МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕУЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Направление подготовки/специальность
09.03.01 Информатика и вычислительная техника

направленность (профиль)/специализация
«Технологии разработки программного обеспечения»

Выпускная квалификационная работа

Разработка мобильного приложения «Игротека вожатого»

Обучающегося 4 курса
очной формы обучения
Мельниковой Татьяны Владимировны

Руководитель выпускной квалификационной работы:
кандидат педагогических наук, доцент
Гончарова Светлана Викторовна

Рецензент:
Ученая степень *(при наличии)*, ученое звание *(при наличии)*, должность
Ф. И. О. *(указывается в именительном падеже)*

Санкт-Петербург
2022

Содержание

ВВЕДЕНИЕ

В настоящее время большое внимание уделяется воспитанию детей, так как для их всестороннего развития, необходима деятельность разного рода. В учебное время они ходят в секции и школы искусств, а летом могут поехать в лагерь, чтобы не заскучать дома и не провести это время года, сидя в компьютере или телефоне.

В Санкт-Петербург свыше полторы тысячи человек каждый год выезжает работать вожатым в лагерь. В том числе бойцы студенческого педагогического отряда “Маэстро” (далее СПО “Маэстро”), который основан на базе Герценовского университета. Основной деятельностью отряда в лагере является организация досуга детей, которые хотят проводить каждый день ярко и красочно. В этом вожатым помогают различные игры.

СПО «Маэстро» имеет большое количество активностей в своей игротеке, методические наработки и коллекцию мастер классов, которые хранятся на Google Drive. У этого вида хранения есть существенные минусы:

- довольно сложно подобрать игру, которая тебя интересует, так как для того, чтобы узнать её детали, необходимо открывать каждый документ, смотреть возраст играющих, реквизиты и т.п.;
- необходим доступ к интернету, чтобы открыть документы, но в лагерях Ленинградской области не всегда есть связь.

Поэтому было принято решение разработать мобильное приложение, в котором будут храниться все нужные материалы, и которое будет содержать возможности фильтра и выбора необходимых параметров.

Актуальность определена заказом СПО “Маэстро” мобильного приложения “Игротека вожатого”. Данный проект сократит время выбора подходящей игры в несколько раз.

Цель работы: разработать мобильное приложение “Игротека вожатого”.

Для достижения поставленной цели необходимо решить следующие задачи:

- провести анализ имеющихся решений;
- рассмотреть особенности создаваемого приложения, разработать структуру;
- выбрать метод разработки программного продукта;
- разработать продукт, учитывая риски;
- провести юзабилити-тестирование.

Объект исследования – мобильная разработка**.**

Предметом исследования является мобильное приложение “Игротека вожатого”.

Работа состоит из введения, первой главы, которая включает в себя рассмотрение имеющихся онлайн приложений игротеки вожатого и выбор технологии и инструментов разработки, второй главы, которая описывает практическую реализацию программного продукта, заключения и списка источников.

При анализе различных источников информации предпочтение отдано книгам и статьям, посвященным методам разработки программного обеспечения, особенностям мобильной разработки и .

ГЛАВА 1 Анализ проблемы и выбор технологии и инструментов

1.1. Обзор имеющихся онлайн приложений игротеки вожатого и поиск новых подходов

В Санкт-Петербурге работают 42 педагогических отряда, работать летом вожатыми выезжают более трех тысяч студентов, в том числе СПО «Маэстро». Большую часть времени вожатого занимает организация культурно-досуговой деятельности детей, для этого необходимо знать множество игр.

Игра – вид непродуктивной деятельности, где мотив лежит не в результате её, а в самом процессе. Поэтому очень удобно пользоваться готовыми игротеками, которые хранят множество необходимой информации. Можно выделить следующие особенности использования игр.

Во-первых, необходимо учитывать период смены и его особенности. Всего их существует три: организационный период (первые 3-4 дня смены), основной и заключительный (последние 3-4 смены). Организационный начинается, когда дети только приехали в лагерь, и конечно первым делом им нужно проводить игры на знакомство. В основной период хорошо подойдут большие отрядные дела, игры на сплочение. Ближе к заключительному периоду необходимо проводить игры на доверие, а также устраивать мероприятия, подводящие детей к тому, что лагерная смена скоро закончится. Также периоды связаны с кризисными точками смены. Например, в середине смены детям уже начинает надоедать лагерная жизнь, поэтому в этот момент важно провести какое-то очень яркое мероприятие, а также поиграть в игры на снятие эмоционального напряжения, так как нередки межличностные конфликты.

Во-вторых, важно учитывать возраст детей, их особенности. Детям 6-9 лет необходимо постоянно делать что-то новое, им не нравится монотонная работа. С ребятами постарше можно проводить масштабные игры, но обязательно опираясь на их интересы и желания, в отличие от младших детей, их невозможно заставить играть против их воли.

В-третьих, одним из главных элементов хорошего дня является наличие смены деятельность, то есть в день должно быть что-то спортивное, интеллектуальное и творческое. Не обязательно, чтобы это были большие мероприятия, но наличие хотя небольших игр, связанных с разными видами деятельность, необходимо. Так дети могут проявить себя, и им не наскучит лагерь.

Все перечисленные особенности должны быть учтены при проектировании мобильного приложения игротеки и найти свое отражение в поиске и фильтрах поиска. Поэтому необходимы фильтры: возраст, тип игры, количество участников.

Рассмотрим существующие онлайн сервисы и приложения.

Сайт «явожатый.рф» (рисунок 1.1).

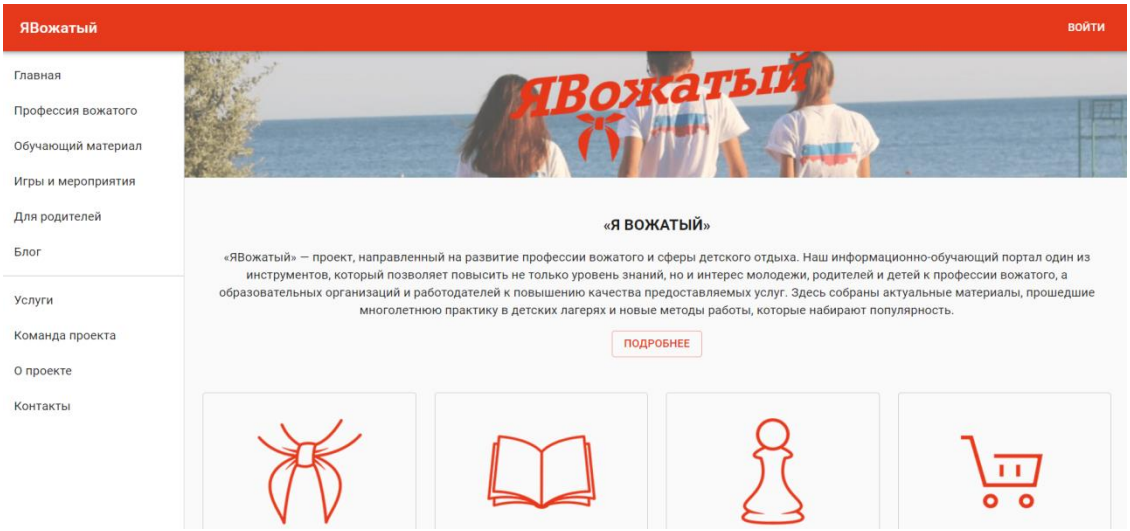

Рисунок 1.1 – явожатый.рф

Преимущества:

- Можно выбрать игру по параметрам: возраст, вид игры, период и участники.
- Понятный интерфейс.

