Частное учреждение образования

«Колледж бизнеса и права»

Заместитель директора

по учебной работе

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.В. Малафей

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ «Авто Подбор»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Руководитель проекта ()

Учащийся (А.Д Янушевич)

2020

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

1

Разраб.

Янушевич А.Д

Провер.

1

Т.контр.

1

Н.контр.

1

Утверд.

1

*Мобильное приложение «Авто Подбор»*

Лит

Листов

81

КБиП

[Введение 4](#_Toc41062343)

[1 Объектно-ориентированный анализ и проектирование приложения 6](#_Toc41062344)

[1.1 Назначение и цели создания мобильного приложения 6](#_Toc41062345)

[1.2 Проектирование модели 6](#_Toc41062346)

[2 Проектирование мобильного приложения](#_Toc41062347) 9

[2.1 Требования к мобильному приложению](#_Toc41062348) 9

[2.2 Структура мобильного приложения](#_Toc41062349) 9

[2.3 Проектирование макета мобильного приложения](#_Toc41062350) 11

[2.4 Программно-технические средства, необходимые для разработки приложения](#_Toc41062351) 14

[2.5 Защита и сохранность данных](#_Toc41062352) 15

[2.6 Организация и ведение информационной базы](#_Toc41062353) 16

[3 Реализация мобильного приложения](#_Toc41062354) 20

[3.1 Описание используемых функций и процедур 2](#_Toc41062355)0

[4 Описание мобильного приложения](#_Toc41062356) 23

[4.1 Общие сведения](#_Toc41062357) 23

[4.2 Функциональное назначение 2](#_Toc41062358)3

[4.3 Описание разделов приложения 2](#_Toc41062359)3

[5 Методика испытаний](#_Toc41062360) 30

[5.1 Технические требования 3](#_Toc41062361)0

[5.2 Функциональное тестирование 3](#_Toc41062362)0

[6 Применение](#_Toc41062363) 38

[6.1 Назначение программы](#_Toc41062364) 38

[7 Охрана труда и окружающей среды](#_Toc41062365) 39

[7.1 Правовые, нормативные, социально-экономические и организационные вопросы охраны труда](#_Toc41062366) 39

[7.2 Разработка мер по нормализации и защиты от ультрафиолетового излучения](#_Toc41062367) 41

[7.3 Пожарная безопасность](#_Toc41062368) 43

[7.4 Охрана окружающей среды](#_Toc41062369) 44

[8 Экономический раздел 4](#_Toc41062370)6

[8.1 Технико-экономическое обоснование разработки программного средства 4](#_Toc41062371)6

[8.2 Составление плана по разработке программного средства 4](#_Toc41062372)6

[8.3 Определение цены программного средства 4](#_Toc41062373)6

[8.4 Экономическая эффективность разработки](#_Toc41062374) 50

[Заключение 5](#_Toc41062375)1

[Список информационных источников 5](#_Toc41062376)3

[Приложение А Текст программы 5](#_Toc41062377)4

# Введение

Целью курсового проекта является разработка мобильного приложения «Авто Подбор». Программное средство должно содержать удобный интрефейс и функционал для подбора автомобиля по заданым характеристикам.

Данная программа служит для решения следующих задач: ведение базы данных для хранения информации о зарегистрированых пользователях и внесённых в базу автомобилях, характеристиках автомобиля, закладках; отображение автомобилей, их сортировка и представление полной информации об автомобиле(тип топлива, привод, каробка передач, трансмисия и т.д.),регистрации и авторизации пользователей; панель администратора, для добовления новых автомобилей.

Для достижения цели курсового проекта, необходимо решить следующие задачи:

* выполнить объектно-ориентированный анализ и проектирование системы, результатом которой будет модель системы;
* определить вычислительную систему, необходимую для создания программного средства;
* по модели выполнить проектирование задачи;
* разработать мобильное приложение;
* описать созданное мобильное приложение;
* выбрать методику испытаний;
* описать процесс тестирования применения;
* привести примеры области.

Решение поставленных задач отражено в пояснительной записке, которая состоит из шесть разделов и содержит необходимую и достаточную информацию по использованию данного программного средства.

В первом разделе «Объектно-ориентированный анализ приложения» раскрывается организационная сущность задачи, описывается предметная область и круг задач, которые должны быть автоматизированы. Описывается задача, перечисляются основные функции программы. Строится информационная модель, отражающая сущности задачи, их свойства и взаимосвязи. Описываются новые возможности программы, а также ее отличия от предыдущих версий.

Во втором разделе «Проектирование мобильного приложения» перечисляются требования к внешнему виду приложения, контенту и мобильному устройству, структуре, макету, защите данных, приводится обоснование выбранного инструмента разработки, описание и взаимосвязь таблиц.

В третьем разделе «Реализация мобильного приложения» производится описание функций мобильного приложения, их взаимосвязи и листинги всех модулей.

В четвертом разделе «Описание мобильного приложения» представлены общие сведения о программном средстве и его функциональном назначении, описание разделов приложения.

В пятом разделе «Методика испытаний» описываются требования к техническим средствам для проведения испытаний, требования к характеристикам программы применительно к условиям эксплуатации, требования к информационной и программной совместимости. Предоставляются результаты функционального тестирования.

Шестой раздел «Применение» предназначен для описания сведений о назначении программного средства и области его применения. В этом разделе приводится структура справочной системы, а также методика ее использования.

В заключении описывается выполнение поставленной задачи, степень соответствия проектных решений заданию, причины несоответствия, если таковые имеются.

В приложении А представлен текст программы.

Графическая часть содержит следующие диаграммы: вариантов использования, классов, последовательности, деятельности и компонентов.

# Объектно-ориентированный анализ и проектирование приложения

## Назначение и цели создания мобильного приложения

Рассмотрим предметную область задачи.

В жизни людей часто случается ситуация, что есть много детских вещей, с которых выросли дети и которые занимают пространство, но выкинуть эти вещи не хочется. Так и начали свое развитие барахолки, так как вещи, которые вам уже не нужны, могут быть полезными другим людям, которые готовы отдать за это деньги. Барахолки существовали и до создания компьютеров и мобильных приложений. С развитием информационных технологий начали создаваться сайты с барахолками, а позднее и мобильные приложения. Мобильные приложения существенно упрощают процесс продажи своих вещей.

Программное средство должно содержать удобный функционал для продажи, покупки и просмотра подержанных товаров.

Потенциальной аудиторией приложения будут лица от достигших совершеннолетнего возраста до людей в возрасте, так как ассортимент товаров очень обширен.

Должны быть автоматизированы все основные функции для работы сервером, а именно занесение в базу данных информации о пользователе, проверка наличия и правильности введенных данных пользователя при авторизации, вывод основной информации о пользователе, создание объявлений, модерация объявлений, просмотр пользователями своих объявлений в структурированном виде, добавление объявлений в закладки, возможность поиска с критериями по объявлениям, возможность просмотра полной информации об объявлении, продавце и его активных объявлениях, возможность фильтра объявлений по категориям, возможность ручной и автоматической деактивации и удаления объявлений, возможность редактирования текущих объявлений, возможность редактирования своего профиля.

Для разработки серверной части необходимо использовать язык программирования PHP, он отличается своей простотой для создания бек-энда. Для реализации мобильного приложения необходимо использовать языки Kotlin и Java. Kotlin является официальным языком для разработки android приложений, а Java является одним из самых популярных языков.

Из аналогов для всех товаров (не только для детей) можно выделить такие мобильные приложения как «Куфар», «Авито». В сравнении с «Куфар», приложение «Каталог товаров для детей» будет иметь более удобную навигацию, информативный внешний вид объявлений и не будет иметь рекламы. Приложение «Авито» разработано для пользователей проживающих в Российской Федерации. От приложения «Куфар» оно отличается более широким функционалом и современным внешним видом.

## Проектирование модели

Цель моделирования данных состоит в обеспечении разработчика информационной системы концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

Наиболее распространенным средством моделирования данных являются диаграммы «сущность-связь» (ERD). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных. Нотация ERD была впервые введена П. Ченном и получила дальнейшее развитие в работах Баркера. Диаграмма «сущность-связь» представлена на рисунке 1.1.

