінності у поведінці різних потенційних цільових груп* | *Вимоги споживачів до товару* |
| 1 | Потреба у  різноманітних квестах, логічних іграх, різних квізах | Всі прошарки населення, що мають смартфони Android/iOS та  доступ до мережі Інтернет | Даний застосунок поки що можна використовувати лише з одною платформою | Гарантія точного та швидкого здобуття інформації про поточну гру |

Після визначення потенційних груп клієнтів проведено аналіз ринкового середовища на фактори загроз та можливостей

* 1. Фактори загроз

|  |  |  |  |
| --- | --- | --- | --- |
| *№* | *Фактор* | *Зміст загрози* | *Можлива реакція компанії* |
| 1 | Низький прибуток від реклами у додатку | На старті при наявній малій аудиторії користувачів процес  інтегрування може лякати існуючих користувачів та потенційних нових. | Дуже довго не інтегрувати рекламу, щоб набрати велику кількість користувачів. Після цього інтегрувати невелику кількість реклами або ж створити певні покращення, які розблоковуються після оплати. |
| 2 | Критичні недоліки | bugs | Швидке здобуття відгуку від користувачів, та подальший фікс bugs |

* 1. Фактори можливостей

|  |  |  |  |
| --- | --- | --- | --- |
| *№* | *Фактор* | *Зміст можливості* | *Можлива реакція компанії* |
| 1 | Залучення інвестицій | Можливість залучення інвестицій від платформи, для якої розроблюється додаток | Реалізація більш глибокого API |
| 2 | Відсутність конкурентів | На момент розробки аналогів та конкурентів знайдено не було | Першість в даній галузі відкриває безліч можливостей наприклад здобуття великої долі ринка |

В даній таблиці можна побачити, що конкуренція на міжнародному ринку відсутня.

*Таблиця 4.8.*

* 1. Ступеневий аналіз конкуренції на ринку

|  |  |  |
| --- | --- | --- |
| *Особливості конкурентного середовища* | *В чому проявляється дана характеристика* | *Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)* |
| 1. Вказати тип конкуренції: чиста | Відсутність великих гравців на ринку або відсутність конкурентів взагалі | Продовження розвивання ідеї, додавання нових можливостей та покращення вже існуючих |
| 2. За рівнем конкурентної боротьби: міжнаціональна | Потенційні компанії- конкуренти можуть працювати на міжнародному ринку | Необхідність працювати на світовому ринку |
| 3. За галузевою ознакою: внутрішньогалузева | Постачальники зовсім не займаються подібним ПО | Сконцентруватися на ринку активних міських ігор |
| 4. Конкуренція за видами товарів: товарно-видова | Невелика кількість авторитетних постачальників | Здобувати авторитет прогресивним розвитком системи |
| 5. За характером конкурентних переваг: інноваційна, якісна | Великі відмінності в якості із схожими продуктами, а також їхня новизна | Забезпечити конкурентну прогресування та якості продукту |
| 6. За інтенсивністю: марочна | Постачальники виступають під певним брендом платформи | Створення чіткого бренду платформи |

Серед наявних прямих конкурентів на ринку конкуренції знайдено не було.

* 1. Аналіз конкуренції в галузі за М. Портером

|  |  |  |  |
| --- | --- | --- | --- |
| *Складові*  *аналізу* | *Потенційні*  *конкуренти* | *Клієнти* | *Замінники* |
|  | *в галузі* |  |  |
|  | Dozor, Qenq, | Приватні особи | Достатня кількість подібних |
| *Висновки* | Вихід на ринок можливий за | Клієнти не | Відсутність |
|  | умови надання конкурен- | висувають вимоги | локальних |
|  | тоздатних умов | до виробників | товарів- |
|  |  | оскільки | замінників |
|  |  | конкуренція |  |
|  |  | відсутня |  |

Робота на ринку можлива та сповнена можливостей.

* 1. Обґрунтування факторів конкурентоспроможності

|  |  |  |
| --- | --- | --- |
| *№* | *Фактор конкурентоспроможності* | *Обґрунтування* |
| 1 | Використання в основі відомих продуктів | Більше довіри до функціоналу створеної системи |
| 2 | Мобільність | Використання клієнту на ОС Android або iOS не прив’язує користувачів до ПК |

|  |  |  |
| --- | --- | --- |
| *1* | *2* | *3* |
| 3 | Першість на ринку | Аналогічних додатків з таким же функціоналом у широкому доступі знайдено не було |

Наступним кроком стане порівняльний аналіз сильних та слабких сторін продукту у порівнянні з конкурентами.