Недостатки:

- Всего шесть категорий игр: на знакомство, на сплочение, развивающие, на выявление лидера, подвижные, интеллектуальные.
- Маленькая база знаний, менее 50 игр.
- Для доступа к играм необходим интернет.

Сайт «mosgortur.ru» (рисунок 1.2).

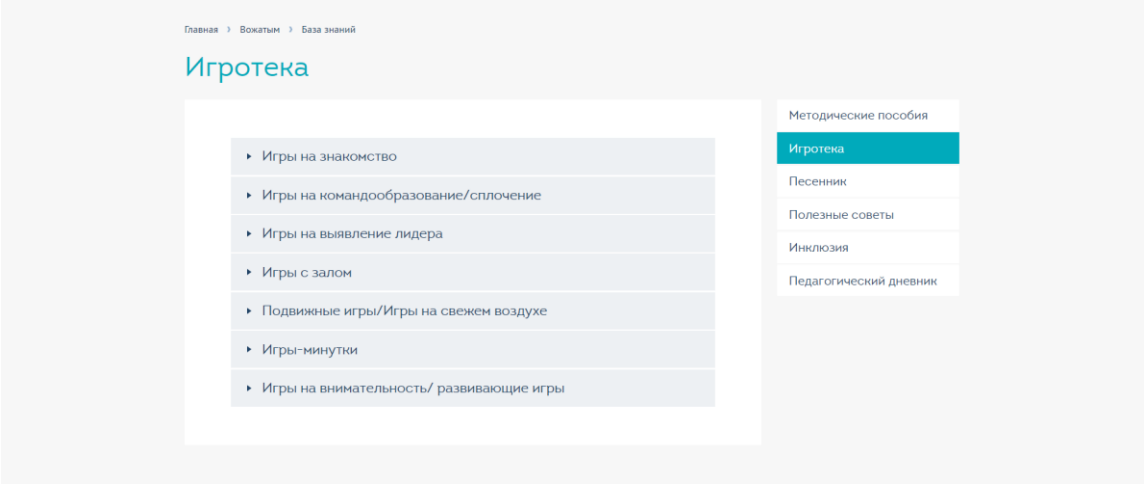

Рисунок 1.2

Преимущества:

- хорошо прописанные игры;
- много категорий игр: на знакомства, сплочение, лидера, подвижные игры, с залом и т. д.

Недостатки:

- нет возможности поставить фильтры;
- нужен доступ к сети Интернет;
- не удобный интерфейс.

Приложение «Справочник вожатого» (рисунок 1.3).

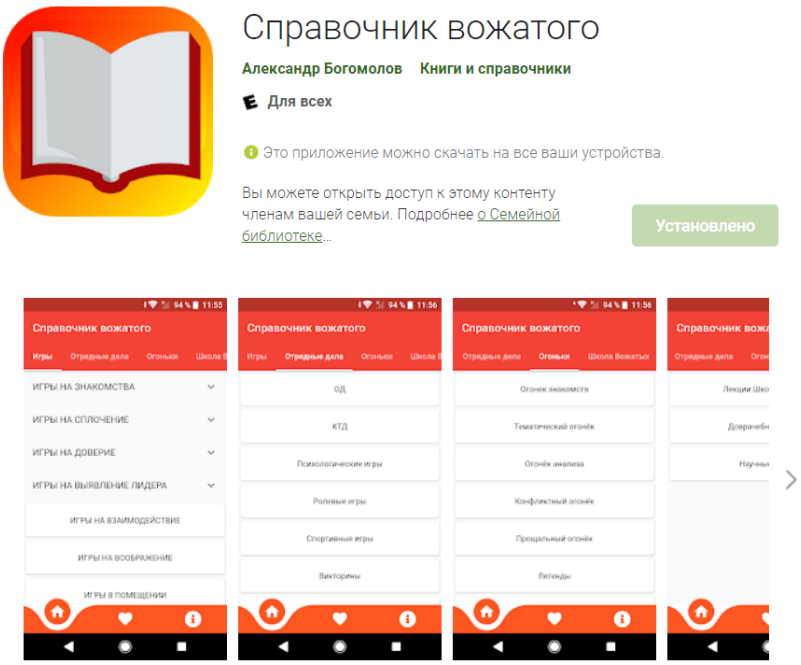

Рисунок 1.3 – Справочник вожатого

Это оффлайн приложение, позволяющее иметь множество игр, легенд и методического материала под рукой.

Преимущества:

- работает без интернета;
- огромная библиотека знаний, более 15 категорий игр, в каждой из которых более 20 игр;
- есть множество больших отрядных дел;
- в некоторых играх можно выбрать интересующий возраст;
- дана дополнительная информация, которая нужна для работы вожатого.

Недостатки:

- нет фильтров;
- есть повторяющиеся и непрописанные игры.

Приложение «Методичка вожатого» (рисунок 1.4).

Рисунок 1.4 – Методичка вожатого

Преимущества:

- кроме игр есть легенды, песенник, отрядные дела;
- имеются игры, у которых прописан возраст;

- расположена дополнительная информация по работе с детьми.

Недостатки:

- основная страница сделана не аккуратно (текст находится не на своём месте);
- небольшая база знаний;
- нет фильтров.

Приложение «Мегавожатый» (рисунок 1.5).

Рисунок 1.5 – МегаВожатый

Преимущества:

- есть дополнительная информация по работе с детьми;
- у игр прописан возраст детей;
- игры разделены по категориям;

Недостатки:

- одна страница загружается несколько минут;
- не работает без доступа в сеть Интернет.

Детские лагеря, как правило, расположены в зеленой зоне, особенно в Ленинградской области. Они находятся в лесах, из-за этого существуют

проблемы со связью и доступом в Интернет. Поэтому оптимально иметь такое приложение, которое можно загружать не на стационарном компьютере, а на других гаджетах, где легко подключить мобильный интернет, и потом пользоваться этим приложением, даже если нет доступа к сети.

Мобильное приложение – программное обеспечение, предназначенное для использования на мобильных устройствах (смартфоны, планшеты и др.). Благодаря таким программным продуктам мы можем управлять услугами, совершать онлайн покупки, использовать различную информацию, читать, развиваться и учится, ориентироваться в пространстве с помощью GPS, общаться, слушать музыку и многое другое. Почти все популярные приложения работают на мобильных платформах iOS и Android, так как это самые распространенные операционные системы для мобильных устройств.

Существует три основных вида мобильных приложений:

- нативные приложения;
- web-приложения;
- гибридные приложения.

Эти типы определяют подходы к разработке программного обеспечения для смартфонов. *Нативные приложения* - самые популярные и дорогие в разработке, потому что создаются для каждой операционной системы индивидуально.

