ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА МОСКВЫ

«КОЛЛЕДЖ СВЯЗИ №54»

ИМЕНИ П.М. ВОСТРУХИНА

КУРСОВАЯ РАБОТА

по: ПМ. 01 Разработка модулей программного обеспечения

междисциплинарного курса: МДК.01.03 Разработка мобильных приложений

на тему:

Разработка мобильного приложения для магазина товаров для животных

Выполнена студентом: Дмитрием Алексеевичем Ганюшкиным

И.О.Фамилия

Группы: 3ИСП11-15

номер группы

Основная профессиональная образовательная программа по специальности: 09.02.07 «Информационные системы и программирование (квалификация –программист)»

шифр и наименование специальности

Форма обучения: очная

Руководитель: Анастасия Николаевна Виеру _____

подпись, дата

Заведующий отделением ИКТ: Ирина Юрьевна Васильева _____

подпись, дата

г. Москва, 2022 г.

УТВЕРЖДАЮ
Зам.директора по ОУП
_____И.Г.Бозрова
«___»_____2022 г.

ЗАДАНИЕ

к курсовой работе студента

Ганюшкина Дмитрия Алексеевича

на тему: «Разработка мобильного приложения для магазина товаров для животных»

Тема курсовой работы утверждена приказом директора ____ т _____

Цель работы: Получение практических навыков по теме: «Разработка Мобильного приложения»

Основные вопросы, подлежащие разработке:

Теоретическая часть:

- Выбор инструментария для реализации приложения
- Основные сведения о системе Android
- Основные сведения о системе IOS
- Сведения о языке программирования Java
- Сведения о языке программирования Swift
- Анализ программного обеспечения для разработки мобильного приложения
- Анализ графических редакторов для создания прототипа
- Вывод по теоретической части

Практическая часть:

- Создание классов приложения
- Создание окон приложения
- Создание моделей приложения
- Создание адаптеров приложения
- Разработка модуля «Регистрация/авторизация»
- Разработка модуля «Карточки питания»
- Разработка модуля «Фильтры»
- Разработка модуля «Корзина»
- Вывод по практической части курсовой работы

Основная литература:

1. *Гриффитс Д.* Head First. Программирование для Android. – 2-е изд.: Питер, 2018. – 912 с.

2. *Дейтел Х., Дейтел П., Уолд А.*: Android для разработчиков. – 3-е издание, Питер 2016. – 512 с.

3. *Романчик В., Блинов И.*: Java. Методы программирования. – Четыре четверти, 2013. – 898 с.

Руководитель: Анастасия Николаевна Виеру _____
И.О.Фамилия подпись

Дата выдачи задания: 19.09.2022

Задание получил: Дмитрий Алексеевич Ганюшкин _____
И.О.Фамилия подпись

Дата получения задачи: 19.09.2022

ПЛАН-ГРАФИК

курсовой работы

Срок выполнения курсовой работы	19.09.2022 – 19.11.2022
Выбор темы курсовой работы	19.09.2022
Подбор и анализ исходной информации	20.09.2022
Подготовка и утверждение плана курсовой работы	20.09.2022
Выполнение задач курсовой работы	20.09.2022 – 12.11.2022
Оформление курсовой работы	01.11.2022
Исправление замечаний курсовой работы	10.11.2022
Предоставление готовой курсовой работы для рецензирования	12.11.2022
Предзащита курсовой работы	14.11.2022-15.11.2022
Защита курсовой работы	16.11.2022-19.11.2022

Руководитель Анастасия Николаевна Виеру

подпись

План принял к исполнению Дмитрий Алексеевич Ганюшкин

подпись

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Наряду с магазинами для человека появились и магазины для животных, зоомагазины. Данный рынок активно развивается, поэтому конкуренция на данном направлении велика. Прогресс идёт вперёд и бизнес вынужден, отвечать постоянно изменяющимся требования, к доступности информации. Многие потребители не желают терять время на поход в зоомагазин, и всё чаще пользуются интернет-магазинами. Перед предпринимателями встала задача, внедрения информационной системы по товарам, представленным в зоомагазине. Информация является наиболее важным ресурсом, а внедрение информационных систем, стало необходимым условием, для ведения конкурентно способного бизнеса. Поэтому было принято решение разработать мобильное приложение, так как большинство людей пользуется именно телефонами.

Актуальность: Разрабатываемое мобильное приложение является актуальным, так как большое количество людей пользуются различными приложениями по поиску и покупке товаров, из-за того, что нуждаются в содержании своих домашних питомцев

Данное программное обеспечение решено разработать для удобного и быстрого приобретения товаров в зоомагазине. Программа предназначена для потребителей. В случае, если потребителю необходим какой-либо товар, программа должна позволить быстро найти и приобрести его.

Объектом исследования в данной курсовой работе выступает автоматизированная информационная система интернет-магазина.

Предметом исследования является мобильное приложение для магазина товаров для животных.

ГЛАВА I. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ МАГАЗИНА ДЛЯ ЖИВОТНЫХ

1.1 Анализ предметной области

ООО «Happy Pet`s» строит бизнес на продаже товаров через интернет. Для повышения прибыли и удобства было принято решение создать мобильное приложение.

Целями предприятия является извлечение прибыли, предоставление услуг, расширение и продвижение компании.

Процесс взаимодействия покупателя с мобильным приложение выглядит следующим образом. Пользователь должен установить приложение,

по окончании установки, открыть его и зарегистрировать аккаунт. Затем пользователю предложит авторизоваться, после авторизации клиенту открывается доступ к интернет-магазину. Пользователь должен выбрать курс, который хочет приобрести, далее его переключает на карточку курса, где он может увидеть фотографию и описание курса.

Затем пользователь может добавить курс в корзину.

1.2 Выбор операционной системы мобильного приложения

Существуют две основные конкурирующие между собой операционные системы на базе которых разрабатываются мобильные приложения: Android и iOS. На данный момент, устройств, работающих на Android несравнимо больше, нежели чем на iOS. Так же, операционная система Android имеет открытый исходный код, что позволяет его легко адаптировать к новым платформам. В свою очередь iOS безопаснее и основана на принципе конфиденциальности.

1.2.1 Основные сведения о системе Android

Архитектура Android состоит из четырех уровней: уровень ядра, уровень библиотек и среды выполнения, уровень каркаса приложений (application framework) и уровень приложений. Система Android основана на

ядре Linux версии 2.6. На уровне ядра происходит управление аппаратными средствами мобильного устройства. На этом уровне работают драйверы дисплея, камеры, клавиатуры, WiFi, аудиодрайверы. Особое место занимают драйверы управления питанием и драйвер межпроцессного взаимодействия (IPC). Следующий уровень — это уровень библиотек и среды выполнения. Данный уровень представлен библиотеками libc (в Android она называется Bionic), OpenGL (поддержка графики), WebKit (движок для отображения Web-страниц), FreeType (поддержка шрифтов), SSL (зашифрованные соединения), SGL (2D-графика), библиотеки поддержки SQLite, Media Framework (нужна для поддержки мультимедиа). На этом же уровне работает Dalvik Virtual Machine — виртуальная машина Java, предоставляющая необходимую функциональность для Java-приложений. Следующий уровень — уровень каркаса приложений. На уровне приложений работает большинство Android-приложений: браузер, календарь, почтовый клиент, навигационные карты и т. д. (рис 1.1).