* 1. Порівняльний аналіз сильних та слабких сторін

|  |  |  |
| --- | --- | --- |
| *№* | *Фактор конкурентоспроможності* | *Бали 1-20* |
| 1 | Використання в основі відомих продуктів | 15 |
| 2 | Мобільність | 18 |
| 3 | Першість на ринку | 15 |

Фінальним кроком є SWOT-аналіз

* 1. SWOT-аналіз стартап-проекту

|  |  |
| --- | --- |
| Сильні сторони: Мобільність | Слабкі сторони: недостатньо розвинута система бо є обмеження з боку платформи активних міських ігор |
| Можливості: розвитку продукту впродовж певного часу без конкуренції та подальше його  Масштабування та розповсюдження | Загрози: потенційні конкуренти можуть з’явитися |

На основі аналізу розроблено альтернативи ринкової поведінки та перелік заходів з їх впровадження

* 1. Альтернативи ринкового впровадження стартап-проекту

|  |  |  |  |
| --- | --- | --- | --- |
| *№* | *Альтернатива ринкової поведінки* | *Ймовірність отримання ресурсів* | *Строки реалізації* |
| 1 | Зацікавлення інших гравців | Продаж проекту чи його  поглинання самою платформою | Декілька років |
| 2 | Переведення продукту на модель opensource | Можливість безкоштовного залучення програмістів | До місяця |
| 3 | Додавання реклами чи платних підписок | Можливість почати заробляти на клієнтах даної платформи | Пару місяців |

ВИСНОВОК

В результаті дисертації була досягнена поставлена мета, а саме, розроблено крос-платформного клієнтського програмного забезпечення для взаємодії із ігровою платформою.

Для досягнення цієї мети були поставлені наступні основні задачі: опис предметної області, аналіз існуючих рішень, аналіз технологій розробки додатків для взаємодії користувача із ігровою системою, проектування та розробка клієнтської частини додатку для взаємодії користувача із ігровою системою.

На основі аналізу предметної області об’єкту дослідження дисертації, було поставлено наступні задачі:

* Реалізувати якомога зручніший та швидкий процес отримання даних гри використовуючи API;
* Можливість відправлення кодів на сервер;
* Забезпечити можливість роботи без підключення до інтернету;
* Зробити роботу додатку стабільною навіть на «слабких» за своїми технічними характеристиками пристроях;

Було обрано наступні технології та методи для оптимальної реалізації поставлених задач:

* Мова програмування JavaScript, бо вона є стандартом для розробки нативних додатків для платформи React Native;
* React Native – крос-платформний фрейворк для побудови нативних додатків на Android та iOS;
* Axios – бібліотека для створення HTTP - запитів до серверу.
* При створенні дизайну додатку використовувалися положення парадигми Material Design, в якості методології розробки було обрано метод проектування згори донизу;

Наукова новизна дисертації полягає у створенні спосіб взаємодії із платформою активних міських ігор, який базується саме на додатку для мобільної операційної системи є більш відомий для користувача. На відміну від інших існуючих рішень цей спосіб сильно спрощує користування платформою активних міських ігор. Аналогів з таким функціоналом не було знайдено у маркетах (магазинах додатків) або у інших місцях.

Практична цінність полягає у розробці нового крос-платформного програмного забезпечення, що дозволяє, на відміну від інших існуючих рішень, більш простіше взаємодіяти із платформою активних міських ігор. Це підвищує ефективність гравців до спрощує досвід користувача.

ПЕРЕЛІК ПОСИЛАНЬ

1. «Android for developers». Офіційна документація Android. [Електронний ресурс] : Офіційна документація Android – Режим доступу : https://developer.android.com
2. «Materialize – modern responsive front-end framework based on Material Design». Офіційна документація фреймворку Materialize. [Електронний ресурс] : Офіційна документація фреймворку Materialize – Режим доступу: http://materializecss.com/
3. «React Native. Learn once, write anywhere: Build mobile apps with React». Офіційна документація фреймворку React Native. [Електронний ресурс] : Офіційна документація фреймворку React Native – Режим доступу: https://facebook.github.io/react-native/
4. «Native Base. Essential cross-platform UI components for React Native». Офіційна документація фреймворку Native Base. [Електронний ресурс] : Офіційна документація фреймворку Native Base – Режим доступу: https://nativebase.io/
5. «Axios». Офіційна документація бібліотеки Axios». [Електронний ресурс] : Офіційна документація бібліотеки Axios – Режим доступу: <https://github.com/mzabriskie/axios>
6. «Learning React Native». Книга». [Електронний ресурс] : Learning React Native». Книга – Режим доступу: <http://shop.oreilly.com/product/0636920085270.do>
7. «ProTech Professional Technical Services, Inc.». [Електронний ресурс] : Course summary – Режим доступу: <https://www.protechtraining.com/pdf/PT20726~.pdf>
8. «Evaluation Targeting React Native in Comparison to Native Mobile Development». [Електронний ресурс]– Режим доступу: <http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8886469&fileOId=8886473>
9. «React Native с точки зрения мобильного разработчика». https:/harb.com». [Електронний ресурс]– Режим доступу: <https://habr.com/ru/company/qlean/blog/416097/>
10. «Flutter». Офіційна документація фреймворку Flutter». [Електронний ресурс] : Офіційна документація фреймворку Flutter – Режим доступу: <https://flutter.dev/>
11. «Comparison of cross-platform mobile development tools». Офіційна документація бібліотеки Axios». [Електронний ресурс] - Режим доступу: <https://ieeexplore.ieee.org/abstract/document/6376023>
12. «Cross-Platform Mobile Development: Challenges and Opportunities».». [Електронний ресурс] :– Режим доступу: <https://link.springer.com/chapter/10.1007/978-3-319-01466-1_21>
13. «ProTech Professional Technical Services, Inc.». [Електронний ресурс] : Course summary – Режим доступу: <https://www.protechtraining.com/pdf/PT20726~.pdf>
14. «A comparative analysis of cross-platform development approaches for mobile applications». [Електронний ресурс] – Режим доступу: <https://dl.acm.org/citation.cfm?id=2490292>
15. «Cross-Platform Mobile Development: A Study on Apps with Animations». Офіційна документація бібліотеки Axios». – Режим доступу: <https://www.math.unipd.it/~mciman/pubb/SAC14.pdf>
16. «Отличительные особенности гибридной, кросплатформенной и нативной разработки приложения». [Електронний ресурс] – Режим доступу: <https://elibrary.ru/item.asp?id=29940398>
17. «React Native application development». [Електронний ресурс] – Режим доступу: <http://www.diva-portal.org/smash/get/diva2:998793/FULLTEXT02.pdf>
18. «React Native application development: A comparison between native Android and React Native». [Електронний ресурс] – Режим доступу: <http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A998793&dswid=4049>
19. «ProTech Professional Technical Services, Inc». [Електронний ресурс] – Режим доступу: <https://www.protechtraining.com/pdf/PT20726~.pdf>
20. «A comparative analysis of cross-platform development approaches for mobile applications». [Електронний ресурс] – Режим доступу: <https://dl.acm.org/citation.cfm?id=2490292>