Преимущества:

- высокая скорость работы, связанная с тем, что приложения создаются с учётом возможностей и особенностей платформ;
- возможность пользоваться функциями устройства (например, Bluetooth, список контактов, камера и другое);
- имеется доступ к системе оповещения устройства;
- можно создать приложение с режимом оффлайн;
- высокая защита данных пользователя.

Недостатки:

- дорогой процесс разработки;

- занимают много памяти на телефоне.

Кроме разработки нативных приложений, IT-специалисты занимаются адаптацией *web-приложений* к использованию их на мобильных устройствах. Они запускаются через браузер и обычно написаны на языке HTML5.

Преимущества:

- процесс разработки такого вида быстрее и дешевле, чем нативная. Это связано с тем, что для его создания не требуется учитывать все особенности платформы;
- не занимает память устройства, так как не требуется его установка на него.

Недостатки:

- какие-то функции могу по-разному отражаться или вовсе не работать, так как зависят от возможностей браузера;
- нет возможности использовать камеру, жесты, датчики положения и другое;
- не работают без подключения к сети Интернет.

Кроме разработки нативных и веб-приложений, существует *гибридный* подход к разработке мобильных приложений. Это использование двух предыдущих подходов в одном. Такие приложения можно загрузить и установить на устройство. Но мобильные приложения разрабатываются при помощи веб-фреймворков. Они обрабатываются через браузер в само приложение.

Преимущества:

- большая часть кода является одинаковой для разных платформ;
- работают быстрее и более плавно, что web-приложения;
- есть возможность использовать функции мобильного устройства;
- можно создать приложение, которому не нужно подключаться к сети Интернет.

Недостатки:

- уступают по показателям нативным приложениям.

1.2. Выбор технологии и инструментов разработки мобильного приложения

Для выбора технологии создания мобильного приложения для игротеки вожатого был проведен опрос бойцов СПО «Маэстро». В результате опроса выяснилось, что на телефонах почти в равной степени установлены как ОС Android, так и iOS (процентное соотношение: 40% к 60%). Поэтому необходимо создать гибридное приложение, которое будет обладать кроссплатформенностью, то есть способностью программного обеспечения работать на нескольких платформах. Такая разработка позволит охватить две операционные системы одним кодом. Это обеспечивает почти нативный подход к разработке благодаря интерфейсу визуализации с использованием собственных элементов управления.

Каждая операционная система мобильных устройств имеет свои технические возможности для запуска межплатформенных программных продуктов. Рассмотрим решения создания кроссплатформенных приложения, которые существуют на данный момент.

Есть три варианта, которые можно использовать с технической точки зрения. Первое - использование WebView, которое имеется у всех операционных систем. Второй вариант - использование низкого уровня OpenGL/DirectX и языка C/C++ (Qt) или скомпилированного языка Dart (Flutter), с помощью которого можно получить высокую производительность, но не всегда очевидный Look and Feel. И третий подход - это применение Xamarin или ReactNative, которые обеспечат нативный пользовательский интерфейс и высокую производительность с минимальными расходами за счёт системного API верхнего уровня.

На рисунке 1.6 отражены архитектура, а также основные возможности и ограничения этих фреймворков.

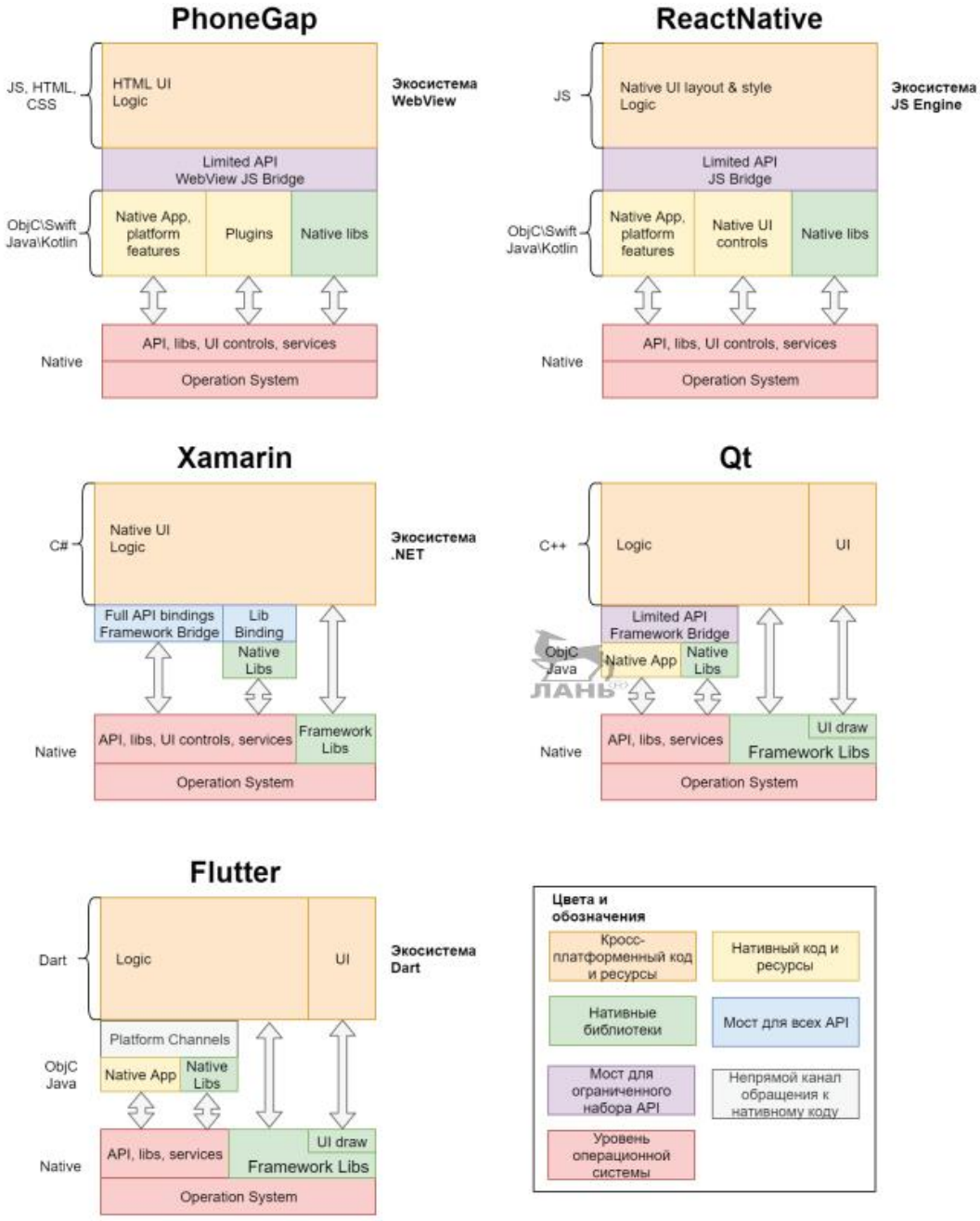

Рисунок 1.6

Первый фреймворк, который необходимо рассмотреть: PhoneGap. Он основан на использовании WebView и не очень сложен в реализации, так как с помощью него разрабатывается небольшое нативное приложение, отображающее встроенный web-браузер и страницу HTML5. Все элементы интерфейса на странице имитируются под родные, так как отсутствуют нативные контроллеры и доступ к API. Чтобы получить доступ к системной функциональности необходимо иметь специальные плагины, через которые

происходит нативная реализация на любой платформе с помощью JS-методов внутри web-браузера.