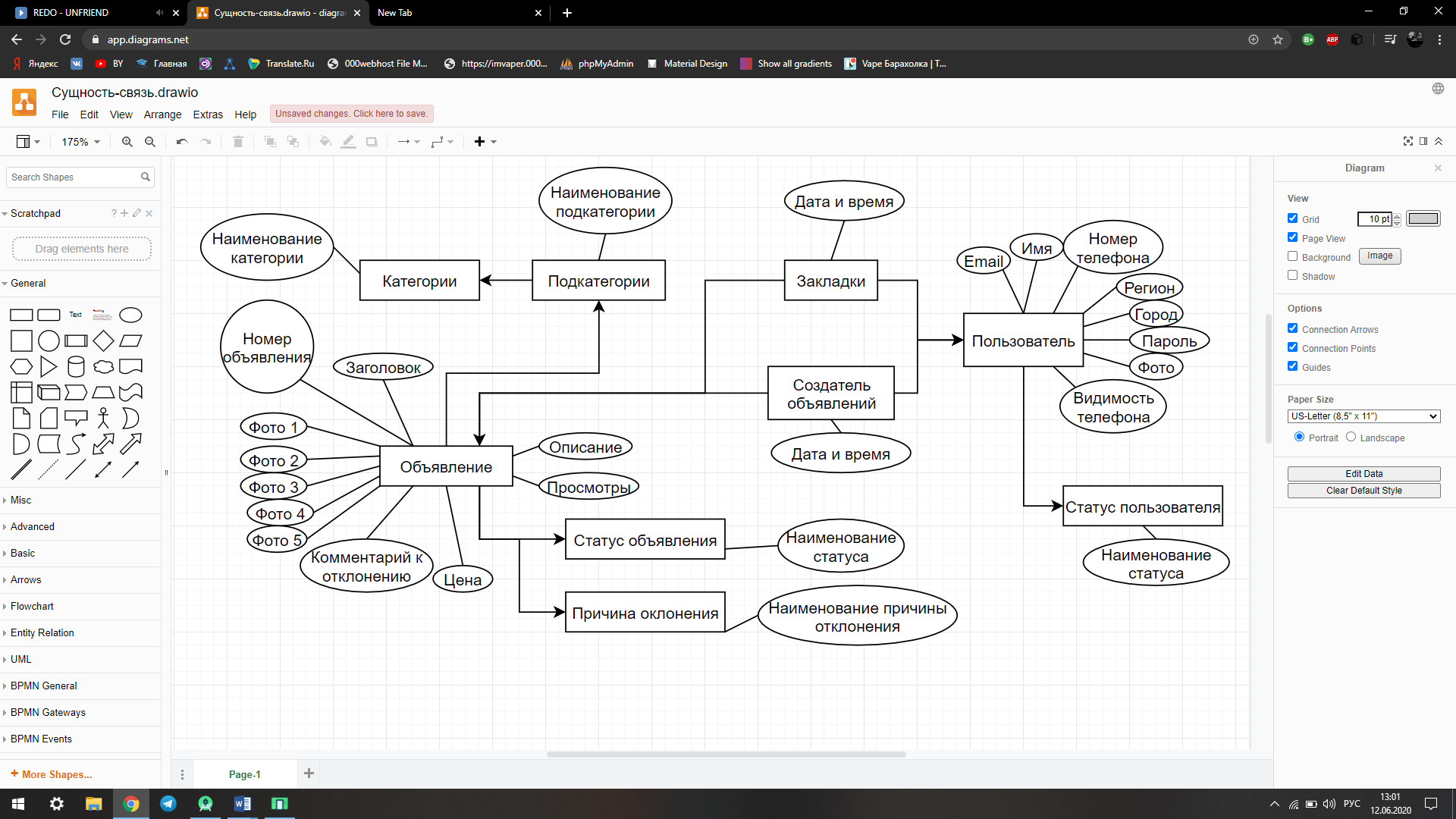


Рисунок 1.1 – Диаграмма «Сущность-связь»

Исходя из исследования предметной области, можно выделить следующие сущности разработки: «Пользователь», «Объявление», «Создатель объявлений», «Закладки», «Категории», «Подкатегории», «Причина отклонения», «Статус пользователя», «Статус объявления».

Для сущности «Пользователь» можно выделить следующие атрибуты:

* + имя;
  + email;
  + номер телефона;
  + регион;
  + город;
  + пароль;
  + фото;
  + видимость телефона.

Для сущности «Объявление» атрибутами будут являться:

* + номер объявления;
  + заголовок;
  + описание;
  + фото1;
  + фото2;
  + фото3;
  + фото4;
  + фото5;
  + цена;
  + просмотры;
  + комментарий к отклонению.

Для сущности «Создатель объявлений» атрибутом будет являться «Дата и время».

Для сущности «Закладки» атрибутом будет являться «Дата и время».

Для сущности «Категории» атрибутом будет являться «Наименование категории».

Для сущности «Подкатегории» атрибутом будет являться «Наименование подкатегории».

Для сущности «Причина отклонения» атрибутом будет являться «Наименования причины отклонения».

Для сущности «Статус пользователя» атрибутом будет являться «Наименование статуса пользователя».

Для сущности «Статус объявления» атрибутом будет являться «Наименование статуса объявления».

Суть диаграммы вариантов использования состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования.

Варианты использования описывают не только взаимодействия между пользователями и сущностью, но также реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами сущности. Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы. Множество вариантов использования в целом должно определять все возможные стороны ожидаемого поведения системы.

К основным функциям разрабатываемой программы относятся следующие функции:

* создание объявлений;
* регистрация;
* авторизация;
* разграничение доступа к данным;
* мои объявления;
* закладки;
* настройки;
* техническая поддержка;
* просмотр полной информации по объявлению.

Структура мобильного приложения представляет собой наглядное описание разделов приложения и их расположение. Основным блоком является главная страница приложения, с которой начинается навигация приложения. Структура мобильного приложения представлена в графической части на листе 1.

Диаграммы вариантов использования служит для отражения отношений между актерами и прецендентами. Основная цель использования таких диаграмм – визуализация возможностей пользователя в программном средстве. Диаграмма вариантов использования представлена в графической части на листе 2.

Центральное место в объектно-ориентированном программировании (ООП) занимает разработка логической модели системы в виде диаграммы классов. Нотация классов в языке унифицированного моделирования (UML) проста и интуитивно понятна. Схожая нотация применяется и для объектов — экземпляров класса, с тем различием, что к имени класса добавляется имя объекта и вся надпись подчеркивается.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений.

Диаграмма классов представляет собой некоторый граф, вершинами кото­рого являются элементы типа «классификатор» и которые связаны различными типами структурных отношений. Диаграмма классов построена с помощью плагина Simple Uml, который встроен в Android Studio. Поэтому диаграмму классов принято считать графическим представлением таких структурных взаимосвязей логической модели системы, которые не зависят от времени. Диаграмма классов представлена в графической части на листе 3.

Для моделирования взаимодействия объектов в UML используется диаграмма последовательности. Диаграмма последовательности представлена в графической части на листе 4.

Все предыдущие диаграммы отражали концептуальные аспекты построения модели системы и относились к логическому уровню представления и оперировали понятиями, которые не имеют самостоятельного материального воплощения. Для описания реальных сущностей предназначен другой аспект модельного представления, а именно физическое представление модели.

Для представления структуры компонентов программы используются диаграмма развертывания. Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения, она содержит графические изображения процессов и связей между ними.

Диаграмма развертывания представлена в графической части на листе 5.

# Проектирование мобильного приложения

## Требования к мобильному приложению

Основными требованиями к внешнему виду приложения является использование ярких цветов, современных элементов дизайна с применением библиотеки Material Design. Суть библиотеки состоит в том, чтобы упростить создание современного и лаконичного дизайна мобильного приложения. Логотип приложения необходимо реализовать с помощью шрифта Aldrich, все остальное текстовое оформление необходимо выполнить с помощью шрифта Roboto-Regular.

Основными минимальными требованиями, выдвигаемыми к аппаратному обеспечению мобильного телефона, являются:

* процессор с частотой не менее 1.2 ГГц;
* оперативная память 1 Гбайт и более;
* внутренняя память не менее 8 Гбайт.

Мобильный телефон должен быть с установленной операционной системой (ОС) Android версии не ниже 5.0.0, так как разработка мобильного приложения не подразумевает версии ниже из-за усложнения процесса разработки.

Основными требованиями к подаче объявления будут являться: соответствие фотографии, заголовка и описания, обязательное присутствие хотя бы одной фотографии товара, запрещено упоминание о материалах, действиях, веществах или устройствах запрещенных в РБ, наличие корректной цены товара, указание его категории и подкатегории, заголовка и описания.

Все страницы приложения должны иметь полосу прокрутки для устройств с маленькой диагональю экрана.

## Структура мобильного приложения

Логическая структура приложения представлена на рисунке 2.1

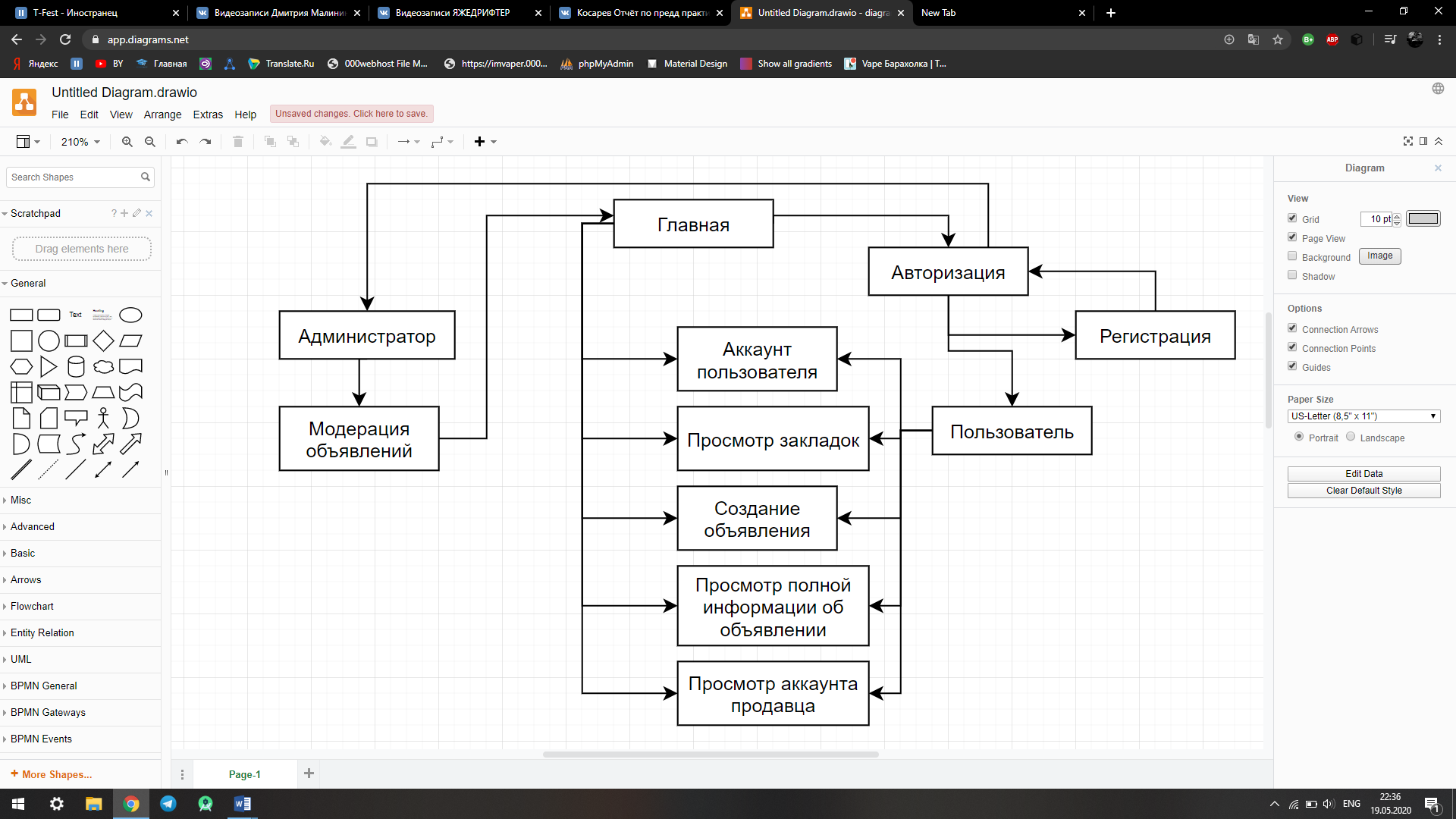


Рисунок 2.1 – Логическая структура приложения

Физическая структура мобильного приложения представлена на рисунке 2.2.