ЛІСТИНГ ДОДАТКУ

**import** { action, observable } **from** 'mobx';  
**import** BackgroundTimer **from** 'react-native-background-timer';  
**import** onGlobalTimerTick **from** '../../core/events/onGlobalTimerTick';  
  
**import** API **from** '../../util/API';  
  
  
**class** GameStore {  
 @observable gameModel = {};  
 @observable globalTimerCounter = 0;  
 @observable isRefreshing = **false**;  
 @observable actualCode = '';  
 @observable lastUpdateTimestamp = Date.now();  
  
 globalTimer = **null**;  
  
 @action updateGameModel = **async** (requestData) => {  
 **if** (**this**.isRefreshing) **return**;  
  
 **this**.isRefreshing = **true**;  
  
 **try** {  
 **this**.gameModel = **await** API.*getGameModal*(requestData);  
 **this**.isRefreshing = **false**;  
 } **catch** (e) {  
 **this**.isRefreshing = **false**;  
 **return**;  
 }  
  
 **this**.lastUpdateTimestamp = Date.now();  
 **this**.globalTimerCounter = 0;  
  
 **if** (**this**.globalTimer) BackgroundTimer.clearInterval(**this**.globalTimer);  
  
 **this**.globalTimer = BackgroundTimer.setInterval(() => {  
 onGlobalTimerTick();  
 }, 1000);  
 };  
  
 @action sendCode = **async** () => {  
 **const** requestData = {  
 LevelId: **this**.gameModel.Level && **this**.gameModel.Level.LevelId,  
 LevelNumber: **this**.gameModel.Level && **this**.gameModel.Level.Number,  
 'LevelAction.Answer': **this**.actualCode,  
 };  
  
 **this**.updateGameModel(requestData);  
 };  
  
 @action changeActualCode = (code) => {  
 **this**.actualCode = code;  
 };  
}  
  
**export default new** GameStore();

**import** React, { Component } **from** 'react';  
**import** MainView **from** './views/MainView';  
  
**export default class** App **extends** Component {  
 render() {  
 **return** (  
 <MainView />  
 );  
 }  
}

**import** React, { Component } **from** 'react';  
**import** { View } **from** 'react-native';  
**import** { observer, Provider } **from** 'mobx-react/native';  
**import** gameStore **from** '../core/stores/gameStore';  
**import** GameView **from** './GameView';  
**import** LoadingView **from** './LoadingView';  
  
**class** MainView **extends** Component {  
  
 render() {  
 **const** { gameModel } = gameStore;  
  
 **return** (  
 <Provider gameStore={gameStore}>  
 {gameModel.Event === 0 ? <GameView /> : <LoadingView />}  
 </Provider>  
 );  
 }  
}  
  
**export default** observer(MainView);

**import** React **from** 'react';  
**import** { View } **from** 'react-native';  
**import** { observer, inject } **from** 'mobx-react/native';  
**import** { Button, Container, Header, Tab, Tabs, Text, Title, ScrollableTab, TabHeading, Icon, Badge } **from** 'native-base';  
  
**import** IconEntypo **from** 'react-native-vector-icons/Entypo';  
**import** IconMC **from** 'react-native-vector-icons/MaterialCommunityIcons';  
  