PhoneGap дает возможность делить буквально весь код между платформами, но для этого требуется осуществление нативной части на ObjectiveC, Java и C#. Ещё этот фреймворк применяется в известном Ionic, который дает большую численность готовых плагинов для различных функций системы.

Но интерфейсы приложений, созданных на базе WebView, лишь делаются подобными на нативные с поддержкой HTML/CSS стилей.

Для создания программного продукта на PhoneGap необходимы знания работы с HTML, JavaScript, CSS, ObjectiveC, Java и неплохие навыки для интеграции нативной и кроссплатформенной частей. К сожалению, в приложениях с непростым интерфейсом такие одностраничные страницы будут тормозить и показывать не очень хороший результат.

Чтобы передавать сложные структуры данных и классов, их надо сериализовать и дисериализовать в формате JSON.

В завершении отметим, что PhoneGap довольно взрослый подход и имеет большое количество работающих плагинов.

Следующее решение – ReactNative, которое в настоящее время является самым популярным решением для создания кроссплатформенного приложения. Данный фреймворк позволяет использовать JavaScript для создания нативного интерфейса, он предоставляет высокую производительность, такую же как у нативных приложений. Но в архитектуре фреймворка есть мост, который снижает скорость функциональности и UI.

При разработке программного продукта с помощью ReactNative разработчику будет необходимо создать нативную часть, инициализирующую JS-движок и код. Дальше уже JS-приложение создает нативные объекты и управляет ими. В ReactNative имеется две кнопки перезапуска Hot Restarting и Hot Reloading. Вторая позволяет обновлять

приложение без необходимости публиковать новую версию в магазины приложений (например, GooglePlay и AppStore). Также у этого фреймворка есть много библиотек и плагинов.

ОС iOS имеет некоторые ограничения, из-за которых код будет интерпретироваться, а не компилироваться. Но это не особо влияет на скорость работы приложения.

Чтобы передавать сложные структуры данных и классов, их надо сериализовать и дисериализовать в формате JSON.

Чтобы работать с ReactNative необходимы знания JavaScript, iOS и Android. У фреймворка имеется большая документация по интеграции приложения, а интерфейс имитируется из JavaScript. Но, к сожалению, пока у архитектуры есть недоработанные места, потому что фреймворк ещё молодой.

Третий фреймворк – Qt, использующийся очень широко для разработки десктопных приложений. Он дает возможность импортировать в ОС, где есть API для C++ (iOS и Android обладают таким свойством). Плюсом этой архитектуры является действенная система отрисовки интерфейса с помощью графического движка или на основе OpenGL, поэтому его портировать. То есть приложение можно стилизовать так, как захочется, потому что есть механизмы отрисовки UI.

Пользовательский интерфейс на базе Ot не считается нативным, он просто делает его подобным, так как поддерживает стилизацию.

Qt имеет много недостатков, потому что хранит в себе небольшую библиотеку и что-то новое подключить очень сложно, а также имеются трудности при сборке и отладки продукта.

Последняя архитектура, которую необходимо рассмотреть – Flutter. Это достаточно молодой фреймворк, впервые опубликованный только в 2015 году, но известный благодаря простоте и своей высокой скорости. В нём имеется простая функциональность для легкой интеграции, а также ему не нужно большое число оберток.

Необходимые элементы интерфейса фреймворк рисует сам на экране, взаимодействует с пользователем и учитывает жесты и касания. Приложения на Flutter разрабатываются с помощью C-подобного языка Dart.Эта архитектура дает возможность получить отзывчивы интерфейс, но он может казаться ненативным и требовать доработки.

Он легко портируется на новые платформы, но существенным минусом является то, что это молодой фреймворк и в сложных приложениях могут возникнуть трудности. Во Flutter реализовано автоматическое обновление с помощью Hot Reload, который помогает ускорить процесс разработки, а потом и дальнейшее пользование продуктом.

После сравнения разных технологий создания кроссплатформенного приложения был выбран флаттер по нескольким причинам:

- возможность работы в оффлайне, он без ограничений поддерживает работу с базами данных, с файловой системой: можно реализовать любой вариант кэширования, который будет отвечать запросам;
- полностью бесплатное;
- высокая скорость разработки;
- сокращенное время тестирования;
- много справочной информации и большое сообщество.

После выбора архитектуры кроссплатформенного фреймворка, необходимо подобрать инструменты разработки.

SDK (сокращение от англ. software development kit — переводится как «комплект для разработки программного обеспечения») — это набор инструментов для разработки программного обеспечения в одном устанавливаемом пакете. Они облегчают создание приложений, имея компилятор, отладчик и иногда программную среду. В основном они зависят от комбинации аппаратной платформы компьютера и операционной системы.

1. Flutter Version Manager или FVM.

Этот инструмент дает возможность легко и быстро возвращаться к любым версиям Flutter, а также управлять ими и кэшировать. Обеспечивает

загрузку различных каналов: Master, Dev, Beta, Stable. Можно просмотреть доступные каналы и выпуски, настроить версию Flutter для каждого проекта. Также SDK отличается динамических подходом к отладке IDE, обеспечивает согласованность между командами и средами CI.

Некоторые релизы могут быть нестабильными, здесь также подойдет FVM, потому что с помощью него можно быстро скачать обновления, переключиться на другую версию Flutter или если ветка не подходит, вернуться назад. Если необходимо сделать небольшое исправление, где не требуется обновлять версию Flutter, библиотеки, проводить тестирование, то FVM также подойдет.

Переключение между версиями через FVM делает с помощью одной команды, но если делать это вручную, то это займет много времени, так как всё зависит от интернет-соединения, мощности компьютера и другого.

2. Visual Studio Code

VS Code — это текстовый редактор, поэтому он работает быстрее, чем Android Studio. Также он имеет простой интерфейс и его легко можно настроить под себя — доступны не только самые базовые версии подсветки кода и отладки, но и широкие возможности кастомизации.

Но в VS Code нет опции создания нового Flutter-проекта. По умолчанию он может только открывать готовый. Для создания нового нужно использовать командную строку.

3. Android Studio

Главным минусом данной IDE является то, что она используется много оперативной памяти и поэтому необходимо иметь мощный компьютер с хорошим процессором.