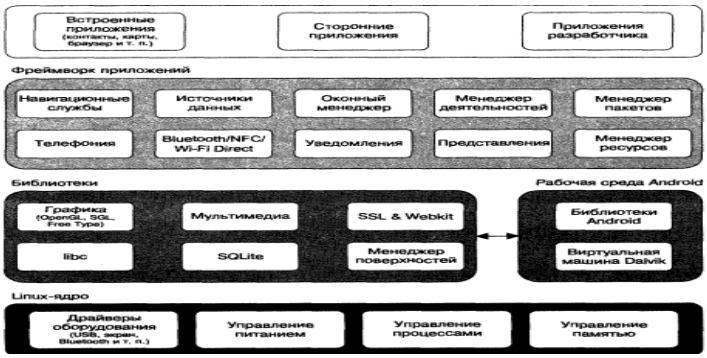

Рисунок 1.1 Элементы, формирующие программный стек Android

Ключевым компонентом для создания визуального интерфейса в приложении Android является activity (активность). Нередко activity ассоциируется с отдельным экраном или окном приложения, а переключение между окнами будет происходить как перемещение от одной activity к другой. Приложение может иметь одну или несколько activity.

Все объекты activity представляют собой объекты класса android.app. Activity, которая содержит базовую функциональность для всех activity. Однако сам класс AppCompatActivity, хоть и не напрямую, наследуется от базового класса Activity.

Все приложения Android имеют строго определенный системой жизненный цикл. При запуске пользователем приложения система дает этому приложению высокий приоритет. Каждое приложение запускается в виде отдельного процесса, что позволяет системе давать одним процессам более 7 высокой приоритет, в отличие от других.

Благодаря этому, например, при работе с одними приложениями не блокировать входящие звонки. После прекращения работы с приложением, система освобождает все связанные ресурсы и переводит приложение в разряд низкоприоритетного и закрывает его.

Все объекты activity, которые есть в приложении, управляются системой в виде стека activity, который называется back stack. При запуске новой activity она помещается поверх стека и выводится на экран устройства, пока не появится новая activity(рис.1.5).

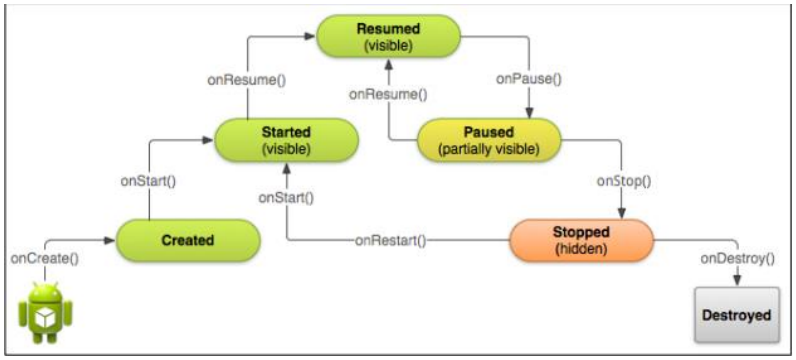

Рисунок 1.2 Схема жизненного цикла приложения на Android

Когда текущая activity заканчивает свою работу (например, пользователь уходит из приложения), то она удаляется из стека, и возобновляет работу та activity, которая ранее была второй в стеке.

1.2.2 Основные сведения о системе IOS

Приложения Xamarin.iOS выполняются в среде выполнения Mono и используют полную компиляцию заранее (AOT) для компиляции кода C# на языке сборок ARM. Это выполняется параллельно со средой Objective-C выполнения. Обе среды выполнения выполняются поверх ядра, подобного UNIX, в частности XNU, и предоставляют различные API-интерфейсы пользовательскому коду, позволяя разработчикам получать доступ к базовой собственной или управляемой системе. На рисунке 1.4 показан базовый обзор этой архитектуры:

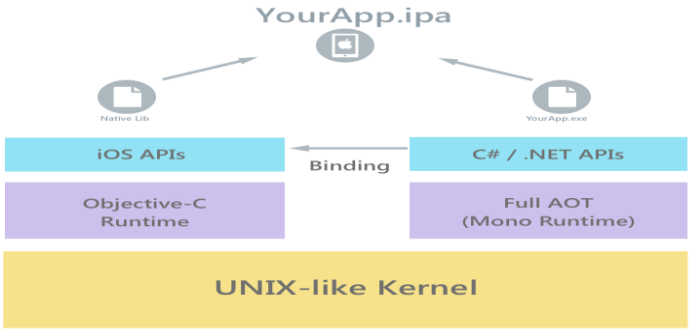

Рисунок 1.3 Архитектура приложения

При разработке для Xamarin часто используются собственные и управляемые коды. Управляемый код — это код, который выполняется с помощью среды CLR платформа .NET Framework или в случае Xamarin: среда выполнения Mono. Машинный код — это код, который будет выполняться изначально на конкретной платформе (например, Objective-C или даже скомпилированный код AOT на микросхеме ARM). В этом руководстве описывается, как AOT компилирует управляемый код в машинный код и объясняет, как работает приложение Xamarin.iOS, что позволяет полностью использовать API iOS Apple через использование привязок, а также доступ к. BCL NET и сложный язык, например C#. При компиляции любого приложения платформы Xamarin компилятор Mono C# (или F#) будет выполняться и компилирует код C# и F# в MSIL. Если вы используете приложение

Xamarin.Android, приложение Xamarin.Mac или даже приложение Xamarin.iOS на симуляторе, среда CLR .NET компилирует MSIL с помощью JIT-компилятора. Во время выполнения это компилируется в машинный код, который может выполняться в правильной архитектуре приложения.

Однако существует ограничение безопасности на iOS, установленное Apple, которое запрещает выполнение динамически созданного кода на устройстве. Чтобы обеспечить соблюдение этих протоколов безопасности, Xamarin.iOS вместо этого использует компилятор AOT для компиляции управляемого кода. Это создает собственный двоичный файл iOS, который при необходимости оптимизирован с LLVM для устройств, которые можно развернуть на процессоре На основе ARM Apple. Примерная схема того, как это соответствует друг другу, показана на рисунке 1.5

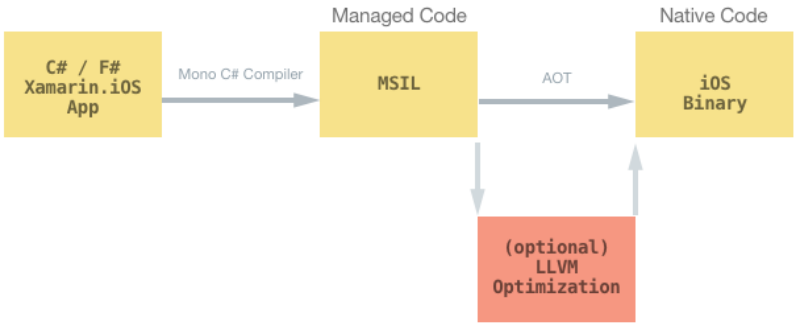

Рисунок 1.5 Компиляция приложения

В Xamarin.iOS используется два типа registrars : динамический и статический:

Динамический registrars — динамический registrar выполняет регистрацию всех типов в сборке во время выполнения. Это делается с помощью функций, предоставляемых Objective-CAPI среды выполнения. Таким образом, динамика registrar имеет более медленный запуск, но более быстрое время сборки. Это значение по умолчанию для симулятора iOS. Собственные функции (обычно в C), называемые батутами, используются в качестве реализаций методов при использовании динамической registrars. Они различаются между разными архитектурами.