**import** API **from** '../util/API';  
**import** Helper **from** '../util/helper';  
  
**import** PushNotification **from** 'react-native-push-notification';  
  
**import** Colors **from** '../constants/colors';  
  
**import** CountableText **from** '../core/components/CountableText';  
  
**import** TaskSection **from** '../sections/TaskSection';  
**import** SectorsSection **from** '../sections/SectorsSection';  
**import** HintsSection **from** '../sections/HintsSection';  
**import** BonusesSection **from** '../sections/BonusesSection';  
**import** CodeSection **from** '../sections/CodeSection';  
  
  
**const** mapStateToProps = (stores => ({  
 globalTimerCounter: stores.gameStore.globalTimerCounter,  
 lastUpdateTimestamp: stores.gameStore.lastUpdateTimestamp,  
 Level: stores.gameStore.gameModel.Level,  
 Levels: stores.gameStore.gameModel.Levels,  
 Hints: stores.gameStore.gameModel.Level.Helps,  
 Bonuses: stores.gameStore.gameModel.Level.Bonuses,  
}));  
  
**const** GameView = ({ globalTimerCounter, lastUpdateTimestamp, Level, Levels, Hints, Bonuses }) => (  
 <Container>  
 <Header style={styles.headerStyle} hasTabs>  
 <View style={styles.timersContainer}>  
 <CountableText  
 increment  
 start={(lastUpdateTimestamp - Helper.*normalizeTime*(Level.StartTime.Value)) / 1000}  
 textStyle={{ color: Colors.white }}  
 />  
 {  
 Level.Timeout > 0 &&  
 <CountableText  
 start={Level.TimeoutSecondsRemain}  
 textStyle={{ color: Colors.upTime }}  
 />  
 }  
 </View>  
 <Title style={styles.levelNumber}>{`${Level.Number} из ${Levels.length}`}</Title>  
 <Button  
 transparent  
 onPress={() => {} }  
 style={styles.menuButton}  
 >  
 <IconEntypo style={{ fontSize: 20, color: 'white' }} name="dots-three-vertical" />  
 </Button>  
 </Header>  
 <Tabs  
 locked  
 renderTabBar={() => <ScrollableTab backgroundColor={Colors.tabBackground} />}  
 >  
 <Tab  
 heading={'ЗАДАНИЕ'}  
 textStyle={styles.tabText}  
 activeTextStyle={styles.tabText}  
 >  
 <TaskSection />  
 </Tab>  
 {  
 Level.SectorsLeftToClose > 0 &&  
 <Tab  
 heading={Helper.*formatWithNewLine*(['СЕКТОРЫ', `(${Level.SectorsLeftToClose})`])}  
 textStyle={styles.tabText}  
 activeTextStyle={styles.tabText}  
 >  
 <SectorsSection />  
 </Tab>  
 }  
 {  
 Hints.length > 0 &&  
 <Tab  
 heading={  
 Helper.*formatWithNewLine*(  
 [  
 'ПОДСКАЗКИ',  
 //Hints.find(hint => hint.RemainSeconds > 0) ? `(${Hints.find(hint => hint.RemainSeconds > 0).RemainSeconds})` : '',  
 ],  
 )  
 }  
 textStyle={styles.tabText}  
 activeTextStyle={styles.tabText}  
 >  
 <HintsSection />  
 </Tab>  
 }  
 {  
 Bonuses.length > 0 &&  
 <Tab  
 heading={  
 Helper.*formatWithNewLine*(  
 [  
 'БОНУСЫ',  
 `(${Bonuses.filter(bonus => bonus.IsAnswered).length}/${Bonuses.length})`,  
 ],  
 )  
 }  
 textStyle={styles.tabText}  
 activeTextStyle={styles.tabText}  
 >  
 <BonusesSection />  
 </Tab>  
 }  
 </Tabs>  
 <CodeSection />  
 </Container>  
);  
  
**const** styles = {  
 mainContainer: {  
 backgroundColor: Colors.background,  
 flex: 1,  
 padding: 7,  
 },  
  
 headerStyle: {  
 justifyContent: 'space-between',  
 alignItems: 'center',  
 },  
  
 timersContainer: {  
 flex: 1,  
 justifyContent: 'center',  
 alignItems: 'center',  
 },  
  
 levelNumber: {  
 flex: 1,  
 },  
  
 menuButton: {  
 flex: 1,  
 justifyContent: 'flex-end',  
 },  
  
 tabIcon: {  
 fontSize: 30,  
 color: Colors.white,  
 },  
  
 tabBadge: {  
 transform: [{  
 scale: 0.8,  
 }],  
 },  
  
 tabText: {  
 textAlign: 'center',  
 fontSize: 14,  
 },  
};  
  
**export default** inject(mapStateToProps)(observer(GameView));

**import** React **from** 'react';  
**import** { observer, inject } **from** 'mobx-react/native';  
**import** { ScrollView, Text, RefreshControl, View } **from** 'react-native';  
**import** HTMLView **from** '../core/components/HTMLView';  
**import** Colors **from** '../constants/colors';  
  
**import** AuthorMessage **from** '../gameComponents/AuthorMessage';  
  
**const** mapStateToProps = stores => ({  
 gameStore: stores.gameStore,  
 Level: stores.gameStore.gameModel.Level,  
});  
  