Рисунок 2.2 – Физическая структура мобильного приложения

Главная страница приложения будет содержать кнопку меню для навигации по приложению, ленту с объявлениями, фильтр по категориям, поиск и кнопку для создания нового объявления.

Структура главной страницы приложения представлена на рисунке 2.3.

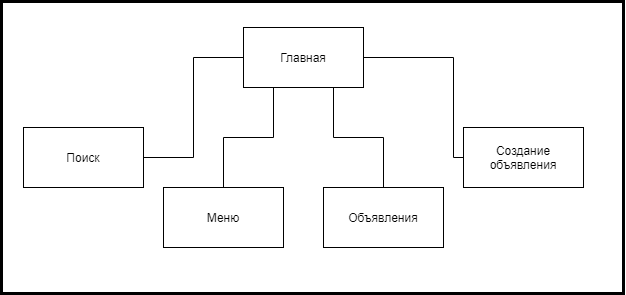


Рисунок 2.3 – Структура главной страницы приложения

Главная страница является основной точкой входа в мобильное приложение.

Назначение главной страницы:

* обеспечивать доступ ко всем объявлениям приложения;
* обеспечивать доступ к углубленному поиску объявлений;
* обеспечивать доступ к сортировке объявлений для большего удобства;
* обеспечивать доступ к созданию нового объявления;
* обеспечивать доступ ко всем основным разделам приложения.

Главная страница должна содержать идентификационные данные.

К идентификационным относятся данные:

* заголовок;
* навигация по разделам;
* кнопка для создания объявлений.

На странице должны присутствовать ссылки на следующие разделы:

* «Аккаунт/Вход»;
* «Мои объявления»;
* «Закладки»;
* «Техническая поддержка»;
* «Настройки».

У мобильного приложения должна быть возможность открытия на любом устройстве с любым соотношениям сторон. У приложения должна быть масштабируемость для открытия и удобного пользования на всех версиях Android, начиная с версии 5.0.0.

В мобильном приложении будут находится следующие разделы: «Все объявления», «Мои объявления», «Закладки», «Техническая поддержка», «Настройки», «Аккаунт», «Авторизация», «Регистрация», «Полное объявления», «Продавец», «Поиск», «Создание объявления».

Раздел «Все объявления» - это начальная страница при входе в приложение, она будет содержать фильтр по категориям, вкладку «Поиск», кнопку создания нового объявления, ленту с объявлениями и элементы меню.

В разделе «Аккаунт» будет собрана информация о пользователе.

В разделе «Мои объявления» будут собраны все объявления пользователя в сгруппированном виде.

В разделе «Закладки» будут содержатся объявления, добавленные пользователем в закладки.

В разделе «Техническая поддержка» будут находится контакты для связи с разработчиком.

В разделе «Настройки» можно будет изменить тему приложения.

В разделе «Создание объявления» можно будет создать новое объявление, после чего оно будет отправляться на модерацию.

В разделе «Поиск» можно будет указать критерии для поиска объявлений.

В разделе «Авторизация» пользователь сможет войти в свой аккаунт, а если его нет, то перейти в раздел «Регистрация».

В разделе «Регистрация» можно будет создать новый аккаунт.

В разделе «Полное объявление» будет отображаться полная информация об объявлении, все фотографии и кнопка для перехода на страницу пользователя.

В разделе «Продавец» можно будет увидеть данные продавца для связи с ним.

## Проектирование макета мобильного приложения

У мобильного приложения должна быть возможность открытия и масштабируемости на любом Android-устройстве, с любым разрешением экрана и соотношением сторон, с версией Android не менее 5.0.0.

На главной странице будут находиться такие компоненты как: кнопка меню, объявления, поиск, фильтр по категориям и кнопка создания нового объявления. Макет главной страницы представлен на рисунке 2.4

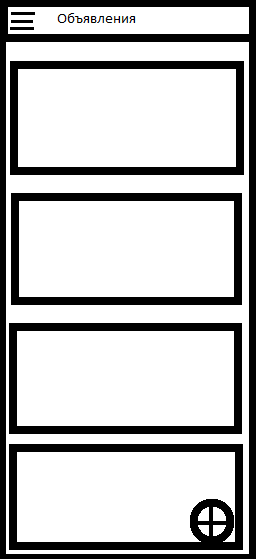


Рисунок 2.4 – Макет главной страницы приложения

На странице авторизации будут находиться такие компоненты как: кнопка входа, кнопка перехода на страницу регистрации, поле ввода логина, поле ввода пароля. Макет страницы авторизации представлен на рисунке 2.5

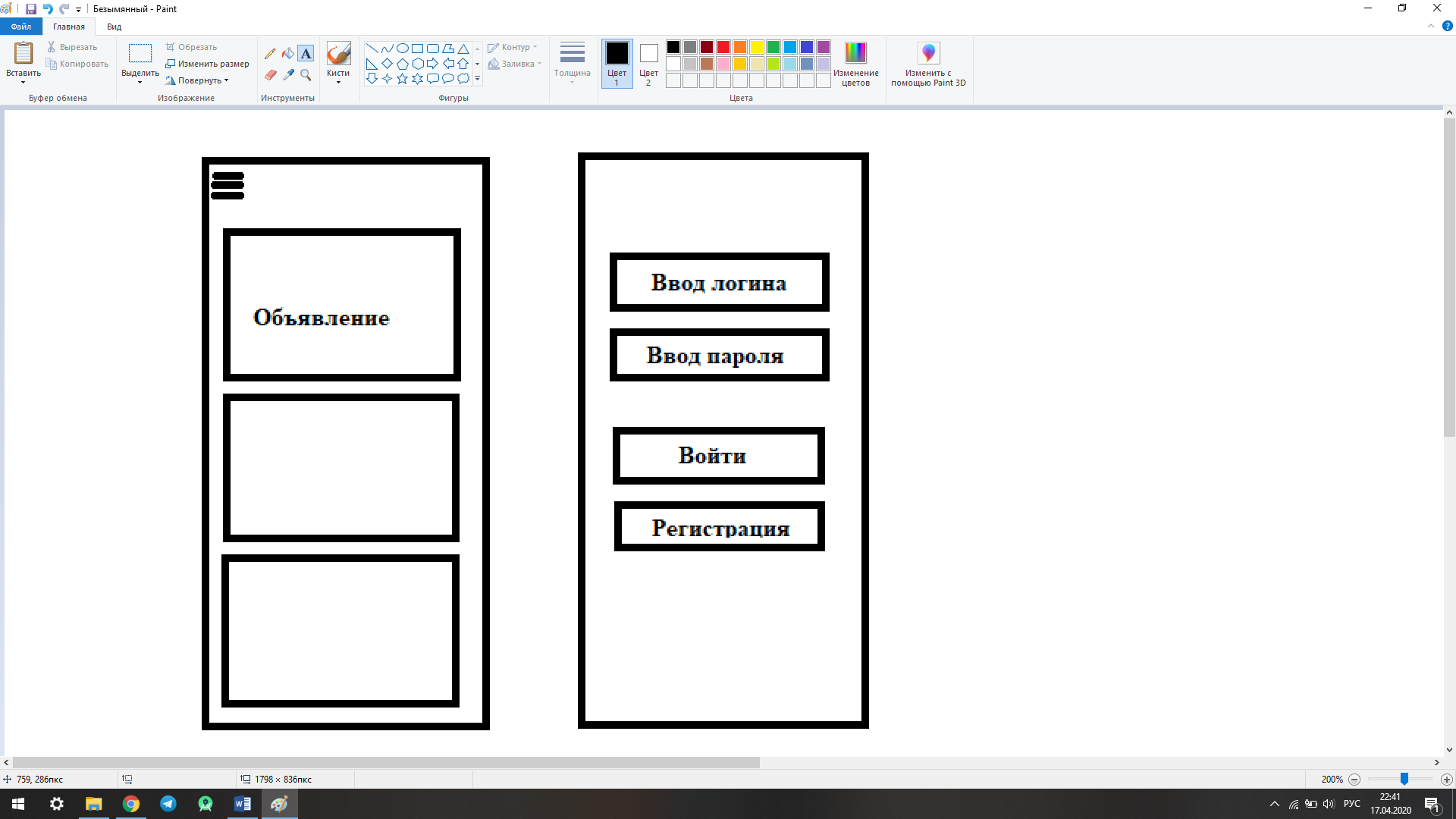


Рисунок 2.5 – Макет страницы авторизации

На странице регистрации будут находиться такие компоненты как: кнопка регистрации, поля для ввода данных необходимых для регистрации. Макет страницы регистрации представлен на рисунке 2.6

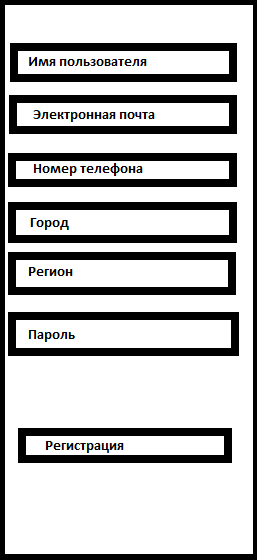


Рисунок 2.6 – Макет страницы регистрации

На странице аккаунта пользователя будут находиться такие компоненты как: текстовые поля данных пользователя, фото пользователя, кнопка выхода из аккаунта. Макет страницы аккаунта пользователя представлен на рисунке 2.7