Для создания мобильного приложения “Игротека вожатого” был выбран Android Studio. Его основные преимущества:

- Поддержка разработки на нескольких языках программирования: Java, Kotlin и другие. Установив Flutter SDK, можно писать еще и на Dart;

- Удобный, настраиваемый редактор с подсветкой синтаксиса, умным завершением кода и другими фичами. Зная комбинации горячих клавиш, можно в несколько раз ускорить свою работу;
- Встроенный AVD Manager с инструментами для работы с эмуляторами;
- Android Profiler для отслеживания загрузки процессора, памяти, сети и батареи устройства;
- Layout Inspector для анализа оптимальности верстки в приложении;
- Android Debugger для просмотра и изменения информации о состоянии объектов и переменных во время отладки. Также показывает стек вызова методов и предоставляет возможность настраивать breakpoints.

Недостаток Android Studio не сильно повлияет на разработку, так как мой проект не большой коммерческой приложение, а скорее учебный, которому не надо выполнять много функций и задач.

ГЛАВА 2. Разработка мобильного приложения “Игротека вожатого”

2.1. Проектирование мобильного приложения “Игротека вожатого”

Рассмотрим этапы, которые необходимо пройти при создании мобильного приложения.

1. Определить цель и задачи данного приложения.
2. Проектирование и дизайн.
3. Процесс разработки.
4. Тестирование.
5. Мониторинг.

Приложение проектируется с целью создания единой системы с игротекой, в которой имеется возможность выбирать нужные параметры.

Задачами этого приложения являются:

- обеспечения сбора и первичной обработки исходной информации, необходимой для создания игротеки;
- повышение качества (полноты, точности, достоверности) информации.

На главной или домашней странице будут кнопки с различными видами игр, а также свечки (коллективное обсуждение на какую-либо тему в конце дня) и мастер-классы. Всего будет 16 категорий, а именно: свечки, ролевые, подвижные, творческие, спортивные, станционные, интеллектуальные, сквозные, общелагерные игры, игры на знакомства, сплочение, доверие и пятиминутки, квесты, отрядные дела и мастер-классы.

После создания главной страницы, необходимо спроектировать страницы для всех игр. На каждой такой странице должен быть фильтр, где будут такие критерии, как возраст детей, период смены и продолжительность. В категории “Свечки” добавиться фильтр типа “свечки”.

Для создания успешного приложения необходимо правильно выбрать метод управления программным продуктом. На данный момент существует три основных метода, а также несколько методологий ведения проектов. В первую очередь при выборе способа управления следует определить

ключевые особенности продукта, его возможности и требования к созданию новых версий.

Особенностью разрабатываемого приложения будет то, что количество основных частей будет заранее определено и не будет меняться. Данные будут храниться в базе данных, в которую можно будет добавлять новые игры.

Модель разработки программного продукта описывает, какие стадии жизненного цикла оно проходит и что происходит на каждой из них.

Waterfall («водопадная», «каскадная») одна из старейших моделей и предполагает постепенное прохождение этапов, завершающихся до начала следующей. Выделяют пять стадий жизненного цикла проекта: инициализация, планирование, разработка, реализация и тестирование, мониторинг и завершение.

Waterfall применяется в том случае, если соблюдаются некоторые условия: низкие риски, невысокая критичность сроков окончания проекта, а также если высока вероятность того, что требования к продукту не будут претерпевать изменений.

Главными преимуществами использования водопадной модели состоит в следующем:

- простота и понятность заказчику;
- постепенное прохождение стадий, завершающихся до начала следующей;
- на каждом этапе создается набор проектной документации, полной и согласованной заказчиком после приемки-сдачи;
- легкое планирование сроков выполнения работ и затрат на них.

Минусы водопадной модели заметны, когда нельзя определить конкретные требования или когда они нестабильны. При данной модели сложно вернуться хотя бы на один шаг назад для исправления недочетов, это

увеличивает затраты на разработку и редактирование ошибок и срывает сроки разработки.

Рассмотрим другую модель управления – инкрементую.

В её основе лежит инкрементирование, то есть процесс пошаговой реализации продукта за счёт постепенного расширения возможностей его функций. За основу берется заблаговременный полный набор требований. На первых фазах жизненного цикла создается архитектурное проектирование системы и вычисляется число необходимых инкрементов и их функций.

После на всех итерациях кодируются, тестируются и устанавливаются очередные инкременты. В первую очередь происходят этапы конструирования, тестирования и установки набора основных функций, которые создают базу продукта, обозначаются требования первой важности, которые нужны для создания успешного проекта.

Задача всех итераций – получение уже на ранних стадиях разработки работающей версии программного обеспечения с определенной функциональностью. После каждой итерации можно анализировать промежуточные результаты и реакцию на них заказчиков.

Такую модель стоит использовать при следующих обстоятельствах:

- если большинство требований и функций можно сформулировать заранее, но они должны появиться не сразу;
- если необходимо выполнить быструю поставку продукта на рынок, хотя бы с базовыми свойствами;
- если проект долгосрочный, то есть на него предусмотрено большое время на разработку, минимум год;
- при разработке продуктов с низкой или средней степенью риска;
- при проекте, использующем новые технологии.

Инкрементная модель подходит для проектов, в которых точное техническое задание прописано уже на старте, а продукт должен быстро выйти на рынок.

Третья модель, которая является комбинацией предыдущих двух, называется спиральная. Её главная особенность заключается в том, что она учитывает риски (например, отсутствие необходимых ресурсов, потеря актуальности приложения, риск сорвать сроки и т.п.). Такая модель помогает внедрить элементы разработки программного обеспечения из нескольких моделей процессов для программного проекта на основе уникальных шаблонов рисков, обеспечивая эффективные процесс разработки.

Спиральная модель жизненного цикла – модель с повторяющимися этапами (рисунок 2.1). Каждый виток спирали – один каскадный или итеративный жизненный цикл. В конце каждого витка получается законченная версия системы, реализующая набор функций. Она предъявляется пользователю.

Рисунок 2.1

На следующий виток переносится документация, разработанная на предыдущем витке, и процесс повторяется. Система разрабатывается постепенно, проходя постоянные согласования с заказчиком. На каждом витке спирали функциональность системы расширяется, постепенно достигая полной величины.

Преимущества спиральной модели:

• наиболее реально (в виде эволюции) отображает разработку программного обеспечения;

• позволяет явно учитывать риск на каждом витке эволюции разработки;

• включает шаг системного подхода в итерационную структуру разработки;

• использует моделирование для уменьшения риска и совершенствования программного изделия.

Недостатки спиральной модели:

• отсутствие достаточной статистики эффективности модели;

• повышенные требования к заказчику;

• трудности контроля и управления временем разработки.

Для разработки данного приложения была выбрана спиральная модель, во-первых, потому что она обеспечивает разбиение большого объема работы на небольшие части: страницы приложения будут создаются поэтапно, после каждой новой страницы приложение проверяется на работоспособность. Во-вторых, во время разработки продукта можно уточнить поставленные цели и повысить у него вероятность предсказуемого поведения. И в-третьих, данная модель сможет повысить производительность товар за счет повторного использования предыдущих результатов.

Всего у спиральной модели выделяют четыре этапа.

I этап - определение целей и альтернативных решений. Здесь собираются требования от клиентов, разрабатываются и анализируются цели в начале каждого этапа, предлагаются альтернативные решения.