Статический registrars — статический registrar создает Objective-C код во время сборки, который затем компилируется в статическую библиотеку и связывается с исполняемым файлом. Это позволяет ускорить запуск, но занимает больше времени во время сборки. Он используется по умолчанию для сборок устройств. Статический registrar также можно использовать с симулятором iOS, передав --registrar:staticmtouch атрибут в параметрах сборки проекта, как показано ниже на рисунке 1.6:

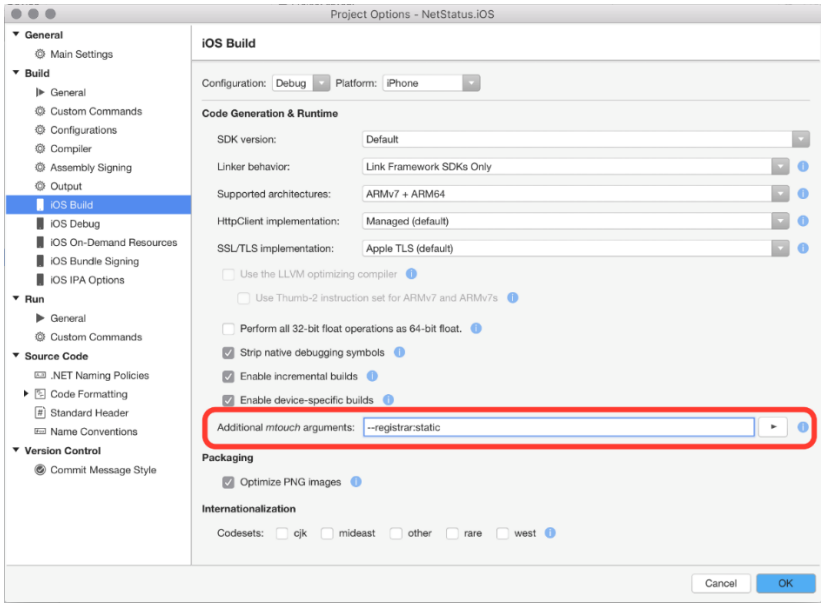

Рисунок 1.6 Пример регистра

Запуск приложения происходит следующим образом: Точка входа всех исполняемых файлов Xamarin.iOS предоставляется функцией, которая xamarin_mainинициализирует mono.

В зависимости от типа проекта выполняется следующее:

- Для обычных приложений iOS и tvOS вызывается управляемый метод Main, предоставляемый приложением Xamarin. Затем этот управляемый метод Main вызывает UIApplication.Main, который является точкой входа для Objective-C. UIApplication.Main — это привязка методаObjective-CUIApplicationMain;

Вся эта последовательность запуска компилируется в статическую библиотеку, которая затем связывается с окончательным исполняемым файлом, чтобы ваше приложение знали, как выйти из среды.

На этом этапе приложение запущено, Mono работает. Следующее, что нужно сделать, — начать добавлять элементы управления и сделать приложение интерактивным.

1.1 Выбор языка программирования

1.1.1 Сведения о языке Swift

Swift – это новый язык программирования, на котором создаются приложения для iOS, macOS, watchOS и tvOS. Впрочем, если вы пользовались C и Objective-C, то могли встречать и там многие части Swift.

Только в Swift заложены свои версии для фундаментальных типов C и Objective-C. Имеются в виду int для целых чисел, Double и Float для показателей с плавающей точкой, Bool для булевых показателей и String для текстовых объектов. Кроме того, в разделе Типы коллекций прописано, что Swift включает в себя основные три типа, а именно, Array, Set и Dictionary, причем довольно сильные их версии. В Swift (как и в C) обращение к значениям выполняется по уникальному имени, а для хранения задействуются переменные. Причем используются и те из них, значения которых меняться не могут. Они считаются константами, и в сравнении с константами в C – они мощнее. Вообще в Swift константы используются очень активно, за счет этого код получается более чистым и безопасным, если в нем есть показатели, которые не должны изменяться.

Опционные типы, работающие с отсутствующими значениями, в языке программирования Swift тоже имеются. Данные типы либо указывают на наличие некоего значения (и определяют его величину, например, x), либо говорят, здесь никакого значения нет. Это нечто схожее с использованием nil указателей в Objective-C, но тут доступна работа не только с классами, а со всеми типами. Вообще в сравнении с nil указателями в Objective-C, опциональные значения безопаснее и чётче, это, собственно, важный элемент многих мощных особенностей Swift.

1.2.2 Сведения о языке программирования Java

Java-интерпретируемый, сильно типизированный язык программирования высокого уровня. Разработан компанией Sun Microsystems. Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, с помощью виртуальной Java-машины. Дата официального выпуска — 23 мая 1995 года.

Ключевой особенностью языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. А затем этот байт-код выполняется виртуальной машиной JVM (Java Virtual Machine). В этом плане Java отличается от стандартных интерпретируемых языков как PHP или Perl, код которых сразу же выполняется интерпретатором. В то же время Java не является и чисто компилируемым языком, как C или C++.

Подобная архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, благодаря чему подобные программы без перекомпиляции могут выполняться на различных платформах - Windows, Linux, Solaris и т.д. Для каждой из платформ может быть своя реализация виртуальной машины JVM, но каждая из них может выполнять один и тот же код. Java является языком с Си-подобным синтаксисом и близок в этом отношении к C/C++ и C#.

Java является объектно-ориентированным языком. Он поддерживает полиморфизм, наследование, статическую типизацию. Объектноориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. В данной работе используется версия платформы Java SE — Java Standard Edition, основное издание Java, содержит компиляторы, API, Java Runtime Environment; подходит для создания пользовательских приложений.

1.3 Выбор СУБД

1.3.1 Основные сведения об СУБД MongoDB

В настоящее время распространены два основных типа СУБД: реляционные и нереляционные, называемые SQL и NoSQL соответственно. Поскольку СУБД типа NoSQL позволяют одновременно резервировать различные типы данных и масштабировать их, функционируя на нескольких серверах одновременно, их популярность понятна. MongoDB

Бесплатная, с открытым исходным кодом, нереляционная СУБД, MongoDB имеет и коммерческую версию. Хотя MongoDB изначально не предназначалась для обработки структурированных данных, эту СУБД можно применять для приложений, которые используют как структурированные, так и неструктурированные данные. В MongoDB базы данных подключаются к приложениям через драйверы баз данных. Несколько типов данных обрабатываются одновременно и для этой цели используется внутренний кэш.