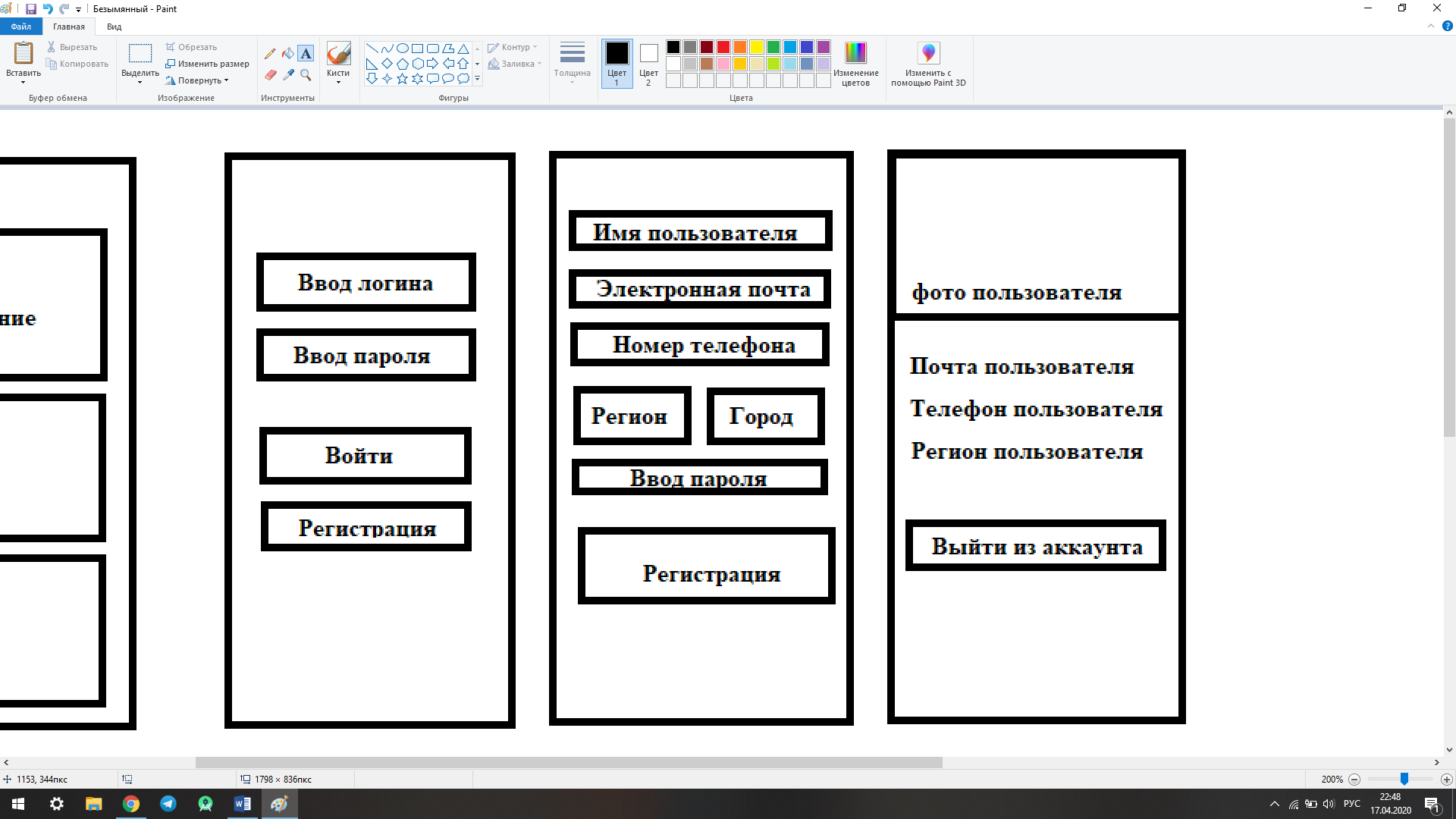


Рисунок 2.7 – Макет страницы аккаунта пользователя

На странице полного объявления будут находиться такие компоненты как: текстовые поля с данными об объявлении, фото объявления, кнопка перехода на страницу продавца. Макет страницы полного объявления представлен на рисунке 2.8

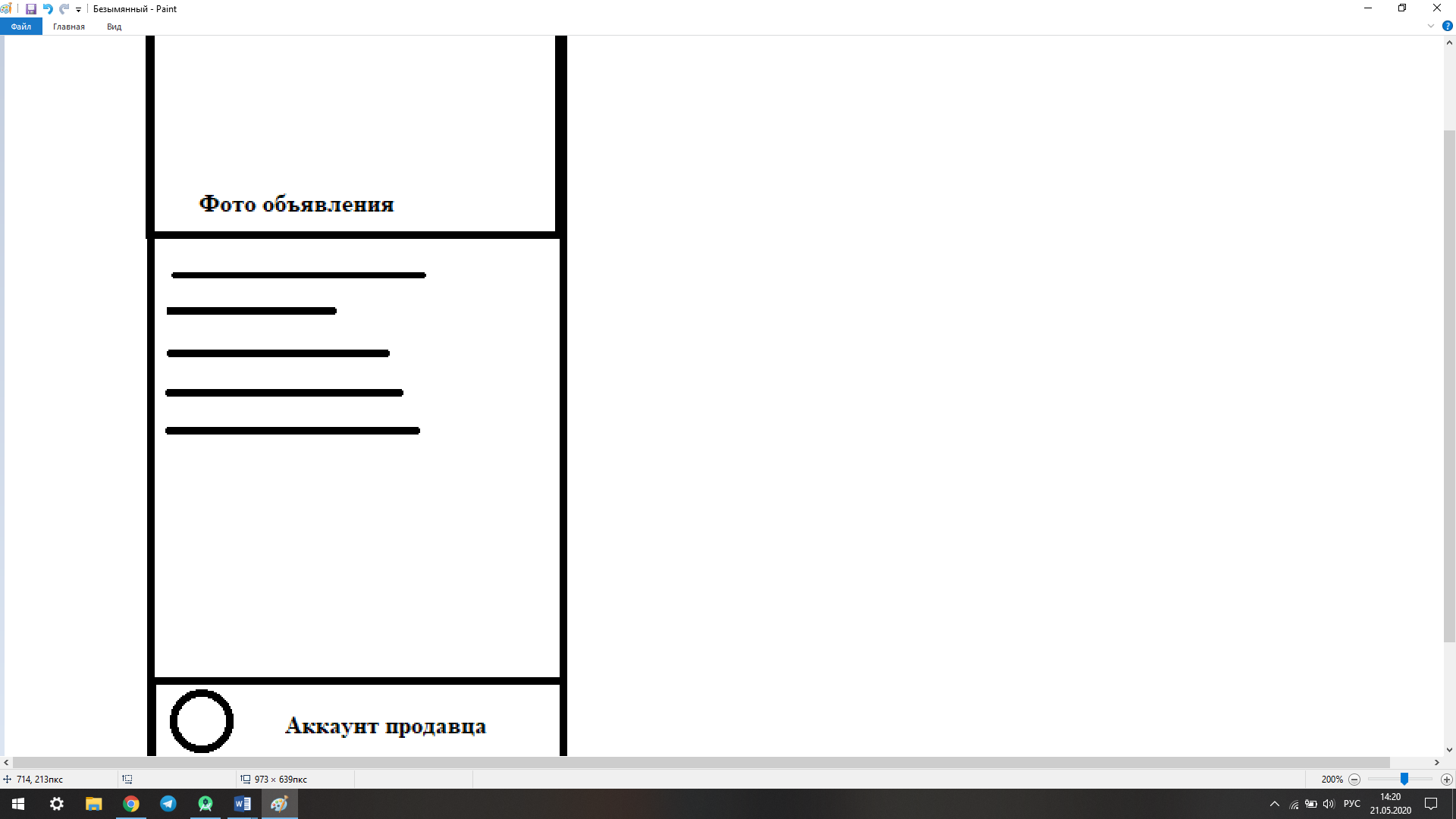


Рисунок 2.8 – Макет страницы полного объявления

На странице продавца будут находиться такие компоненты как: текстовые поля с данными о продавце, фото продавца и его активные объявления. Макет страницы продавца представлен на рисунке 2.9

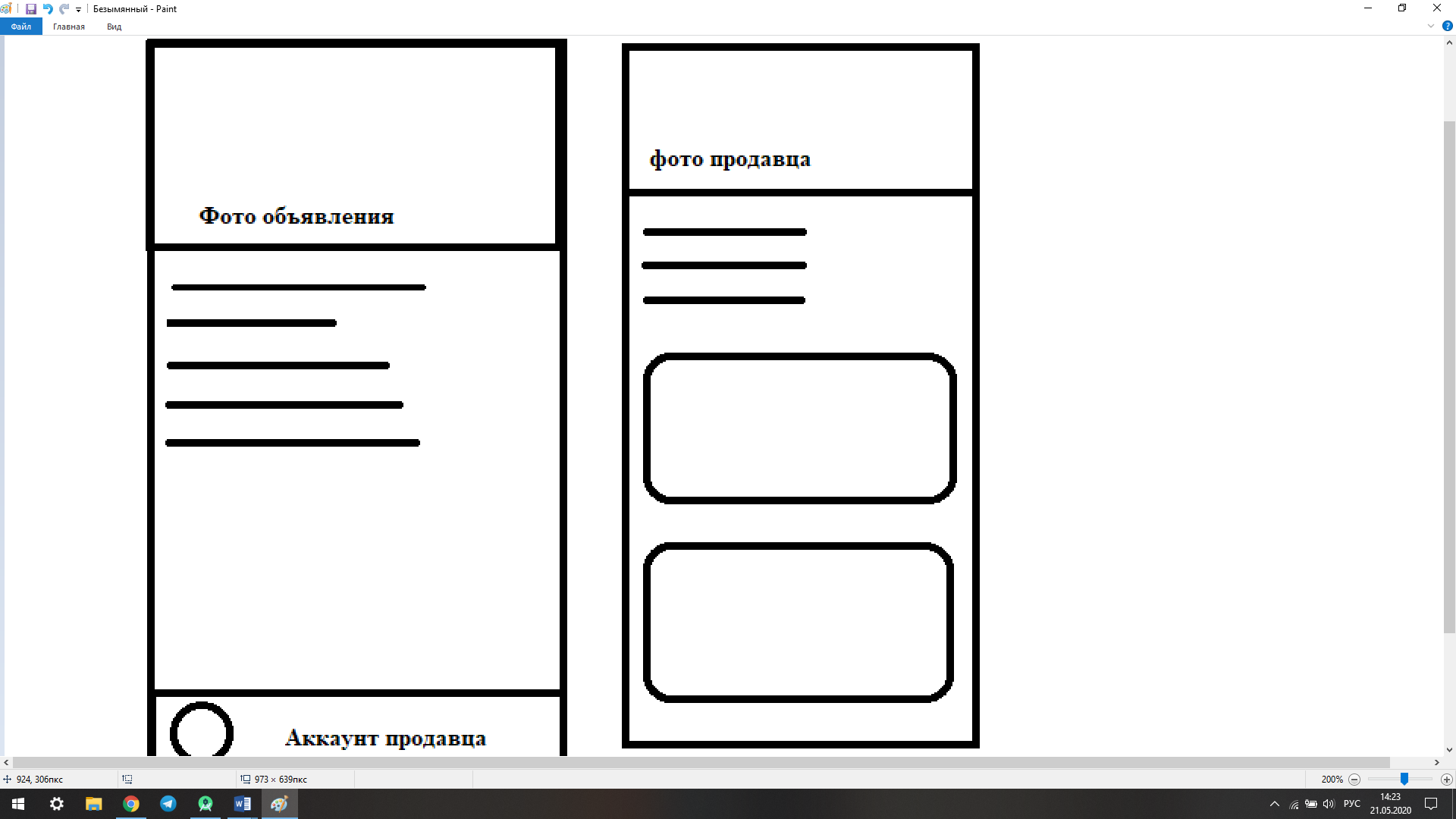


Рисунок 2.9 – Макет страницы продавца

На странице мои объявления будут находиться такие компоненты как меню выбора объявлений, текстовые поля с количеством объявлений с разными статусами и сами объявления. Макет страницы мои объявления представлен на рисунке 2.10