II этап - выявление и устранение рисков. На этой фазе происходит оценка всех возможных вариантов для того, чтобы выбрать подходящий, выявляются риски, которые могут возникнуть при данном решении. После этого создается стратегия, с помощью которой устраняются риски.

III этап - разработка новой версии продукта. Эта стадия связана с разработкой и тестированием продукта, учитывая выбранную стратегию. В конце получаем следующую версию приложения.

IV этап - обзор и планирование следующего этапа. На последнем этапе заказчик оценивает разработанную версию программного продукта, а далее начинается планирование следующих частей приложения.

2.2. Разработка мобильного приложения “Игротека вожатого”

Все части приложения на Flutter состоят из виджетов. Виджеты — это классы, используемые для создания пользовательского интерфейса. Они используются как для элементов макета, так и для элементов пользовательского интерфейса. Например, изображения, тексты, блоки, пустые поля – это всё виджеты. Из простых виджетов строят более сложные.

Создадим в AndroidStudio новый проект Flutter, а также макет мобильного устройства (рисунок 2.2).

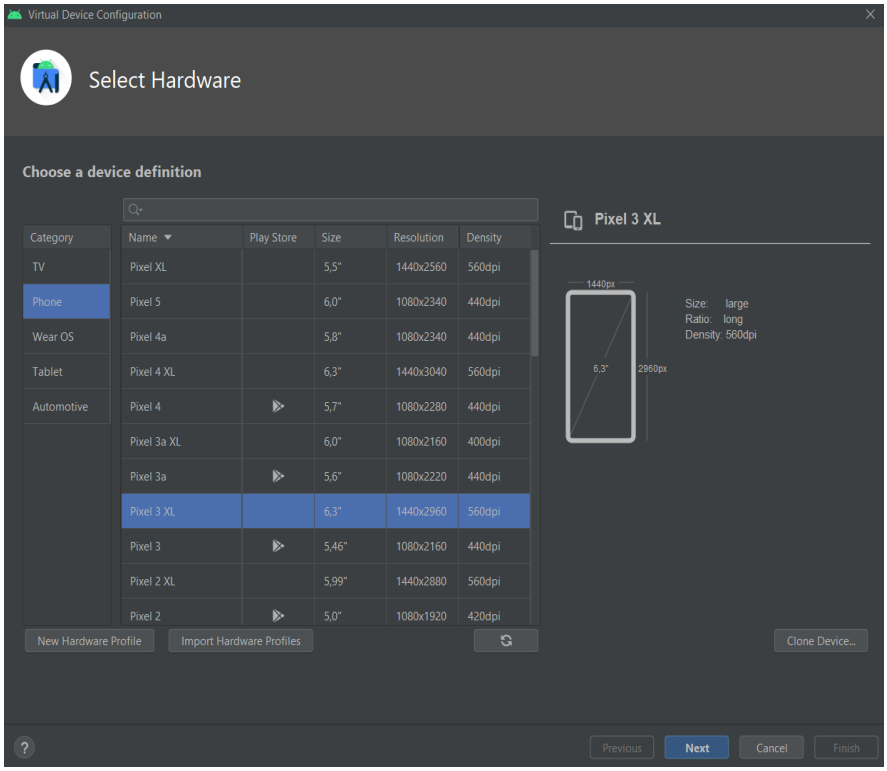

Рисунок 2.2

Создание приложения начинается с файла main.dart. Любое приложение должно иметь функцию main с типом void (то есть она не принимает никаких параметров). С помощью функции runApp будем запускать основную страницу, которая находится в классе MainPage.

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:my_projects_flutter/flutter_app.dart';

void main() => runApp(MainPage());
```

Больше в main ничего добавлять не будем, так как при его изменении, приходится запускать Hot Restart, который перезагружает всё приложение, длится этот процесс пару минут. Для обновления нашего приложения будем пользоваться Hot Reload, который значительно ускоряет процесс разработки.

Далее определим класс MainPage на отдельной странице flutter_app.dart. @override определяет функцию в классе-предке и в текущем классе.

Во Flutter есть два основных виджета: StatelessWidget и StatefulWidget. StatelessWidget подходит для неизменяемых виджетов, они не имеют внутреннего состояния. Внутри класса будем возвращать виджет MaterialApp, который нужен для создания графического интерфейса material design. С помощью виджета можно определить навигацию и другие функции. Он имеет много параметров, но воспользуемся всего одним: home, он задаёт основной виджет, который будет отображаться при загрузке. В него мы поместили класс HomeScreen, который будет находиться в другом файле home_screen.dart.

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:my_projects_flutter/screens/home_screen.dart';

class MainPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: HomeScreen(),
    );
  }
}
```

Создадим класс HomeScreen используя виджет StatelessWidget. Для основной страницы он отлично подходит, так как категории игр заранее определены.

Названия категорий игр расположены в списке строк List<String> events. Перед ним обязательно было добавить final, так как используется Stateless виджет.

После объявления списка необходимо вернуть виджет Scaffold, который создает интерфейс в стиле Material Design. Конструктор имеет довольно много параметров, в числе которых appBar и body, использовавшиеся для создания

главной страницы. AppBar – панель приложений. С помощью параметра title открываем виджет Text и записываем название, которое будет отражаться на верхней панели. Также была возможность определить её цвет благодаря параметру backgroundColor и поставить введенный текст по середине помог параметр centerTitle.

Параметр Body задает основное содержимое данного класса в виде другого виджета. Другим виджетом был выбран ListView.separated. При наполнении ListView можно по своему стилизовать виджет, который применяется для каждого элемента списка - задать отступы, границы, какую-то графику. Но если необходимо создание разделителя между элементами в ListView, то можем воспользоваться еще одним конструктором класса ListView.separated, что и было сделано. Этот конструктор также, как и конструктор ListView.builder позволяет задать с помощью параметра itemBuilder функцию для создания элементов в ListView. Но кроме того с помощью дополнительного параметра separatorBuilder он позволяет задать функцию для создания границ между элементами, например, горизонтальная черта, как реализовано в данном приложении.

Внутри него мы будем получать наши данные из списка с помощью параметра itemBuilder, и там будем создавать контейнер. Container (контейнер) — это класс виджетов, который позволяет настраивать свои дочерние виджеты. Можно использовать **Container** (контейнер), когда вы хотим добавить внутренние отступы, поля, границы или цвет фона, чтобы обозначить некоторые из его характеристик.

В данном примере каждый Text (текстовый) виджет помещается в **Container** (контейнер) для добавления полей. Все Row (строки) также помещаются в **Container**, чтобы добавить внутренний отступ вокруг строки.