**const** TaskSections = ({ Level, gameStore: { isRefreshing, updateGameModel } }) => (  
 <ScrollView  
 style={styles.scroll}  
 contentContainerStyle={styles.scrollContent}  
 refreshControl={  
 <RefreshControl  
 refreshing={isRefreshing}  
 onRefresh={updateGameModel}  
 />  
 }  
 >  
 {  
 Level.Messages.length > 0  
 ? Level.Messages.map(message =>  
 <AuthorMessage  
 key={message.MessageId}  
 authorLogin={message.OwnerLogin}  
 messageText={message.MessageText}  
 replaceNl2Br={message.ReplaceNl2Br}  
 />)  
 : **null** }  
 <View  
 style={styles.mainContent}  
 >  
 {  
 Level.Name  
 ? <Text style={styles.levelName}>{Level.Name}</Text>  
 : **null** }  
 <View style={styles.taskWrapper}>  
 <HTMLView  
 html={Level.Tasks[0].TaskText}  
 shouldReplaceNlToBr={Level.Tasks[0].ReplaceNlToBr}  
 />  
 </View>  
 </View>  
 </ScrollView>  
);  
  
  
**const** styles = {  
 levelName: {  
 fontSize: 19,  
 color: Colors.white,  
 fontWeight: 'bold',  
 marginTop: 18,  
 },  
  
 scroll: {  
 backgroundColor: Colors.background,  
 },  
  
 scrollContent: {  
 backgroundColor: Colors.background,  
 },  
  
 mainContent: {  
 paddingHorizontal: 10,  
 },  
  
 taskWrapper: {  
 marginTop: 18,  
 },  
};  
  
**export default** inject(mapStateToProps)(observer(TaskSections));

**import** React, { Component } **from** 'react';  
**import** { observer, inject } **from** 'mobx-react/native';  
**import** { FlatList, View, Text } **from** 'react-native';  
**import** { Switch } **from** 'native-base';  
**import** Colors **from** '../constants/colors';  
**import** Sector **from** '../gameComponents/Sector';  
  
**const** mapStateToProps = stores => ({  
 gameStore: stores.gameStore,  
 sectors: stores.gameStore.gameModel.Level.Sectors,  
});  
  
**class** SectorsSection **extends** Component {  
 state = {  
 hideIsAnswered: **false**,  
 };  
  
 onChangeHideSwitch = (newValue) => {  
 **this**.setState({  
 hideIsAnswered: newValue,  
 });  
 };  
  
 render() {  
 **let** { sectors, gameStore: { isRefreshing, updateGameModel } } = **this**.props;  
  
 **if** (**this**.state.hideIsAnswered) {  
 sectors = sectors.filter(sector => !sector.IsAnswered);  
 }  
  
 **return** (  
 <View style={styles.mainContainer}>  
 <View style={styles.switchContainer}>  
 <Text style={styles.switchTitle}>{'Скрыть выполненные'}</Text>  
 <Switch  
 value={**this**.state.hideIsAnswered}  
 onValueChange={**this**.onChangeHideSwitch}  
 tintColor={Colors.gray}  
 />  
 </View>  
 <FlatList  
 data={sectors}  
 renderItem={({ item }) => (  
 <Sector  
 order={item.Order}  
 name={item.Name}  
 answerData={item.Answer}  
 isAnswered={item.IsAnswered}  
 />)  
 }  
 keyExtractor={sector => sector.SectorId}  
 refreshing={isRefreshing}  
 onRefresh={updateGameModel}  
 />  
 </View>  
 );  
 }  
}  
  
**const** styles = {  
 mainContainer: {  
 flex: 1,  
 backgroundColor: Colors.background,  
 },  
  
 switchContainer: {  
 flexDirection: 'row',  
 justifyContent: 'space-between',  
 alignItems: 'center',  
 height: 40,  
 paddingHorizontal: 5,  
 },  
  
 switchTitle: {  
 color: Colors.white,  
 fontSize: 16,  
 },  
};  
  
**export default** inject(mapStateToProps)(observer(SectorsSection));

**import** React **from** 'react';  
**import** { observer, inject } **from** 'mobx-react/native';  
**import** { FlatList } **from** 'react-native';  
**import** Colors **from** '../constants/colors';  
**import** Hint **from** '../gameComponents/Hint';  
  
**const** mapStateToProps = stores => ({  
 gameStore: stores.gameStore,  
 hints: stores.gameStore.gameModel.Level.Helps,  
});  
  
**const** HintsSection = ({ hints, gameStore: { isRefreshing, updateGameModel } }) => (  
 <FlatList  
 data={hints}  
 renderItem={({ item }) => (  
 <Hint  
 number={item.Number}  
 hintText={item.HelpText}  
 remainSeconds={item.RemainSeconds}  
 />)  
 }  
 keyExtractor={hint => hint.HelpId}  
 refreshing={isRefreshing}  
 onRefresh={updateGameModel}  
 style={styles.mainContainer}  
 />  
);  
  