Плюсы MongoDB:

1) Простой доступ к данным, их хранение, ввод и извлечение;
2) Простая совместимость с другими моделями данных. MongoDB легко сочетается с различными системами управления базами данных, как SQL, так и NoSQL;
3) Горизонтально масштабируемое решение. Масштабируемость – это фундаментальный аспект природы MongoDB. Этот нюанс становится еще более важным для предприятий, работающих с приложениями больших данных.

Минусы MongoBD:

1) Большое потребление памяти;
2) У MongoDB нет связей между документами и «коллекциями»;

3) Защищенность. Ориентируясь на быструю работу с данными, MongoDB, как и любой представитель типа NoSQL, не обеспечивает безопасность данных;

4) Защищенность. Ориентируясь на быструю работу с данными, MongoDB, как и любой представитель типа NoSQL, не обеспечивает безопасность данных;

1.3.2 Основные сведения об СУБД FireBase

СУБД Firebase — это серверная часть как служба, используемая для разработки веб- и мобильного программного обеспечения. Вообще, существует два варианта: база данных Firebase (обеспечивает доступ в режиме реального времени к данным, находящимся на разных платформах) и облачный магазин Firestore (обеспечивает большую масштабируемость и более сложные модели данных). Таким образом, оба решения прекрасно вписываются в сценарий, когда нужно иметь дело с большим количеством данных в режиме реального времени: изменения в БД извлекаются по мере их возникновения.

Плюсы FireBase:

1) Низкий порог вхождения. Firebase может быть отличным вариантом для быстрого старта;

2) Удобный доступ к данным, удобное управление. Через консоль обеспечивается легкого доступ к данным;

3) Проработанная документация. Документация включает в себя рекомендации, техническую документацию, ссылки на SDK, информацию об интеграции и многое другое. Сообщество развито, что позволяет легко находить ответы на возникающие проблемы.

Минусы FireBase:

1) Ограниченные возможности запроса. Проблема проявляется в ограниченности в выполнении простых запросов, так как для более сложных нет возможностей фильтрации. Это связано с тем, что вся БД представляет

собой консолидированный файл формата JSON без каких-либо опций для моделирования данных;

2) Ограниченная миграция данных. Перенос данных на другую платформу может потребовать серьёзных усилий. Отсутствуют инструменты миграции для передачи данных или установки БД проекта по умолчанию.

1.4 Выбор среды разработки мобильного приложения

Существует несколько программ для создания мобильных приложений для Android: AndroidStudio, Eclispe; и для iOS: XCode.Для анализа данных программных продуктов были выбраны следующие критерии:

- функциональность;
- удобство интерфейса;
- возможность подключения доп. Модулей;
- требовательность к системе.

1.4.1 Основные сведения об IDE Android Studio

Android Studio – продукт компании Google. Основана на программном обеспечении IntelliJ IDEA от компании JetBrains, официальное средство разработки Android приложений. Данная среда разработки доступна для Windows, OS X и Linux. Функционал данного приложения использует язык Java для написания программного кода. Разработка интерфейса производится drag–and–drop методом, но так же имеется возможность использовать XML. Интерфейс данного ПО перегружен. Интерфейс библиотек приложения имеет вид выпадающего древа и под него приходится отводить очень много места в общем интерфейсе, в противном случае, информация становится нечитаемой. Та же самая ситуация и с окном отладки.

В функционале Android Studio возможность подключения дополнительных плагинов отсутствует. Данное средство разработки очень требовательно к технической составляющей ЭВМ, по сравнению с другими средствами разработки. Минимальное количество ОЗУ требуемое для данного

продукта 2 гигабайта. Но для комфортной работы с данной программой рекомендуемое количество памяти 8 гигабайт, что не является проблемой для современных компьютеров, но на ПК старше 2014 года данная среда функционирует очень медленно, не говоря о параллельном запуске других, даже не очень требовательных, приложений. Имеет встроенный модуль для эмуляции Android-устройства. Данный эмулятор требует отдельных ресурсов, что еще сильнее повышает требовательность ПО к ЭВМ.

1.4.2 Основные сведения об IDE Xcode

Xcode — интегрированная среда разработки (IDE) компании Apple, которая предоставляет разработчикам инструменты для создания приложений под iPhone, iPad, Mac, Apple Watch и Apple TV. Последняя версия — Xcode 8 — доступна для загрузки бесплатно. Xcode запускается только на компьютерах с OS X (iMac, Macbook и Mac Mini). Годовая лицензия разработчика для публикации приложения в iTunes или Mac OS X App Store стоит $99.

Среда разработки Xcode обеспечивает эффективность работы как небольших, так и крупных девелоперских команд. В Xcode IDE используется схема разделения данных приложения Model-View-Controller (Модель-Представление-Контроллер или MVC) для сегментации каждого слоя приложения. Так проще вносить изменения в код. К примеру, слой UI разделен инструментами, такими как новый Interface Builder, с его помощью можно помещать на экран средства визуального контроля. Auto Layout позволяет динамично управлять презентацией объектов для экранов разных размеров; с помощью Storyboard удобно располагаются экраны приложения; режим Preview быстро покажет, как выглядят экраны приложения. Ни один из этих инструментов не затрагивает программный код, который вы создаете.

Прежде коды в писались языком Objective-C. В июне 2014 Apple представила Swift, новый язык для создания мобильных приложений. Это самый

быстро осваиваемый язык по сравнению с другими языками программирования. Людям из Apple потребовалось достаточно много времени, чтобы разработать Swift. Как итог всех усилий, появился язык, который разработчикам освоить намного проще, чем тот же Objective-C. К тому же допускается присутствие в одном проекте как Swift, так и Objective-C.

С Xcode могут работать индивидуальные разработчики. Программный код проверяется в репозитории Git, после чего им можно делиться с другими. Поддерживается концепт непрерывной интеграции и инструменты тестирования. В текущей версии Xcode также присутствует инструмент Test Assistants — обеспечивает корректность кода и тестов; инструмент для тестирования Test Navigator; поддержка для ботов в Xcode Server, которые запускаются после проверки кода в элементе, есть средства проверки производительности, асинхронности и UI-тестов.

Таким образом, анализ средств разработки мобильных приложений можно свести в таблицу, оценивая рассмотренные критерии по пятибалльной шкале (таблица 1).

Таблица 1 – Сравнительный анализ средств разработки мобильных приложений.

Функциональность	AndroidStudio	Xcode
Удобство интерфейса	5	3
Возможность подключения дополнительных модулей	4	5
Требовательность к системе	4	5
Наличие встроенных компонентов тестирования приложения	3	1

На основе анализа средств разработки была выбрана Android Studio, так как Android Studio менее требовательна к системным ресурсам компьютера и не менее мощная, чем остальные среды разработки.

1.5 Выбор графических редакторов для создания прототипа

Для создания прототипа мобильного приложения следует определиться с графическим редактором, на нынешнее время самыми популярными из них являются Figma и Sketch.

Figma -графический редактор для создания прототипов сайтов и приложений. Над проектом одновременно могут работать несколько человек, так как можно выдать доступ на редактирование или комментирование любому. В фигме обычно создают прототипы сайтов и приложений, иллюстрации, векторную графику, рисуют элементы интерфейса. Ещё здесь создают макеты сайтов для тильды: есть возможность импортировать дизайн.