Остальная часть пользовательского интерфейса в этом примере контролируется свойствами. Установите цвет значка, используя его цветовое свойство **color**. Используйте свойство **Text.style** для установки шрифта, его цвета, ширины и так далее. Столбцы и строки имеют свойства, которые

позволяют указать, как их дочерние элементы выровнены по вертикали или горизонтали, и сколько места должно занимать дочернее пространство.

```
import 'package:flutter/material.dart';
import
'package:my_projects_flutter/screens/connection_screen.dart';

class HomeScreen extends StatelessWidget {
  final List<String> events = [
    "Игры на знакомство",
    "Игры на сплочение",
    "Игры на доверие",
    "Игры пятиминутки",
    "Ролевые игры",
    "Подвижные  игры",
    "Творческие  игры",
    "Спортивные  игры",
    "Станционные  игры",
    "Интеллектуальные  игры",
    "Сквозные  игры",
    "Общелагерные  игры",
    "Квесты",
    "Отрядные дела",
    "Мастер-классы",
    "Свечки",
  ];
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("ИГРОТЕКА ВОЖАТОГО"),
        backgroundColor: Colors.cyan[300],
        centerTitle: true,
      ),
      // ignore: prefer_const_constructors
      body: ListView.separated(
        physics: const BouncingScrollPhysics(),
        padding: const EdgeInsets.all(20),
        itemCount: events.length,
        itemBuilder: (_, index) =>
            Container(
              color: Colors.lightBlue,
              padding: const EdgeInsets.all(20),
              child: Column(
                children: [
                  Text(events[index], style: const
TextStyle(fontSize: 30),),
                  ElevatedButton.icon(
                    label: Text('Узнать подробнее', style:
TextStyle(fontSize: 20),),
                    icon: const Icon(
                      Icons.arrow_drop_down,
                      color: Colors.white,
```

```
                        size: 30.0,
                      ),
                      onPressed: () {Navigator.push(context,
MaterialPageRoute(builder: (context) => ConnectionScreen()));},
                    ),
                  ],
                ),
              ),
          separatorBuilder: (_, __) => const Divider(color:
Colors.white,),
        ),
      );
    }
}
```

Рисунок 2.3

SQLite это наиболее популярный способ для хранения данных на мобильных устройствах. Sqflite — одна из наиболее часто используемых и актуальных библиотек для подключения SQLite базы данных в Flutter.

База данных необходима для хранения всех игр и их оперативного поиска. Так как игры могут добавляться, то это не Stateless, а Stateful виджет.

StatefulWidget – рекомендуется для изменяемых виджетов, с изменяемым внутренним состоянием.

Чтобы использовать SQLite в приложении Flutter, первым шагом является включение пакета sqflite в pubspec.yaml и path_provider на 1.6.0 или выше. Помимо этого, проект остается простым, чтобы с ним было легко работать и понимать.

Для хранения данных простой класс модели данных предоставит необходимые методы для преобразования между дружественным к SQLite форматом данных и объектом, который можно использовать в приложении. Ккласс Model будет служить базовым классом для моделей данных.

Класс Model очень прост и создан для удобства, чтобы определить свойства / методы, которые можно ожидать от моделей данных, например id, как показано выше. Это позволяет создавать одну или несколько конкретных моделей данных, которые будут соответствовать этому базовому шаблону проектирования. Для этого приложения класс модели элементов Games создается в lib/models/games.dart.

Класс GamesItem содержит свойства type, age, period и а также имеет простой конструктор для создания нового элемента Games. Для преобразования между экземплярами объектов GamesItem и Map, используемых базой данных, были определены методы toMap и fromMap.

Вместо случайного смешивания логики базы данных в приложении основные методы обработки базы данных помещены в lib/services/db.dart для удобства и простоты обслуживания:

Этот abstract класс, поскольку он не предназначен для создания экземпляра, и требуется только одна его копия в памяти. Внутренне он содержит ссылку на базу данных SQLite в свойстве _db. Номер версии базы данных жестко запрограммирован (1), но в более сложных приложениях версию базы данных можно использовать для переноса схем базы данных

вверх или вниз по версии, чтобы обеспечить развертывание новых функций без необходимости стирать базу данных и начинать с нуля.

Экземпляр базы данных SQLite создается в методе init с использованием имени базы данных для этого проекта example. Если база данных example еще не существует, автоматически вызывается onCreate. Здесь размещаются запросы на создание структуры таблицы. В этом случае у нас есть таблица games_items с первичным ключом id, а также поля, соответствующие свойствам в приведенном выше классе GamesItem.

Метод query наряду с insert, update и delete определены для выполнения стандартных операций CRUD в базе данных. Они предоставляют простые абстракции и позволяют содержать логику базы данных в этом классе, что может быть чрезвычайно полезно при рефакторинге или выполнении другого обслуживания приложения вместо того, чтобы, например, выполнять поиск и замену строк в нескольких файлах или исправлять странные ошибки, которые появляются при создании простые изменения.

И последнее, но не менее важное: у нас есть основная логика приложения и UX в lib/main.dart.

Когда приложение запускается и виджет HomeScreen отображается, делается вызов списка задач с помощью refresh() из таблицы games_items и сопоставление его с одним из объектов GamesItem.

Игры могут быть добавлены путем нажатия плавающей кнопки с действием и ввода имени задачи. Когда нажата Save, создается элемент списка с помощью DB.insert. Задача добавляется в базу данных с помощью щелчка по задаче в списке, который переключает логическое значение complete и сохраняя измененный объект обратно в базу данных с DB.update. Свайп по горизонтали на задаче удалит элемент Games, предоставив его методу DB.delete. Всякий раз, когда в список вносятся изменения, обращение к refresh() и последующему setState() гарантирует, что список обновляется должным образом.

SQLite предоставляет удобный стандартный для отрасли способ сохранения данных локально в приложении. В этом примере показано, как реализовать базовые операции CRUD для создания и управления простыми записями в базе данных SQLite (рисунок 2.4).

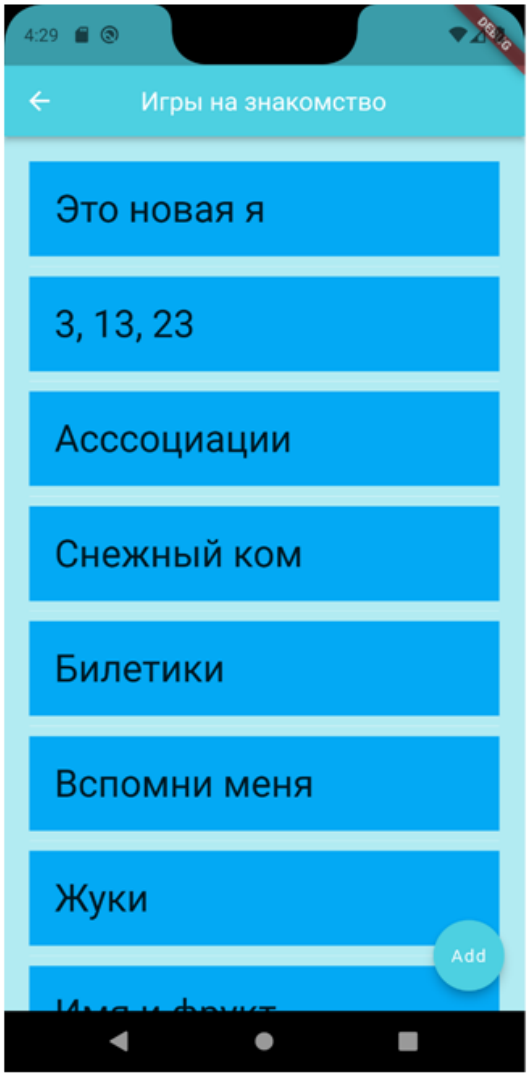

Рисунок 2.4

После создания программного продукта необходимо провести его тестирование для того, чтобы убедиться в его работоспособности [22]. Существую разные виды тестирования интерфейсов. Я остановила свой выбор на проверке посредством наблюдения за пользователем, так как это один из самых легких видов оценки приложения. Пользователь получает задания, которое связанно с функционалом приложения, и он его просто

выполняет. В то время, как он решает поставленные перед собой задачи, экран фиксирует его действия дл дальнейшего анализа.