**const** styles = {  
 mainContainer: {  
 flex: 1,  
 backgroundColor: Colors.background,  
 },  
};  
  
**export default** inject(mapStateToProps)(observer(HintsSection));

**import** React, { Component } **from** 'react';  
**import** { View, TextInput, Alert } **from** 'react-native';  
**import** { observer, inject } **from** 'mobx-react/native';  
**import** { Icon } **from** 'native-base';  
  
**import** Helper **from** '../util/helper';  
  
**import** Colors **from** '../constants/colors';  
  
**const** mapStateToProps = stores => ({  
 actualCode: stores.gameStore.actualCode,  
 changeActualCode: stores.gameStore.changeActualCode,  
 sendCode: stores.gameStore.sendCode,  
 oldCodes: stores.gameStore.gameModel.Level.MixedActions,  
 hasAnswerBlockRule: stores.gameStore.gameModel.Level.HasAnswerBlockRule,  
});  
  
**class** CodeSection **extends** Component {  
  
 confirmSendCode = () => {  
 **const** { sendCode } = **this**.props;  
  
 Alert.*alert*(  
 'На уровне ограничение на перебор!',  
 'Разрешить ввод старого неверного кода?',  
 [  
 { text: 'Отмена', onPress: () => {}, style: 'cancel' },  
 { text: 'Разрешить', onPress: () => sendCode() },  
 ],  
 );  
 };  
  
 render() {  
 **const** { actualCode, changeActualCode, sendCode, oldCodes, hasAnswerBlockRule } = **this**.props;  
  
 **const** oldCode = oldCodes.find(codeObject => Helper.*isEqualCode*(codeObject.Answer, actualCode));  
 **let** highlightColor;  
 **let** iconName;  
  
 **if** (oldCode) {  
 **if** (oldCode.IsCorrect) {  
 highlightColor = oldCode.Kind === 1 ? Colors.rightCode : Colors.bonus;  
 iconName = 'checkmark-circle';  
 } **else** {  
 highlightColor = Colors.wrongCode;  
 iconName = 'close-circle';  
 }  
 } **else** {  
 highlightColor = Colors.white;  
 }  
  
 **return** (  
 <View style={styles.mainContainer}>  
 <View style={[styles.inputWrapper, { borderColor: highlightColor }]}>  
 <TextInput  
 blurOnSubmit  
 selectTextOnFocus  
 autoCorrect={**false**}  
 underlineColorAndroid="transparent"  
 returnKeyType="send"  
 onChangeText={code => changeActualCode(code)}  
 onSubmitEditing={hasAnswerBlockRule && oldCode && !oldCode.IsCorrect ? **this**.confirmSendCode : sendCode}  
 value={actualCode}  
 style={[styles.codeInput, { color: highlightColor }]}  
 />  
 { oldCode && <Icon style={{ color: highlightColor, fontSize: 25 }} name={iconName} /> }  
 </View>  
 </View>  
 );  
 }  
}  
  
**const** styles = {  
 mainContainer: {  
 justifyContent: 'center',  
 alignItems: 'center',  
 backgroundColor: Colors.background,  
 padding: 3,  
 },  
  
 inputWrapper: {  
 flexDirection: 'row',  
 justifyContent: 'center',  
 alignItems: 'center',  
 padding: 2,  
 borderWidth: 1,  
 borderRadius: 7,  
 },  
  
 codeInput: {  
 flex: 1,  
 color: Colors.rightCode,  
 fontSize: 20,  
 padding: 0,  
 },  
};  
  
**export default** inject(mapStateToProps)(observer(CodeSection));

**import** React **from** 'react';  
**import** { observer, inject } **from** 'mobx-react/native';  
**import** { FlatList } **from** 'react-native';  
**import** Colors **from** '../constants/colors';  
**import** Bonus **from** '../gameComponents/Bonus';  
  
**const** mapStateToProps = stores => ({  
 gameStore: stores.gameStore,  
 bonuses: stores.gameStore.gameModel.Level.Bonuses,  
});  
  
**const** BonusesSection = ({ bonuses, gameStore: { isRefreshing, updateGameModel } }) => (  
 <FlatList  
 data={bonuses}  
 renderItem={({ item }) => (  
 <Bonus  
 number={item.Number}  
 name={item.Name}  
 task={item.Task}  
 isAnswered={item.IsAnswered}  
 answerData={item.Answer}  
 hint={item.Help}  
 awardTime={item.AwardTime}  
 expired={item.Expired}  
 secondsToStart={item.SecondsToStart}  
 secondsLeft={item.SecondsLeft}  
 />)  
 }  
 keyExtractor={bonus => bonus.BonusId}  
 refreshing={isRefreshing}  
 onRefresh={updateGameModel}  
 style={styles.mainContainer}  
 />  
);  
  
**const** styles = {  
 mainContainer: {  
 flex: 1,  
 backgroundColor: Colors.background,  
 },  
};  
  
**export default** inject(mapStateToProps)(observer(BonusesSection));

**import** React **from** 'react';  
**import** { observer } **from** 'mobx-react/native';  
**import** { Text, View } **from** 'react-native';  
  