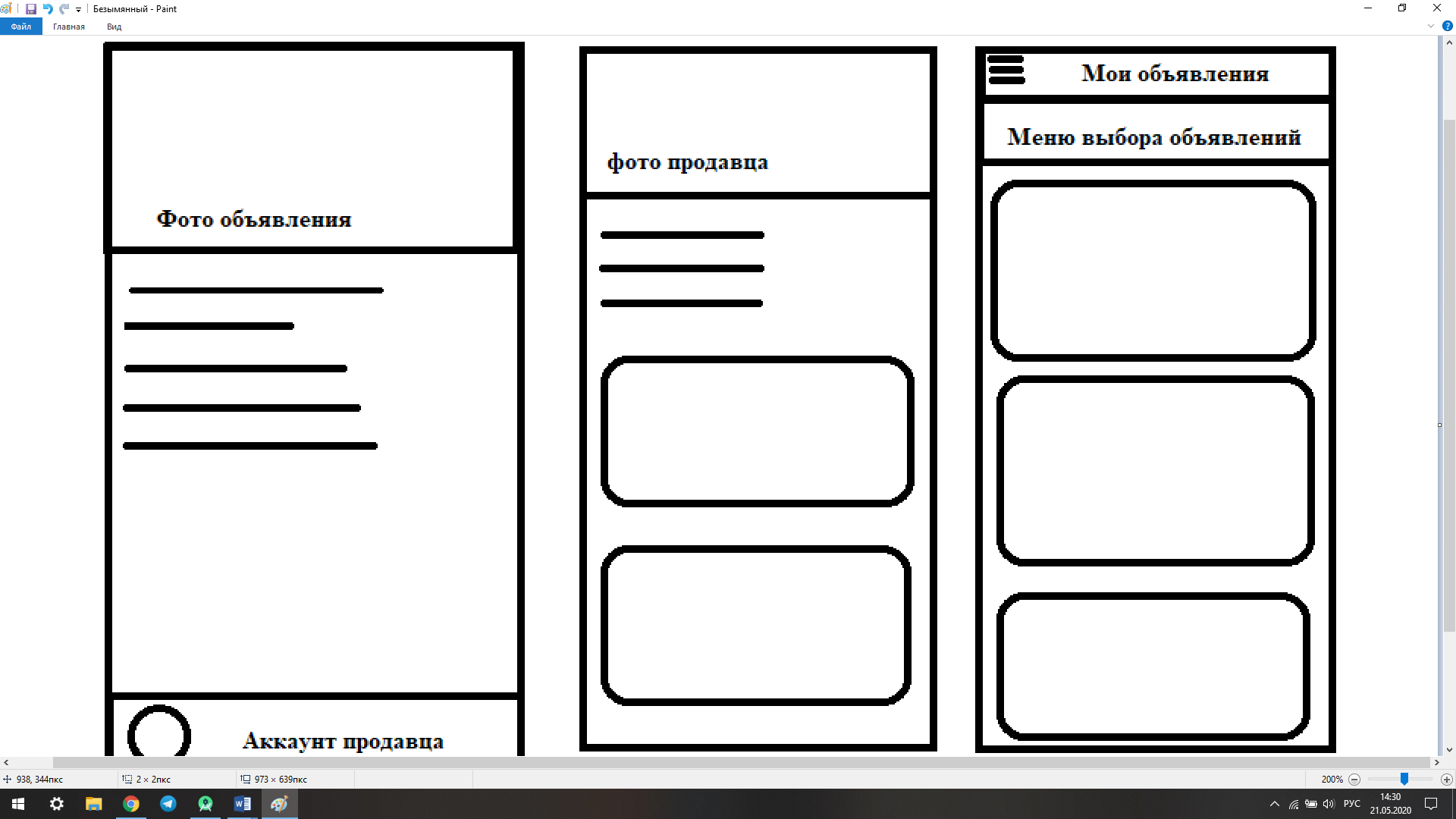


Рисунок 2.10 – Макет страницы мои объявления

## Программно-технические средства, необходимые для разработки приложения

Инструментами разработки будут являться:

* операционная система Windows 10;
* среда разработки Android Studio;
* языки программирования Java, Kotlin;
* программа для построения блок-схем Draw.io;
* веб-интерфейс для администрирования СУБД MySQL;
* скриптовый язык общего назначения PHP.

Мобильное приложение будет разрабатываться и тестироваться под управлением операционной системы Windows 10. Данная операционная система является оптимальным решением для предприятия любого размера. Данная версия операционной системы Windows сочетает в себе преимущества предыдущих версий ОС этого производителя. Это делает Windows 10 наиболее подходящей операционной системой для настольных компьютеров, применяемых в корпоративной среде. Независимо от того, где устанавливается Windows 10 – на одном компьютере или в масштабе небольшой рабочей группы, – эта система повышает вычислительные возможности предприятия, одновременно сокращая совокупную стоимость программного обеспечения всех настольных компьютеров [15].

Android Studio — интегрированная среда разработки (IDE) для работы с платформой Android.

Android Studio, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, — официальное средство разработки Android приложений. Java – сильно типизированный [объектно-ориентированный язык программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), разработанный компанией [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems)[4].

Программы на Java [транслируются](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в [байт-код Java](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4_Java), выполняемый [виртуальной машиной Java](https://ru.wikipedia.org/wiki/Java_Virtual_Machine) (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как [интерпретатор](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80). Достоинством подобного способа выполнения программ является полная независимость байт-кода от [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и [оборудования](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0), что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание [6].

Kotlin — статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine и разрабатываемый компанией JetBrains. Также компилируется в JavaScript и в исполняемый код ряда платформ через инфраструктуру LLVM. Kotlin является официальным языком программирования под Android.

Авторы ставили целью создать язык более лаконичный и типобезопасный, чем Java, и более простой, чем Scala. Следствием упрощения по сравнению со Scala стали также более быстрая компиляция и лучшая поддержка языка в IDE. Язык полностью совместим с Java, что позволяет java-разработчикам постепенно перейти к его использованию; в частности, в Android язык встраивается с помощью Gradle, что позволяет для существующего android-приложения внедрять новые функции на Kotlin без переписывания приложения целиком [4].

PhpMyAdmin — веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL. PhpMyAdmin позволяет через браузер и не только осуществлять администрирование сервера MySQL, запускать команды SQL и просматривать содержимое таблиц и баз данных. Приложение пользуется большой популярностью у веб-разработчиков, так как позволяет управлять СУБД MySQL без непосредственного ввода SQL команд [5].

PHP — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания Api мобильных приложений [5].

## Защита и сохранность данных

В качестве защиты данных выступает авторизация путём ввода логина и пароля. В дальнейшем возможна доработка этих средств, введение аутентификации.

Данные будут храниться в базе данных, доступ к которой имеется только со стороны разработчиков. Эти данные защищены шифрованием со стороны хоста.

При регистрации пользователя приложение хеширует пароль и загружает его на сервер, далее, при авторизации хеши введенного пароля и пароля на сервере сравниваются, в результате равенства хешей пользователь получает доступ к своему аккаунту. В случае взлома базы данных на сервере злоумышленник не увидит реальные пароли пользователей. По статистике Android-приложения взламываются гораздо реже web-сайтов, исходя из этого было принято решение не применять дополнительные меры по защите данных.

Приложение будет иметь три разрешения, этими разрешениями являются: разрешение на доступ в интернет (для доступа к базе данных, это разрешение будет выдаваться автоматически), разрешение на доступ к камере (должно выдаваться по желанию пользователя), разрешение на запись информации в хранилище телефона (будет выдаваться автоматически при выдаче разрешения на доступ к камере).

## Организация и ведение информационной базы

Организация данных подразумевает создание модели данных, главными элементами которой являются сущности и их связи.

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы - атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация - это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

Структура базы данных разрабатываемого программного средства включает девять таблиц.

Структура данных таблиц, и их краткое описание приводится в таблицах 2.1-2.9.

Таблица «user» хранит информацию о пользователях. Структура приведена в таблице 2.1.

Таблица 2.1 – Структура таблицы «user»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер, байт | Описание поля |
| id | Int | 4 | Идентификатор пользователя |
| nickname | Varchar | 18 | Имя пользователя |
| email | Varchar | 50 | Email пользователя |
| phoneNumber | Varchar | 15 | Номер телефона пользователя |
| region | Varchar | 20 | Регион пользователя |
| town | Varchar | 20 | Город пользователя |
| password | Varchar | 18 | Пароль пользователя |
| image | Varchar | 200 | Ссылка на фото пользователя |
| user\_status\_id | Int | 4 | Идентификатор статуса пользователя |
| isPhoneHide | Tinyint | 1 | Флаг, который показывает скрывать ли телефон пользователя |

Таблица «user\_status» хранит информацию о статусах пользователя. Структура приведена в таблице 2.2.

Таблица 2.2- Структура таблицы «user\_status»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер, байт | Описание поля |
| id | Int | 4 | Идентификатор статуса |
| status\_name | Varchar | 25 | Наименование статуса пользователя |

Таблица «ad» хранит информацию обо всех объявлениях пользователей. Структура приведена в таблице 2.3.