Sketch - это программа для векторного дизайна для Mac, ориентированная на дизайн экрана. Она используется главным образом дизайнерами, которые создают веб-сайты, значки и пользовательские интерфейсы для настольных и мобильных устройств. Мощное и простое в использовании приложение Sketch для начинающих и профессиональных дизайнеров отрасли позволяет сосредоточиться на том, что они делают лучше всего - дизайне.

Функциональность Sketch расширена благодаря фантастическим сторонним разработчикам, которые создали множество плагинов, чтобы ускорить рабочий процесс проектирования. Sketch также легко интегрируется со многими приложениями для прототипирования и совместной работы.

1.6 Вывод по теоретической части

В теоретической части курсового проекта были проанализированы: предметная область для создания нативного мобильного приложения зоомагазина, инструментарий для реализации приложения. Средой разработки в ходе анализа была выбран Android Studio, имеющая ряд преимуществ по сравнению с аналогами. Приложение будет реализовано на языке Java, которые больше всех подходит для создания мобильных приложений. Прототип будет составлен в Figma, который в свою очеред является удобным веб-редактором, с большим количеством функций и возможностей, а также обладает интуитивно понятным интерфейсом.

Проанализировав несколько СУБД, для создания базы данных приложения была выбрана Firebase, которая полностью удовлетворяет запросам для разработки мобильного приложения – магазина для животных.

ГЛАВА II. РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ МАГАЗИНА ДЛЯ ЖИВОТНЫХ.

2.1 Прототипированное проектирование программного модуля.

В веб-графическом редакторе Figma был реализован прототип мобильного приложения интернет магазина и показан на рисунке 2.1.

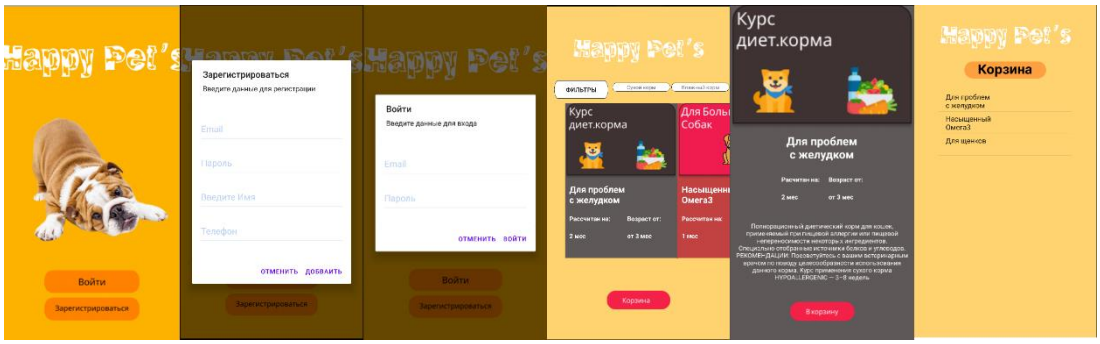

Рисунок 2.1 Прототип приложения

Прототип состоит из 6 окон: главного экрана где пользователю предлагается войти в приложение или зарегистрироваться; два всплывающих диалоговых окна для регистрации/авторизации, куда пользователь вводит данные; каталог магазина; карточка товара; корзина.

2.2 Разработка мобильного приложения

2.2.1 Создание классов приложения

Для начала написания приложения необходимо создать классы. Java позволяет создавать классы которые представляют объекты. Созданы классы: «MainActivity»; «KatalogActivity»; «CoursePageActivity»; «OrderPageActiviy». (рис.2.2)

Рисунок 2.2 Классы приложения

Классы в данном приложении отвечают за: модуль авторизации/регистрации; работу с СУБД; модуль фильтров; модуль корзины; модуль курсов.

2.2.2 Создание окон приложения

Для полноценной работы приложения недостаточно создать только классы, кроме классов необходимо создать окна приложения. Окна отвечают за пользовательский интерфейс. Все элементы в макете простроены с использование иерархии объектов View.

Сформированы следующие окна: «activity_main»; «activity_katalog»; «activity_course_page»; «activity_register»; «activity_order»; «activity_signin»; «category_item»; «course_item». (рис 2.3)

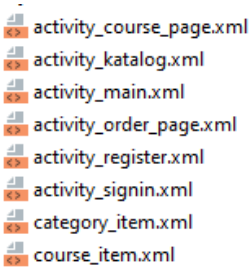

Рисунок 2.3 Окна приложения

2.2.3 Создание моделей приложения

Модель класса – класс в Java, в котором мы определяем все поля, которые будем использовать в программе, а именно определяются сеттер-геттер, конструктор. Модели изображены на (рис.2.4).

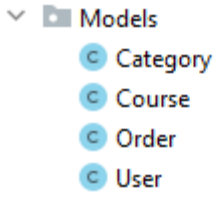

Рисунок 2.4 Модели классов приложения

В данном приложении модели класса необходимы для описания товаров, курсов, пользователей и категорий.

2.2.4 Создание адаптеров приложения

Для заполнения RecyclerView – компонент используемый для отображения элементов списка, требуется создать адаптеры. Adapter – объект класса, который отвечает за извлечение данных из массива данных и за создание объекта View, на основе этих данных.

Созданные адаптеры показаны на (рис.2.5).

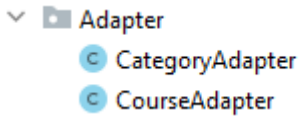

Рис 2.5 Адаптеры приложения

В этом приложении адаптеры используются для отображения данных о курсах и категориях, взятых из массива.

2.2.5 Разработка модуля «Регистрация/авторизация»

Для приложения по продаже товаров необходима система учета пользователей. Для того чтобы воспользоваться магазином ему нужно для начала зарегистрироваться, а затем авторизоваться. Система регистрации будет создана в классе «MainActivity». Для реализации этого метода было решено создать отдельное окно, в котором будет реализован дизайн диалогового окна регистрации.(рис.2.6).

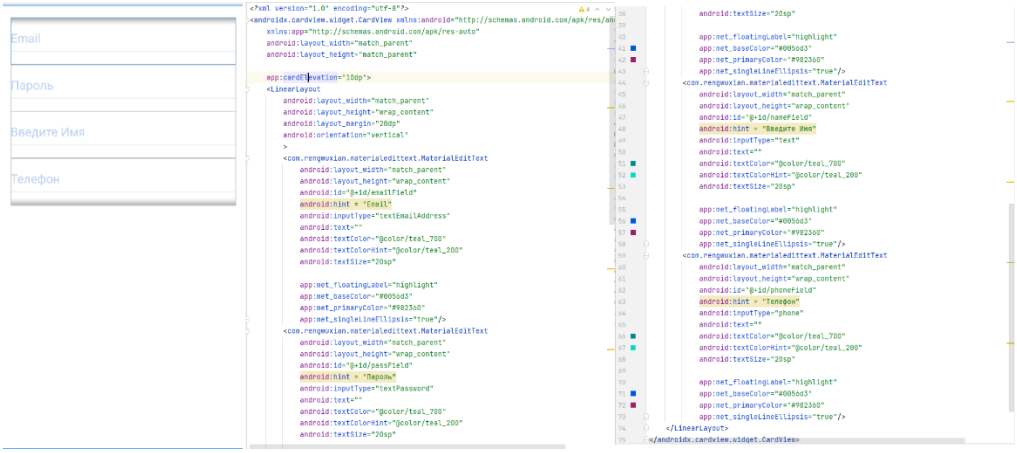

Рисунок 2.6 Реализация диалогового окна регистрации

В окне был создан компонент cardElevation, который отображает окно поверх другого окна. Далее, были созданы текстовые компоненты materialedittext, которые были импортированы ранее. В них пользователь будет записывать данные которые затем передадутся в базу данных. В след за диалоговым окном регистрации создано также отдельное всплывающее диалоговое окно авторизации. (рис.2.7)