**import** Helper **from** '../util/helper';  
  
**import** Colors **from** '../constants/colors';  
  
**const** Sector = ({ order, name, answerData, isAnswered }) => (  
 <View style={styles.mainContainer}>  
 <View  
 style={[  
 styles.coloredLabel,  
 { backgroundColor: isAnswered ? Colors.green : Colors.wrongCode },  
 ]}  
 />  
 <View style={styles.sectorContainer}>  
 <Text style={styles.sectorName}>{`#${order} ${name}`}</Text>  
 <Text  
 style={[  
 styles.sectorValue,  
 { color: isAnswered ? Colors.rightCode : Colors.gray },  
 ]}  
 >  
 { isAnswered ? answerData.Answer : '—'}  
 </Text>  
 </View>  
 { isAnswered &&  
 <View  
 style={[  
 styles.sectorContainer,  
 { alignItems: 'flex-end', flex: 1 },  
 ]}  
 >  
 <Text style={styles.sectorInfo}>{ Helper.*formatTime*(answerData.AnswerDateTime.Value) }</Text>  
 <Text style={[styles.sectorInfo, { marginTop: 2 }]}>{answerData.Login}</Text>  
 </View>  
 }  
 </View>  
);  
  
**const** styles = {  
 mainContainer: {  
 minHeight: 50,  
 flexDirection: 'row',  
 backgroundColor: Colors.background,  
 },  
  
 coloredLabel: {  
 width: 5,  
 backgroundColor: Colors.green,  
 },  
  
 sectorContainer: {  
 flex: 2,  
 paddingHorizontal: 10,  
 paddingVertical: 5,  
 borderColor: Colors.gray,  
 borderBottomWidth: 0.5,  
 borderTopWidth: 0.5,  
 },  
  
 sectorName: {  
 color: Colors.white,  
 fontFamily: 'Verdana',  
 fontSize: 15,  
 },  
  
 sectorValue: {  
 color: Colors.gray,  
 fontFamily: 'Verdana',  
 fontSize: 15,  
 },  
  
 sectorInfo: {  
 color: Colors.gray,  
 fontFamily: 'Verdana',  
 fontSize: 13,  
 },  
};  
  
**export default** observer(Sector);

**import** React **from** 'react';  
**import** { observer } **from** 'mobx-react/native';  
**import** { Text, View } **from** 'react-native';  
  
**import** Colors **from** '../constants/colors';  
  
**import** CountableText **from** '../core/components/CountableText';  
**import** HTMLView **from** '../core/components/HTMLView';  
  
**const** Hint = ({ number, hintText, remainSeconds }) => (  
 <View style={styles.mainContainer}>  
 <View  
 style={[  
 styles.coloredLabel,  
 { backgroundColor: remainSeconds === 0 ? Colors.green : Colors.gray },  
 ]}  
 />  
 <View style={styles.messageContainer}>  
 <Text  
 style={[  
 styles.authorLogin,  
 { color: remainSeconds === 0 ? Colors.yellow : Colors.white },  
 ]}  
 >  
 {`Подсказка ${number}`}  
 </Text>  
 { remainSeconds === 0  
 ? <HTMLView html={hintText} />  
 : <CountableText start={remainSeconds} textStyle={{ color: Colors.gray }} />  
 }  
 </View>  
 </View>  
);  
  
**const** styles = {  
 mainContainer: {  
 flexDirection: 'row',  
 backgroundColor: Colors.background,  
 },  
  
 coloredLabel: {  
 width: 5,  
 backgroundColor: Colors.green,  
 },  
  
 messageContainer: {  
 flex: 1,  
 paddingHorizontal: 10,  
 paddingVertical: 5,  
 borderColor: Colors.gray,  
 borderBottomWidth: 0.5,  
 borderTopWidth: 0.5,  
 },  
  
 authorLogin: {  
 fontFamily: 'Verdana',  
 fontSize: 15,  
 },  
};  
  
**export default** observer(Hint);

**import** React **from** 'react';  
**import** { observer } **from** 'mobx-react/native';  
**import** { Text, View } **from** 'react-native';  
  
**import** Helper **from** '../util/helper';  
  
**import** Colors **from** '../constants/colors';  
  
**import** HTMLView **from** '../core/components/HTMLView';  
**import** CountableText **from** '../core/components/CountableText';  
  