Таблица 2.3- Структура таблицы «ad»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер, байт | Описание поля |
| id | Int | 4 | Идентификатор объявления |
| title | Varchar | 30 | Заголовок объявления |

Продолжение таблицы 2.3

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер, байт | Описание поля |
| description | Varchar | 150 | Описание объявления |
| image1 | Varchar | 300 | Ссылка на фото объявления |
| image2 | Varchar | 300 | Ссылка на фото объявления |
| image3 | Varchar | 300 | Ссылка на фото объявления |
| image4 | Varchar | 300 | Ссылка на фото объявления |
| image5 | Varchar | 300 | Ссылка на фото объявления |
| price | Int | 4 | Стоимость товара в объявлении |
| views | Int | 4 | Количество просмотров объявления |
| ad\_status\_id | Int | 4 | Идентификатор статуса объявления |
| subcategory\_id | Int | 4 | Идентификатор подкатегории объявления |
| reject\_reason\_id | Int | 4 | Идентификатор причины отклонения |
| reject\_message | Varchar | 150 | Комментарий к отклонению |

Таблица «creator» хранит информацию о создании объявлений. Структура приведена в таблице 2.4.

Таблица 2.4- Структура таблицы «creator»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер, байт | Описание поля |
| id | Int | 4 | Идентификатор создания объявления |
| dateTime | Datetime | 8 | Дата и время создания объявления |
| user\_id | Int | 4 | Идентификатор пользователя, создавшего объявления |
| ad\_id | Int | 4 | Идентификатор созданного объявления |

Таблица «ad\_status» хранит информацию о статусах объявлений. Структура приведена в таблице 2.5.

Таблица 2.5- Структура таблицы «ad\_status»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер, байт | Описание поля |
| id | Int | 4 | Идентификатор статуса объявления |
| ad\_status\_name | Varchar | 25 | Наименование статуса объявления |

Таблица «categories» хранит информацию о категориях объявлений. Структура приведена в таблице 2.6.

Таблица 2.6- Структура таблицы «categories»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер, байт | Описание поля |
| id | Int | 4 | Идентификатор категории |
| category\_name | Varchar | 40 | Наименование категории объявления |

Таблица «bookmarks» хранит информацию об объявлениях, добавленных пользователем в закладки. Структура приведена в таблице 2.7.

Таблица 2.7- Структура таблицы «bookmarks»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер, байт | Описание поля |
| id | Int | 4 | Идентификатор закладки |
| bookmark\_datetime | Datetime | 8 | Дата и время добавления объявления в закладки |
| ad\_id | Int | 4 | Идентификатор объявления, добавленного в закладки |
| user\_id | Int | 4 | Идентификатор пользователя, добавившего объявление в закладки |

Таблица «subcategories» хранит информацию о подкатегориях объявлений. Структура приведена в таблице 2.8.

Таблица 2.8- Структура таблицы «subcategories»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер, байт | Описание поля |
| id | Int | 4 | Идентификатор подкатегории |
| subcategory\_name | Varchar | 40 | Наименование подкатегории объявлений |
| category\_id | Int | 4 | Идентификатор подкатегории объявления |

Таблица «reject\_reasons» хранит информацию о подкатегориях объявлений. Структура приведена в таблице 2.9.

Таблица 2.9- Структура таблицы «reject\_reasons»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер, байт | Описание поля |
| id | Int | 4 | Идентификатор причины отклонения |
| reason\_name | Varchar | 40 | Наименование причины отклонения |

Структура базы данных представлена на схеме на рисунке 2.10.

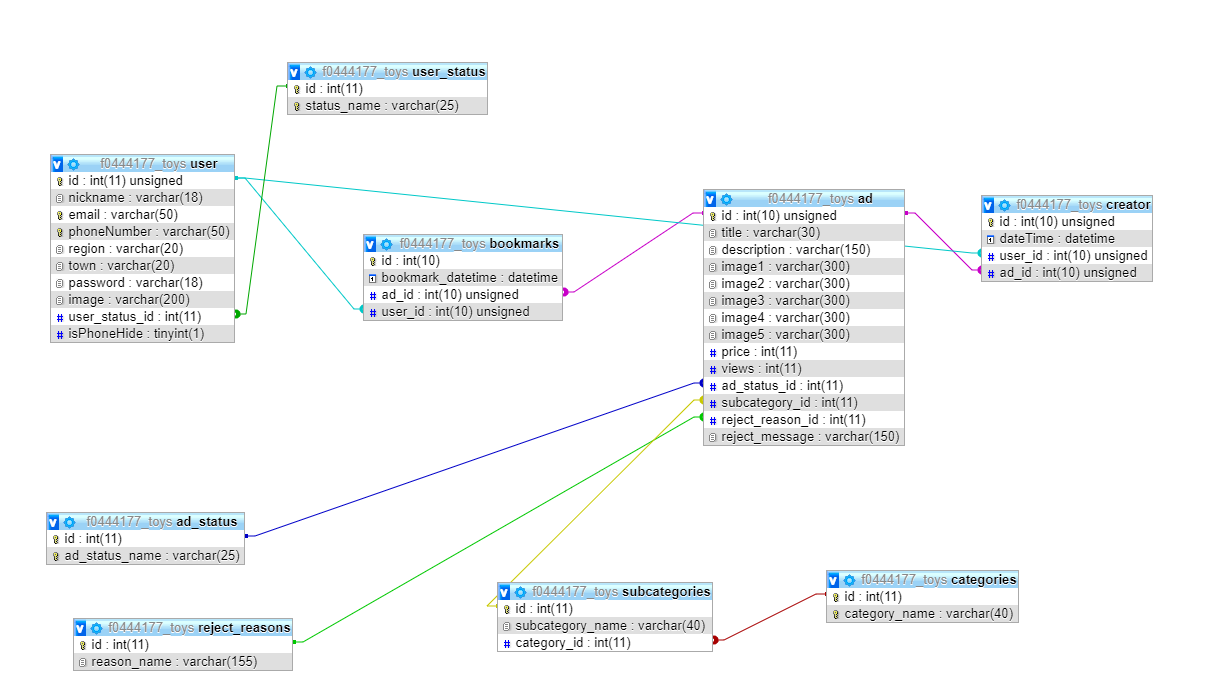


Рисунок 2.10 – Логическая схема базы данных

# Реализация мобильного приложения

## Описание используемых функций и процедур

Административная часть представлена модерацией контента.

Административная часть разрабатываемого приложения состоит из разработки интерфейса и функциональной части, реализующей возможность одобрения и отклонения создаваемых объявлений. При отклонении объявления администратор обязан указать причину отклонения и по желанию добавить комментарий к отклонению.

Назначение администратора происходит путем регистрации и смены статуса пользователя непосредственно через базу данных с простого пользователя на администратора.

Клиентская часть – сторона пользователя. В ней представлен весь функционал приложения, кроме административной части.

В клиентской части производится:

* валидация данных, введенных пользователем;
* проверка на заполнение;
* отображение визуальных эффектов интерфейса;
* отображение причины отклонения для отклоненного объявления;
* поиск объявлений с критериями;
* фильтр объявления по категориям.

Клиентская часть разрабатываемого приложения состоит из разработки интерфейса и функциональной части, реализующей возможность покупки, просмотра данных о товаре, его продавце. Для более широкого функционала необходимо зарегистрироваться, с целью ускорения регистрации некоторые поля необязательны для заполнения и могут быть заполнены позже в профиле пользователя.

Для организации в приложении графического пользовательского интерфейса используются элементы управления в виде кнопок, таблиц и форм.

Пользователю предоставляется возможность просматривать, вводить/изменять и выводить данные. При нажатии на активный элемент управления, событие выполняет соответствующее действие.

Для выполнения вышеперечисленных функций необходимо зайти в мобильное приложение. В мобильном приложении запустится главное меню для проведения всех описанных операций.

На основании диаграммы вариантов использования были реализованы следующие функции:

* просмотр подробной информации об объявлении;
* просмотр аккаунта продавца и его активных объявлений;
* вход и регистрация;
* закладки;
* создание объявления.

Для просмотра подробной информации об объявлении пользователю необходимо нажать на объявление. Код функции представлен ниже:

view.setOnClickListener {

val myIntent = Intent(MainActivity.activity, FullAdActivity::class.java)

myIntent.putExtra("adId", ad?.id)

MainActivity.activity.startActivity(myIntent)

}

Для просмотра аккаунта продавца необходимо перейти объявление и нажать на кнопку с фото и именем продавца. Код функции представлен ниже:

sellerLayout = findViewById(R.id.seller\_profile\_layout);

sellerLayout.setOnClickListener(v -> {

Intent myIntent = new Intent(FullAdActivity.this, SellerProfileActivity.class);

myIntent.putExtra("sellerId", ad.getIdUser());

v.getContext().startActivity(myIntent);

});

Для входа или регистрации необходимо перейти в боковое меню и нажать на фото. Код функции представлен ниже:

Call<UserResponse> call = ApiClient.getApi().auth(numberOrEmailStr, inputManager, passwordStr);

call.enqueue(new Callback<UserResponse>() {

@Override

public void onResponse(Call<UserResponse> call, Response<UserResponse> response) {

pDialog.dismiss();

if (response.body().getSuccess()) {

User user = response.body().getUser().get(0);

SharedPreferences.Editor editor = MainActivity.sp.edit();

editor.putString("image", user.getImage());

editor.apply();

Toast.makeText(getContext(), "Добро пожаловать, " + user.getNickname() + "!", Toast.LENGTH\_SHORT).show();

MainActivity.showItemsNavigationDrawer(R.id.MY\_ADS, R.id.FAVOURITES);

} else if (!response.body().getSuccess() && response.body().getCorrectEmail()) {

password.setError("Неверно введен пароль!");

Toast.makeText(getContext(), "Неверно введен пароль!", Toast.LENGTH\_LONG).show();

}

});

Для добавления объявления в закладки необходимо нажать на звезду в объявлении. Код функции представлен ниже:

int user\_id = MainActivity.sp.getInt("id", 0);

formatter.setTimeZone(TimeZone.getTimeZone("GMT+3"));

String datetime = formatter.format(date);

Call<ServerResponse> call = ApiClient.getApi().setDeleteBookmark(ad\_id, user\_id, datetime, flag);

call.enqueue(new Callback<ServerResponse>() {

@Override

public void onResponse(Call<ServerResponse> call, Response<ServerResponse> response) {

if (response.body() != null && response.body().getSuccess() == 1) {

if (flag.equals("set")) {

Toast.makeText(context, "Объявление добавлено в закладки", Toast.LENGTH\_SHORT).show();

bookmark.setChecked(true);

ad.setFavourite(true);