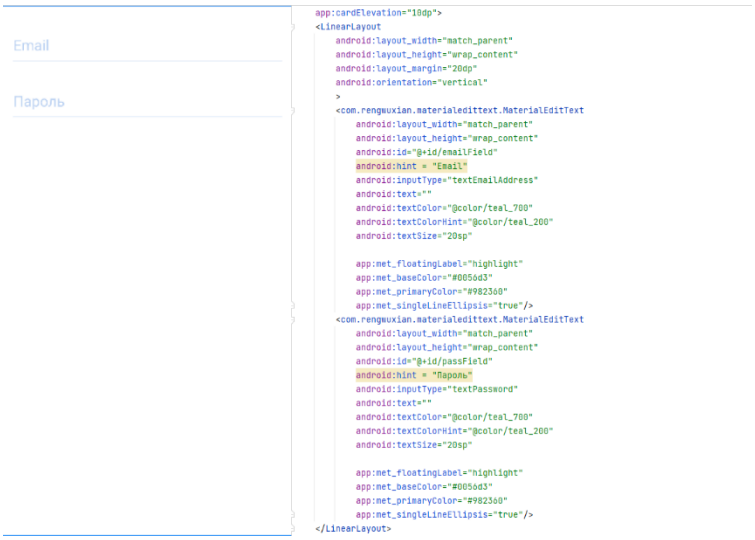

Рисунок 2.7 Диалоговое окно авторизации

Данное окно было создано аналогично окну регистрации.

Затем создано окно главного экрана «activity_main», где пользователь будет проходить процедуру регистрации/авторизации. (рис 2.8)

Рисунок 2.8 Создание экрана входа в приложение

На этом экране созданы кнопки, которые затем инициализируются в классе «MainActivity».

В классе «MainActivity» создадим и инициализируем переменные, которые будут необходимы для реализации модуля, а также обработчики нажатий. (рис 2.9)

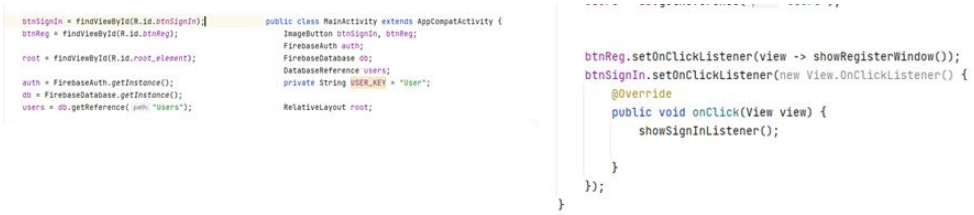

Рисунок 2.9 Инициализация переменных и создание обработчика нажатий

Затем были созданы методы, которые будут отвечать за вывод диалогового окна для регистрации и авторизации, соответственно. При нажатии пользователем соответствующих кнопок. (рис 2.10)

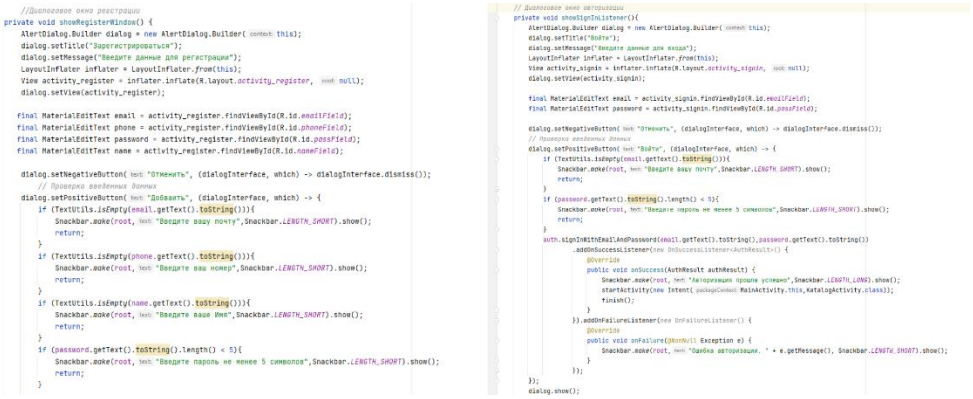

Рисунок 2.10 Создание методов вывода диалогового окна

Метод проверяет введенные данные, поля не должны быть пустыми. Также пароль должен быть не менее 5 символов. При регистрации по нажатии кнопки «Добавить» вызывается функция auth.createUserWithEmailAndPassword, которая добавляет данные пользователя в базу. (рис 2.11)

Рисунок 2.11 Добавление данных пользователя в базу данных

Создадим тестовую учетную запись, чтобы проверить созданный метод и проверим ее наличие в базе данных. (рис 2.12)

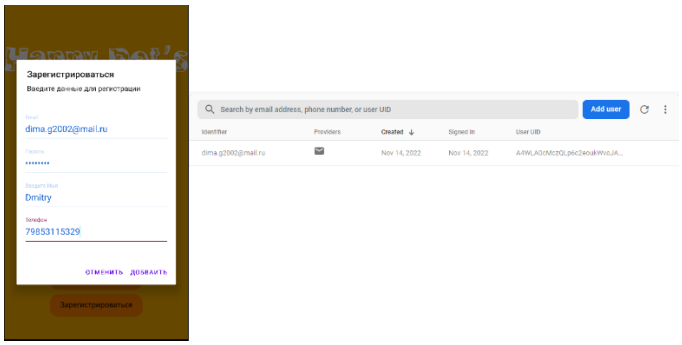

Рисунок 2.12 Создание учетной записи

Учетная запись успешно создалась и отобразилась в базе данных. Попробуем используя эти данные войти в приложение. (рис 2.13)

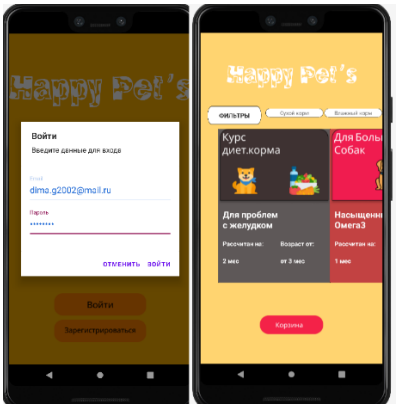

Рисунок 2.13 Вход в учетную запись

Как можем наблюдать, мы успешно авторизовались и нам открылся доступ к магазину.

2.2.6 Разработка модуля «Карточки питания»

После успешной авторизации пользователь попадает на экран Каталога, где он может выбрать курс питания для своего питомца.