**const** Bonus = ({ number, name, task, isAnswered, answerData, hint, awardTime, expired, secondsToStart, secondsLeft }) => (  
 <View style={styles.mainContainer}>  
 <View  
 style={[  
 styles.coloredLabel,  
 { backgroundColor:  
 **do** {  
 **if** (expired || secondsToStart > 0) { Colors.gray }  
 **else if** (isAnswered) { Colors.green }  
 **else** { Colors.wrongCode }  
 },  
 },  
 ]}  
 />  
 <View style={styles.bonusContentWrapper}>  
 <View style={{ flexDirection: 'row' }}>  
 <View style={styles.bonusContainer}>  
 <Text style={styles.bonusName}>{`#${number} ${name || `Бонус ${number}`}`}</Text>  
 {secondsLeft > 0 && (  
 <View style={{ flexDirection: 'row' }}>  
 <Text style={styles.bonusValue}>{'осталось '}</Text>  
 <CountableText start={secondsLeft} textStyle={styles.bonusValue} />  
 </View>  
 )}  
 <View style={{ flexDirection: 'row' }}>  
 <Text  
 style={[  
 styles.bonusValue,  
 { color: isAnswered ? Colors.rightCode : Colors.gray },  
 ]}  
 >  
 {  
 **do** {  
 **if** (expired) { 'время выполнения истекло' }  
 **else if** (secondsToStart > 0) { 'будет доступен через ' }  
 **else if** (isAnswered) { answerData.Answer }  
 **else** { '—' }  
 }  
 }  
 </Text>  
 {secondsToStart > 0 && (  
 <CountableText start={secondsToStart} textStyle={styles.bonusValue} />  
 )}  
 {(isAnswered && awardTime > 0) && (  
 <Text style={[styles.bonusValue, { color: Colors.white }]}>  
 {' - '}  
 </Text>  
 )}  
 {(isAnswered && awardTime > 0) && (  
 <Text style={[styles.bonusValue, { color: Colors.bonus }]}>  
 {`награда ${Helper.*formatCount*(awardTime, { collapse: **true**, withUnits: **true** })}`}  
 </Text>  
 )}  
 </View>  
 </View>  
 { isAnswered && (  
 <View  
 style={[  
 styles.bonusContainer,  
 { alignItems: 'flex-end', flex: 1 },  
 ]}  
 >  
 <Text style={styles.bonusInfo}>{ Helper.*formatTime*(answerData.AnswerDateTime.Value) }</Text>  
 <Text style={[styles.bonusInfo, { marginTop: 2 }]}>{answerData.Login}</Text>  
 </View>  
 )}  
 </View>  
 { (!isAnswered && task) ? (  
 <View >  
 <Text style={styles.bonusTaskHintTitle}>{'Задание:'}</Text>  
 <HTMLView html={task} />  
 </View>  
 ) : **null** }  
 { (isAnswered && hint) ? (  
 <View >  
 <Text style={styles.bonusTaskHintTitle}>{'Подсказка:'}</Text>  
 <HTMLView html={hint} />  
 </View>  
 ) : **null** }  
 </View>  
 </View>  
);  
  
**const** styles = {  
 mainContainer: {  
 minHeight: 50,  
 flexDirection: 'row',  
 backgroundColor: Colors.background,  
 },  
  
 coloredLabel: {  
 width: 5,  
 backgroundColor: Colors.green,  
 },  
  
 bonusContentWrapper: {  
 flex: 1,  
 paddingHorizontal: 10,  
 paddingVertical: 5,  
 borderColor: Colors.gray,  
 borderBottomWidth: 0.5,  
 borderTopWidth: 0.5,  
 },  
  
 bonusContainer: {  
 flex: 2,  
 },  
  
 bonusName: {  
 color: Colors.white,  
 fontFamily: 'Verdana',  
 fontSize: 15,  
 },  
  
 bonusValue: {  
 color: Colors.gray,  
 fontFamily: 'Verdana',  
 fontSize: 15,  
 },  
  
 bonusInfo: {  
 color: Colors.gray,  
 fontFamily: 'Verdana',  
 fontSize: 13,  
 },  
  
 bonusTaskHintTitle: {  
 color: Colors.bonus,  
 fontFamily: 'Verdana',  
 fontSize: 13,  
 },  
};  
  
**export default** observer(Bonus);

**import** React **from** 'react';  
**import** { observer } **from** 'mobx-react/native';  
**import** { Text, View } **from** 'react-native';  
  
**import** Colors **from** '../constants/colors';  
  
**import** HTMLView **from** '../core/components/HTMLView';  
  
**const** AuthorMessage = ({ authorLogin, messageText, replaceNl2Br }) => (  
 <View style={styles.mainContainer}>  
 <View  
 style={styles.coloredLabel}  
 />  
 <View style={styles.messageContainer}>  
 <Text  
 style={styles.authorLogin}  
 >  
 {`${authorLogin}:`}  
 </Text>  
 <HTMLView html={messageText} shouldReplaceNlToBr={replaceNl2Br} />  
 </View>  
 </View>  
);  
  
**const** styles = {  
 mainContainer: {  
 flexDirection: 'row',  
 backgroundColor: Colors.background,  
 },  
  
 coloredLabel: {  
 width: 5,  
 backgroundColor: Colors.green,  
 },  
  
 messageContainer: {  
 flex: 1,  
 paddingHorizontal: 10,  
 paddingVertical: 5,  
 borderColor: Colors.gray,  
 borderBottomWidth: 0.5,  
 borderTopWidth: 0.5,  
 },  
  
 authorLogin: {  
 fontFamily: 'Verdana',  
 fontSize: 15,  
 color: Colors.gray,  
 },  
};  
  
**export default** observer(AuthorMessage);