} else if (flag.equals("delete")) {

Toast.makeText(context, "Объявление убрано из закладок", Toast.LENGTH\_SHORT).show();

bookmark.setChecked(false);

ad.setFavourite(false);

}

}

}

});

Для создания нового объявления необходимо быть авторизированным, нажать на плюс в главной ленте и заполнить все поля. Код функции представлен ниже:

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.forLanguageTag("ru"));

dateFormat.setTimeZone(TimeZone.getTimeZone("GMT+3"));

String title = titleEditText.getEditText().getText().toString();

String description = descriptionEditText.getEditText().getText().toString();

for (int i = 0; i < uris.size(); ++i) {

parts.set(i, MultipartBody.Part.createFormData("image" + (i + 1), "-" + (i + 1) + ".png", RequestBody.create(MediaType.parse("image/\*"), files.get(i))));

}

int price = Integer.parseInt(priceEditText.getEditText().getText().toString());

String nickname = MainActivity.sp.getString("nickname", "");

String datetime = dateFormat.format(new Date());

String subcategory = subcategoryFromSpinner;

ApiClient.getApi().insertAd(RequestBody.create(MediaType.parse("multipart/form-data"), title),

RequestBody.create(MediaType.parse("multipart/form-data"), description),

parts.get(0),

parts.get(1),

parts.get(2),

parts.get(3),

parts.get(4),

RequestBody.create(MediaType.parse("multipart/form-data"), String.valueOf(price)),

RequestBody.create(MediaType.parse("multipart/form-data"), nickname),

RequestBody.create(MediaType.parse("multipart/form-data"), datetime),

RequestBody.create(MediaType.parse("multipart/form-data"), subcategory)).enqueue(new Callback<ServerResponse>() {

@Override

public void onResponse(Call<ServerResponse> call, Response<ServerResponse> response) {

pDialog.dismiss();

if (response.body() != null) {

Toast.makeText(getApplicationContext(), "Обьявление создано!", Toast.LENGTH\_LONG).show();

Intent myIntent = new Intent(appCompatActivity, MainActivity.class);

appCompatActivity.startActivity(myIntent);

}

}

@Override

public void onFailure(Call<ServerResponse> call, Throwable t) {

pDialog.dismiss();

Toast.makeText(getApplicationContext(), "Произошла ошибка", Toast.LENGTH\_LONG).show();

}

});

Текст программы представлен в приложении А.

# Описание мобильного приложения

## Общие сведения

Мобильное приложение «Каталог товаров для детей» призвано автоматизировать действия по поиску, просмотру, продаже и покупке товаров для детей, бывших в употреблении. Данное приложение позволяет пользователям продать или купить неиспользуемые товары, мобильное приложение предоставляет удобный и интуитивно-понятный интерфейс.

Установка мобильного приложения происходит путём загрузки APK-файла под названием «Барахолка.apk» на мобильное устройство, далее этот файл необходимо запустить и приложение установится. Запуск мобильного приложения происходит при нажатии на значок приложения, после чего запустится начальная страница приложения.

Приблизительный размер APK-файла составляет 5.5 Мбайт.

Все данные о пользователях, объявлениях находятся в базе данных на сервере, далее эти данные загружаются при переходе на необходимую страницу. Отображение объявлений происходит путем постраничной загрузки их с сервера и расположения в ленте объявлений в необходимом порядке.

Написание мобильного приложения производилось на языках Java, PHP и Kotlin. Приложение создано при помощи следующих библиотек: «Material Design», «Paging Library», «Glide», «Retrofit 2». В качестве главного цвета приложения был выбран красно-оранжевый оттенок, в качестве фона – белый. При переключении на темную тему приложение приобретает темно-серые оттенки.

## Функциональное назначение

Мобильное приложение предназначено для упрощения поиска необходимых вещей по цене ниже новых.

Потенциальная аудитория приложения – люди, старше 18 лет, которые хотят приобрести товары по сниженной цене.

В качестве средств защиты выступает авторизация. Для авторизации пользователь должен иметь аккаунт. Авторизация проходит, путём ввода логина и пароля. Оба параметра указываются при самой регистрации пользователем. Логином является почта или номер телефона пользователя. Пароль может являться любой набор буквенных и численных символов, но длинною не меньше 8 символов. Сам пароль маскируется точками в панели ввода.

## Описание разделов приложения

Раздел «Все объявления» - это начальная страница при входе в приложение. Страница содержит фильтр по категориям, вкладку поиск, кнопку создания нового объявления, ленту с объявлениями и элементы меню.

В разделе «Аккаунт» собрана информация о пользователе.

В разделе «Мои объявления» собраны все объявления пользователя в сгруппированном виде.

В разделе «Закладки» содержатся объявления, добавленные пользователем в закладки.

В разделе «Техническая поддержка» находятся контакты для связи с разработчиком.

В разделе «Настройки» можно изменить тему приложения.

В разделе «Создание объявления» можно создать новое объявление, после чего оно отправится на модерацию.

В разделе «Поиск» можно указать критерии для поиска объявлений.

В разделе «Авторизация» пользователь может войти в свой аккаунт, а если его нет, то перейти в раздел «Регистрация».

В разделе «Регистрация» можно создать новый аккаунт.

В разделе «Полное объявление» отображается полная информация об объявлении, все фотографии и кнопка для перехода на страницу пользователя.

В разделе «Продавец» можно увидеть данные продавца для связи с ним.

На рисунке 4.1 представлен раздел «Все объявления»



Рисунок 4.1 – Раздел «Все объявления»

На рисунке 4.2 представлен раздел «Аккаунт»

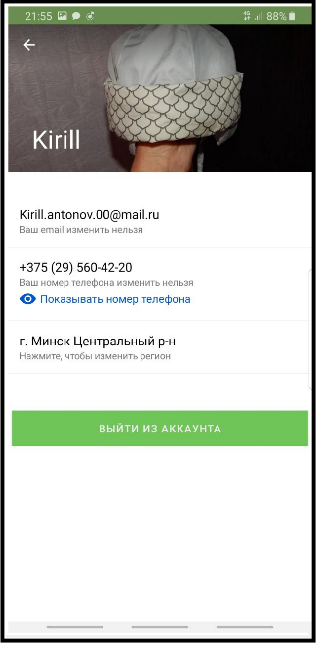


Рисунок 4.2 – Раздел «Аккаунт»

На рисунке 4.3 представлен раздел «Мои объявления»

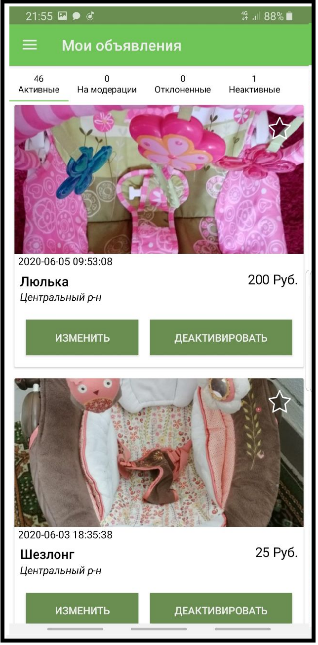


Рисунок 4.3 – Раздел «Мои объявления»

На рисунке 4.4 представлен раздел «Закладки»



Рисунок 4.4 – Раздел «Закладки»

На рисунке 4.5 представлен раздел «Техническая поддержка»



Рисунок 4.5 – Раздел «Техническая поддержка»

На рисунке 4.6 представлен раздел «Настройки»

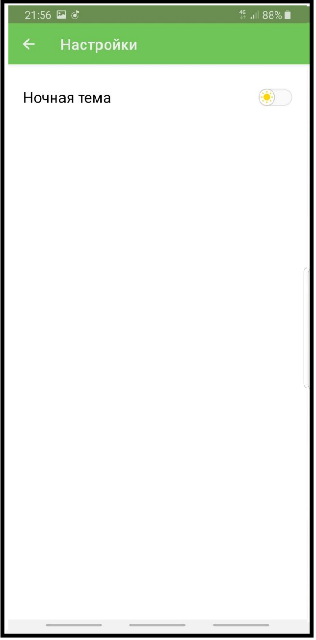


Рисунок 4.6 – Раздел «Настройки»

На рисунке 4.7 представлен раздел «Создание объявления»

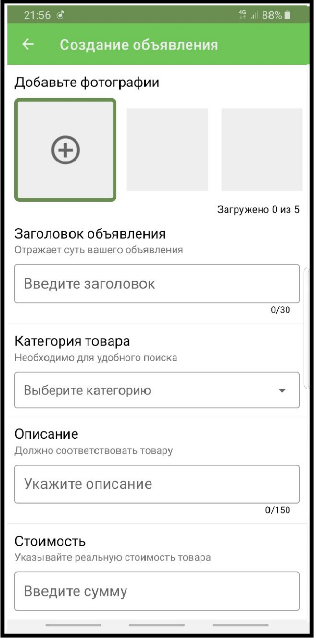


Рисунок 4.7 – Раздел «Создание объявления»

На рисунке 4.8 представлен раздел «Поиск»

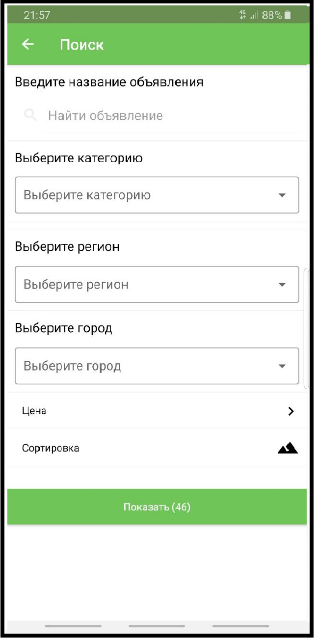


Рисунок 4.8 – Раздел «Поиск»

На рисунке 4.9 представлен раздел «Авторизация»