Было создано отдельно окно с шаблоном карточки товара, который затем используется для автоматического заполнения из списка с помощью адаптера «CourseAdapter». (рис 2.14)

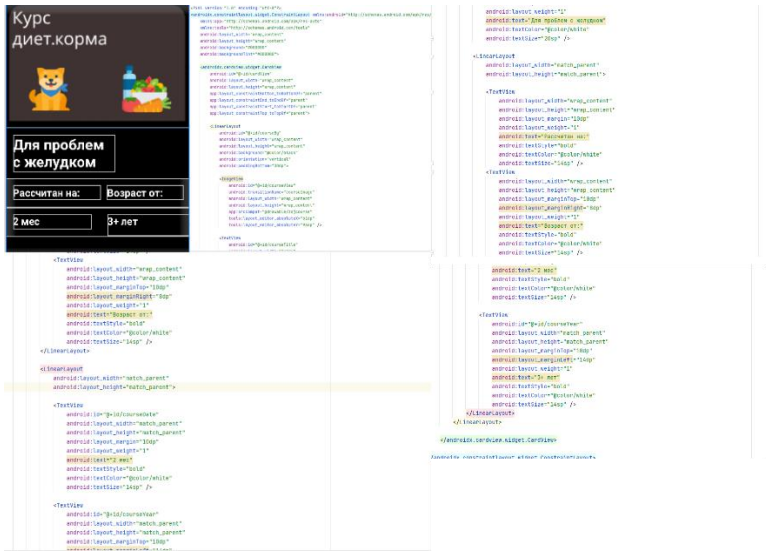

Рисунок 2.14 Создание окна карточки

Затем в модуле «Course» описываем карточку товара. Указывая поля: id, title, age, color, text. Далее создаем публичный метод, в котором прописываем при помощи конструктора геттеры-сеттеры, из которых адаптер в будущем будет брать данные. (рис 2.15)

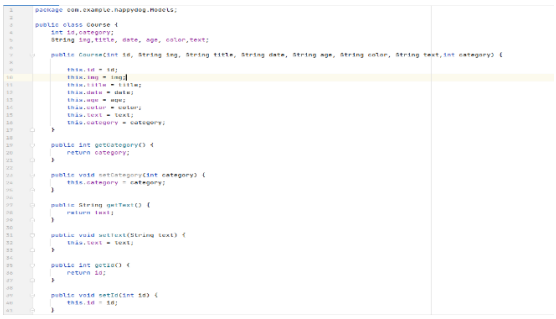

Рисунок 2.15 Класс-модель «Course»

После чего был создан класс-адаптер «CourseAdapter». Адаптер наследует данные из класса «RecyclerView», который обращается к классу «Adapter». Далее создаем вложенный класс «CourseViewHolder» и реализовываем все необходимые методы. (рис 2.16)

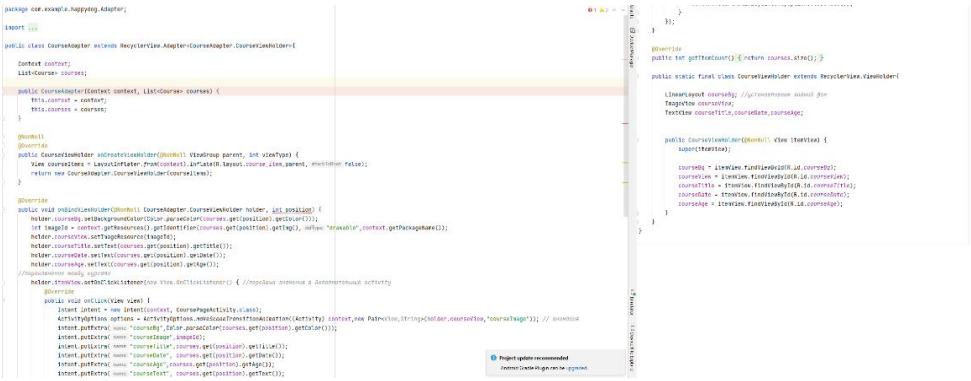

Рисунок 2.16 Класс-адаптер «CourseAdapter»

Затем в «KatalogActivity» создаем список, состоящий из различных категорий. (рис 2.17)

Рисунок 2.17 Список из которого в карточки идут значения

Затем устанавливаем правила, что «RecyclerView» прокручивается горизонтально. (рис 2.18)

```
private void setCourseRecycler(List<Course> courseList) {
    RecyclerView.LayoutManager layoutManager = new LinearLayoutManager( context: this,RecyclerView.HORIZONTAL, reverseLayout: false);

    courseRecycler = findViewById(R.id.courseRecycler);
    courseRecycler.setLayoutManager(layoutManager);

    courseAdapter = new CourseAdapter( context: this, courseList);
    courseRecycler.setAdapter(courseAdapter);

}
```

Рисунок 2.18 Правила для «RecyclerView»

Далее необходимо при нажатии на карточку открывать отдельное окно где будет указана информация про ту или иную карточку. Для этого было создано окно «activity_course_page» по аналогии с «course_item». (рис 2.19)

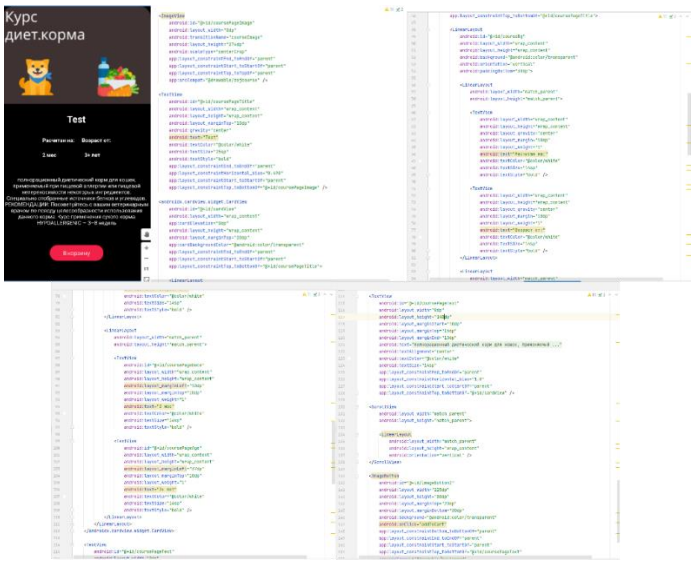

Рисунок 2.19 Окно «activity_course_page»

Затем добавляем обработчик нажатия для нажатия на товар и передачу значений между окнами карточек, дополнительно добавив анимацию. (рис 2.20)

Рисунок 2.20 Обработчик нажатия

Затем устанавливаем значения в объекты дизайна в классе-адаптере «CoursePageActivity». (рис 2.21)

Рисунок 2.21 Передача значений в дизайн

Проверив модуль, ошибок не обнаружено все карточки отображаются корректно. (рис 2.22)

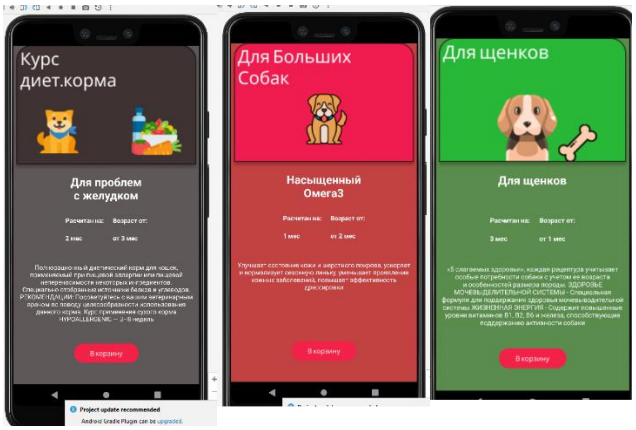

Рисунок 2.22 Проверка модуля «Карточки питания»

2.2.7 Разработка модуля «Фильтры»

В приложении была реализована возможность фильтрации курсов питания по категориям: сухой корм, влажный корм.