Как правило, человека оставляют одного, чтобы он не тревожился, так как посторонние предметы отвлекают от текущей задачи и из-за этого человек может показать неправильный результат.

Для тестирования текущего приложения было приглашено пять человек, задание выполнялось на компьютере, так как пока нет возможности открывать приложение на мобильном телефоне. Было поставлено 3 задачи:

1. Посчитать количество ролевых игр для детей 10 – 12 лет;
2. Найти игру на сплочение «Равновесие»;
3. Выбрать подходящий мастер-класс, если из материалов есть только бумага, клей и ножницы.

По итогам поисков, все справились с задачами, которые им были предложены. Интерфейс приложения интуитивно понятен.

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены особенности игровой деятельности со школьниками в летних оздоровительных лагерях. Выявлено, что детям нужна смена деятельности каждый день, поэтому вожатым просто необходимо знать множество вариантов развлечения детей.

Так как всё в голове не удержишь, надо чтобы у тебя всегда было что-то под рукой, куда можно посмотреть и вдохновиться. Для этого существуют игротеки вожатого. Были проанализированы разные варианты онлайн решений, выявлены их преимущества и недостатки.

Рассмотрены различные технологии разработки мобильного приложения. По итогам анализа был выбран Flutter, так как он бесплатен, у него высокая скорость разработки, но сокращенное время тестирования. А в качестве инструмента была выбрана AndroidStudio, потому что здесь легко создать проекты Flutter, много подсказок и удобная подсветка синтаксиса.

После анализа были рассмотрены особенности создаваемого приложения, а именно необходимость создания фильтров с параметрами возраста, тип игры и количество участников. С учетом этого составлена структура: на главной странице все категории игр, а после открытия нужного типа игры открывается информация обо всех играх такого вида.

После выявленных особенностей приложения, была выбрана спиральная модель жизненного цикла, так как у проекта имеются некоторые риски.

Разработано мобильное приложение «Игротека вожатого», используя официальную документацию Flutter и источники из сети Интернет. А также оно было протестировано на будущих пользователях и показало хорошие результаты.

Задачи ВКР выполненные полностью.

СПИСОК ЛИТЕРАТУРЫ

1. Веденеева О. А. Теоретические основы подготовки вожатого: учебное пособие / Веденеева О. А., Сайгушев Н. Я. - Москва: ФЛИНТА, 2021. - 120 с.

2. Калинина З. Н. Советы начинающему вожатому: Учебно-методическое пособие / Калинина З. Н., Декина Е. В., Пазухина С. В. и другие. - Тула: Тульский государственный педагогический университет им. Л. Н. Толстого, 2018. - 153 с.

3. Виды мобильных приложений [Электронный ресурс] / Мобильная рекламная платформа BYYD - URL : https://www.byyd.me/ru/blog/2021/12/mobile-apps-types/

4. Черников В. А. Разработка мобильных приложений на C# для iOS и Android. - Москва: ДМК Пресс, 2020. - 188 с.

5. Васильев Н. П. Введение в гибридные технологии разработки мобильных приложений : учебное пособие для ВО / Васильев Н. П., Заяц А. М. - Санкт-Петербург : Лань, 2020. - 160 стр.

6. Маркин Е. И. Использование реактивного программирования при разработке мобильных приложений / Маркин Е. И., Рябова К. М. // Computational nanotechnology. - 2016. - №2. - с. 170 - 173.

7. Рязанов О. Ю. Интерпретируемые мобильные приложения / Рязанов О. Ю., Рязанов Ю. Д. // Вестник ВГУ, серия: системный анализ и информационные технологии. - 2018. - №3. - с. 140 - 145.

8. Соколова В. В. Разработка мобильных приложений: учебное пособие / Соколова В. В.; Томский политехнический университет. - Томск: Изд-во Томского политехнического университета, 2014. - 176 с.

9. Какую IDE выбрать Android, Flutter и iOS-разработчику [Электронный ресурс] / Академия разработки MEDIASOFT - URL : https://academy.mediasoft.team/article/kakuyu-ide-vybrat-android-flutter-i-ios-razrabotchiku/ (дата обращения: 10.03.2022)

10. Кузин А. В. Разработка баз данных в системе Microsoft Access : учебник / Кузин А. В., Демин В. М. - 4-е изд. - Москва : ФОРУМ : ИНФРА-М, 2022. - 224 с. - (Среднее профессиональное образование).

11. Ревунков Г. И. Базы данных : учебно-методическое пособие / Ревунков Г. И., Ковалёва Н. А., Силантьева Е. Ю. и другие. - Москва : Издательство МГТУ им. Н. Э. Баумана, 2020. - 125 с.

12. Рочев К. В. Информационные технологии. Анализ и проектирование информационных систем : учебное пособие . 2-е изд. СПб.: Лань, 2019.

13. Ехлаков Ю. П. Управление программными проектами. Стандарты, модели : учебное пособие для вузов . 2-е изд. СПБ.: Лань, 2020.

14. Модели и методологии разработки ПО [Электронный ресурс] / GeekBrains - URL : https://geekbrains.ru/posts/methodologies (дата обращения: 10.04.2022)

15. Зубкова Т. М. Технология разработки программного обеспечения : учебное пособие. - СПб. : Лань, 2019.

16. Спиральная модель и архитектура разработки программного обеспечения [Электронный ресурс] / Industry - URL : https://janberg.by/spiral-model-spiralnaya-model-i-arxitektura-razrabotki-programmnogo-obespecheniya/ (дата обращения: 15.04.2022)

17. Семахин А. Н. Методы верификации и оценки качества программного обеспечения : учебное пособие. - Курган : издательство КГУ, 2018. - 150 с.

18. Заметти Ф. Flutter на практике: Прокачиваем навыки мобильное разработки с помощью открытого фреймворка от Google / пер. с анг. А. С. Тищенко. - Москва: ДМК Пресс, 2020. - 328 с.

19. Documentation Flutter [Электронный ресурс] / Flutter - URL : https://docs.flutter.dev (дата обращения: 25.04.2022)

20. Flutter и Dart документация на русском [Электронный ресурс] / Flutter.su - URL : https://flutter.su/docs (дата обращения: 20.05.2022)

21. Руководство по фреймворку Flutter [Электронный ресурс] / Сайт о программировании Metanit - URL : https://metanit.com/dart/flutter/ (дата обращения: 20.05.2022)

22. Батенькина О. В. Юзабилити информационных систем : учеб. пособие. - Омск : Изд-во ОмГТУ, 2015.