Создан Класс-модель «Category» в котором были прописаны геттеры-сеттеры. После чего был прописан класс-адаптер «CategoryAdapter», в котором были прописаны обработчик нажатия и указаны элементы с которыми он будет работать в дизайне. (рис 2.23)

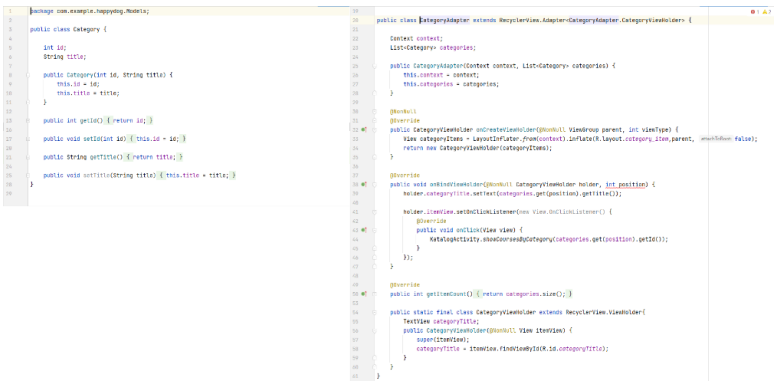

Рисунок 2.23 Модуль «Category» и класс «CategoryAdapter»

Затем были проинициализированы и созданы необходимые списки и классы, с помощью который будет выводиться только те курсы, которые отфильтровал пользователь. (рис 2.24)

Рисунок 2.24 Реализация отображения курсов

В классе «KatalogActivity» создадим списки и, в которых будут храниться значения и методы, которые помогут отобразить корректно курсы по категориям. (рис 2.25)

Рисунок 2.25 Реализация отображения курсов по категориям

2.2.8 Разработка модуля «Корзина»

Для разработки корзины было реализовано окно «activity_order_page» (рис 2.26)

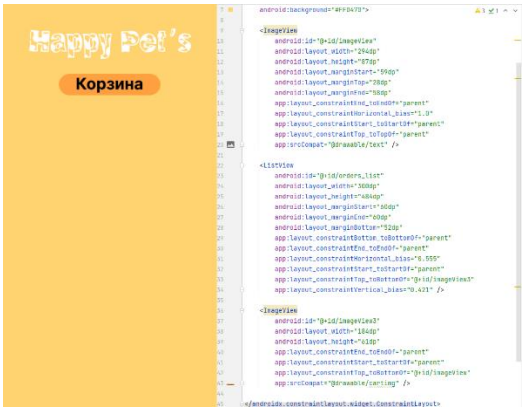

Рисунок 2.26 Окно «activity_order_page»

Затем в модуле «Order» создан метод, который содержит в себе список товаров добавленных в корзину. Затем создан метод в классе «CoursePageActivity» в котором реализован обработчик нажатия, который будет добавляет товар в корзину собирая данные из «из CourseAdapter». (рис 2.27)

Рисунок 2.27 Реализация модуля «Корзина»

Затем нам надо прописать вывод элементов, для этого в «OrderPageActivity» создаем список, к которому укажем адаптер и указываем для него стандартный дизайн AndroidStudio, далее добавляем список теъ элементов которые будут выведены в ListView, затем прописываем логику которая будет выводить названия курсов. (рис 2.29)

Рисунок 2.29 Готовый модуль «Корзина»

Логика метода такова, для начала получается письменное значение курсов, затем проверяем id элемента, и если он принадлежит корзине, то название перебираемого значения перемещаем в courseTitle.

2.2.9 Вывод по практической части курсовой работы

В процессе выполнения практической части курсовой работы по разработке мобильного приложения для интернет-магазина для животных «Happy Pet`s». Функционал приложения полностью удовлетворяет требованиям современного мобильного приложения. В ходе разработки были разработаны различные окна, классы, модели, адаптеры, которые без ошибок функционируют. В рамках данной работы были созданы модули, такие как: модуль регистрации/авторизации; модуль фильтрации; модуль корзины; модуль карточек питания.

ЗАКЛЮЧЕНИЕ

Целью данной курсовой работы являлось создание актуального мобильного приложения интернет-магазина для животных на базе Android.

В ходе выполнения курсовой работы были реализованы следующие задачи:

- Изучены методы создания различных модулей для реализации функционала современного приложения;
- Был создан прототип интерфейса приложения по которому оно было создано
- Прописаны модули позволяющие вести учет пользователей, перемещать товары в корзину, создавать карточки питания;
- Были проведены модульные и интеграционные тестирования приложения на проверку работоспособности как отдельных модулей, так и всего приложения в целом;
- Скомпилировано приложения.

В ходе выполнения курсовой работы было создано полноценное приложение с приятным и понятным интерфейсом, где нет ничего лишнего. Дополнительно изучены методы работы с графическими редакторами и методы проектирования приложения в целом.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Физические ресурсы:

1. *Гриффитс Д.* Head First. Программирование для Android. – 2-е изд.: Питер, 2018. – 912 с.
2. *Дейтел Х., Дейтел П., Уолд А.*: Android для разработчиков. – 3-е издание, Питер 2016. – 512 с.
3. *Романчик В., Блинов И.*: Java. Методы программирования. – Четыре четверти, 2013. – 898 с.
4. *Кэти С., Бэйтс Б.*: Изучаем Java. – Эксмо, 2020. – 720 с.
5. *Ретабоуил С.*: Android NDK. Руководство для начинающих. – МДК Пресс, 2016. – 518 с.

Электронные ресурсы:

1. https://code.tutsplus.com/ru/tutorials/get-started-with-firebase-for-android--cms-27248 - работа с FireBase.
2. https://drach.pro/blog/hi-tech/item/197-popular-nosql-dbms-2022 - обзор СУБД.
3. https://ru.hexlet.io/blog/posts/yazyk-programmirovaniya-java-osobennosti-populyarnost-situatsiya-na-rynke-truda - обзор языка Java
4. https://proglib.io/p/osnovy-java-za-30-minut-samouchitel-dlya-nachinayushchih-2021-09-06 - основы языка программирования Java.
5. https://translated.turbopages.org/proxy_u/en-ru.ru.0b8f216f-63721e76-d0e0ed85-74722d776562/https/www.geeksforgeeks.org/a-complete-guide-to-learn-android-studio-for-app-development/ - Основы разработки приложений в Android Studio