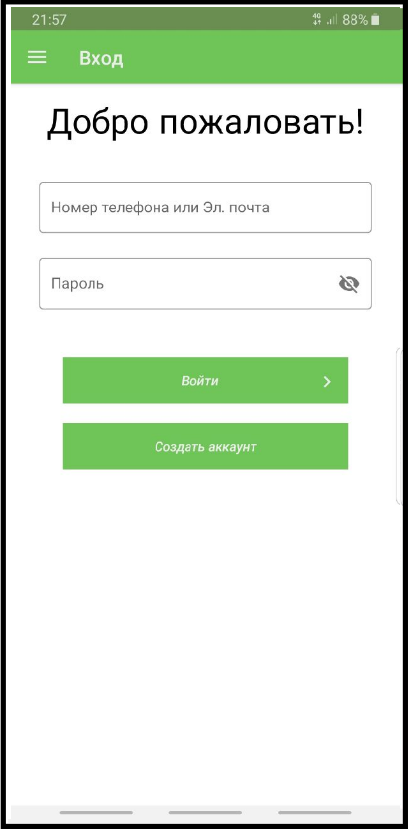


Рисунок 4.9 – Раздел «Авторизация»

На рисунке 4.10 представлен раздел «Регистрация»

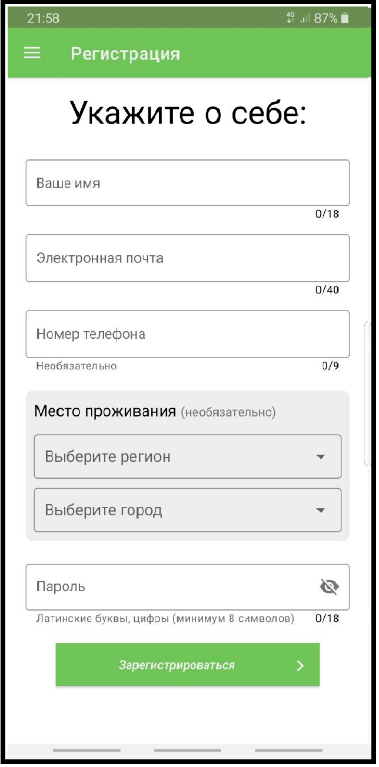


Рисунок 4.10 – Раздел «Регистрация»

На рисунке 4.11 представлен раздел «Полное объявление»

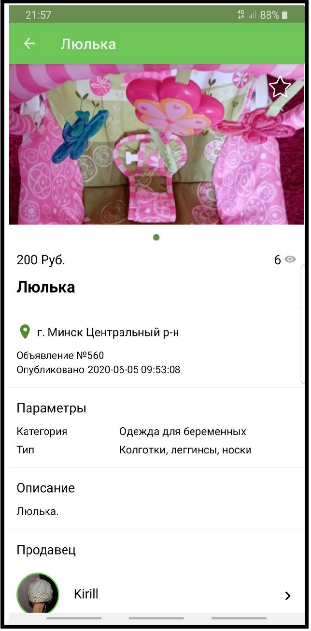


Рисунок 4.11 – Раздел «Полное объявление»

На рисунке 4.12 представлен раздел «Продавец»

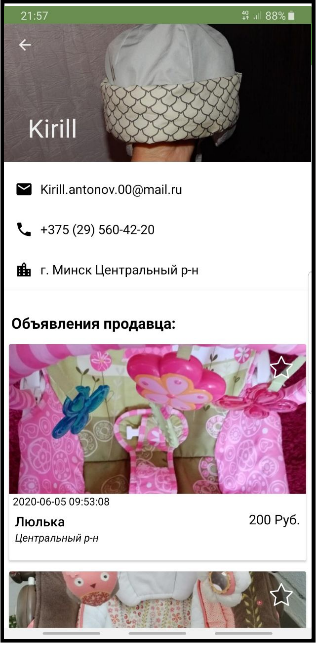


Рисунок 4.12 – Раздел «Продавец»

# Методика испытаний

## Технические требования

Основными минимальными требованиями, выдвигаемыми к аппаратному обеспечению мобильного телефона, являются:

* процессор с частотой не менее 1.2 ГГц;
* оперативная память 1 Гбайт и более;
* внутренняя память не менее 8 Гбайт;
* версия android не ниже 5.0.0.

Мобильный телефон должен быть с установленной ОС Android версии не ниже 5.0.0, так как разработка мобильного приложения не подразумевает версии ниже из-за множества урезанных функций.

## Функциональное тестирование

В процессе написания программного средства необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тестирование программы будет производиться последовательно, переходя из одной части программы в другую. Во время теста будут проверяться все действия с программой, навигация пунктам меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены.

При запуске мобильного приложения будет открываться главная страница, представлена на рисунке 4.1.

В таблице 5.1 представлен тест-кейс для тестирования функции регистрации нового пользователя.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип теста | Описание | Шаги воспроизведения | Результат |
| 001 | Позитивный | Регистрация незарегистриро-ванного пользователя, ввод корректных данных | 1.Запустить приложение;  2. На панели навигации выбрать: «Войти/Регистрация» (рисунок 5.1);  3.Нажать: «Зарегистрироваться» (рисунок 4.9);  4. Заполнить поля:  Имя – Иванов,  Почта – Ivanov@mail.ru,  Пароль – Qwe12345678;  5. Нажать кнопку: «Зарегистрироваться». | Ожидаемый результат: сообщение об успешном входе. |
| Фактический результат:  результат соответствует ожидаемому. Все условия соблюдены. Результат неправильного ввода данных в форме регистрации, согласно рисунка 5.2. |

Таблица 5.1 – Test-case для регистрации нового пользователя

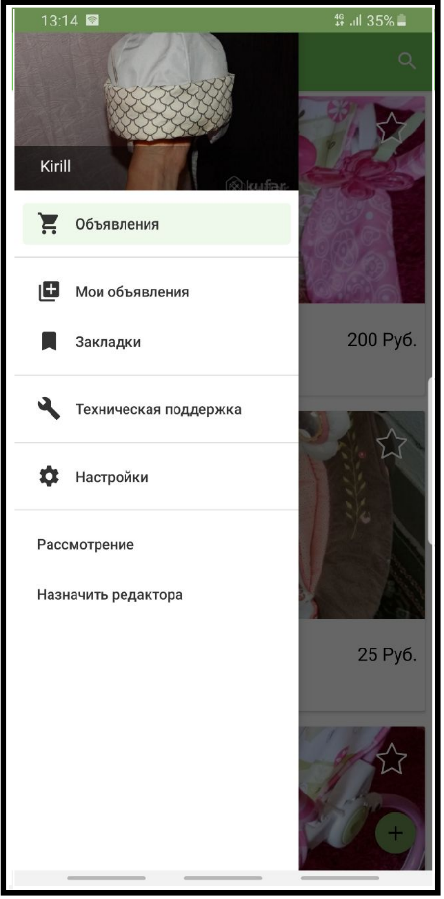


Рисунок 5.1 – Панель навигации

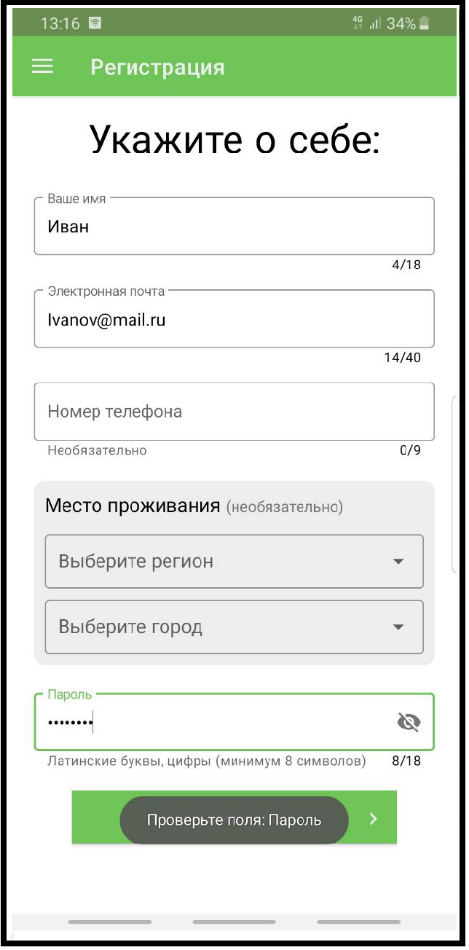


Рисунок 5.2 – Результат неправильного ввода данных в форме регистрации

В таблице 5.2 представлен тест-кейс для авторизации уже существующего пользователя.

Таблица 5.2 – Test-case для авторизации существующего пользователя

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип теста | Описание | Шаги воспроизведения | Результат |
| 002 | Позитивный | Авторизация существующего пользователя, ввод корректных данных | 1. Запустить приложение;  2. На панели навигации выбрать «Войти/Регистрация»;  3. Заполнить поля:  Почта или телефон – Ivanov@mail.ru,  Пароль – Qwe12345678, (рисунок 4.9);  4. Нажать кнопку: «Войти». | Ожидаемый результат: предупреждение об успешном входе. |
| Фактический результат:  результат соответствует ожидаемому. Все условия соблюдены. Результат авторизации, согласно рисунка 5.3. |

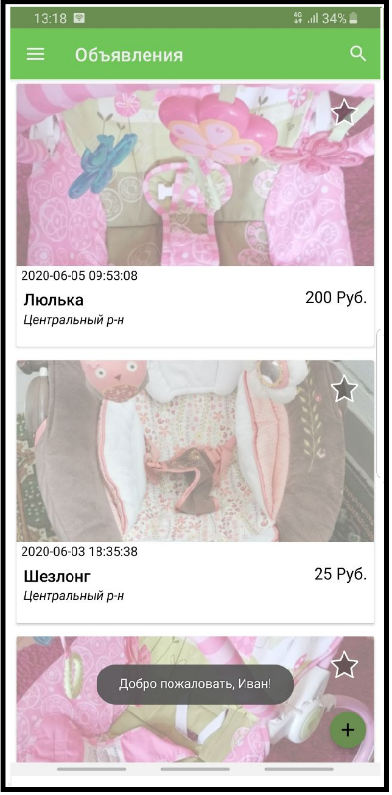


Рисунок 5.3 – Результат авторизации

В таблице 5.3 представлен тест-кейс для добавления аватара на аккаунт пользователя с камеры устройства.

Таблица 5.3 – Test-case для добавления аватара с камеры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип теста | Описание | Шаги воспроизведения | Результат |
| 003 | Позитивный | Установка аватара пользователя с камеры | 1. Запустить приложение;  2. На панели навигации выбрать: «Иван»;  3. Нажать на фото по умолчанию;  4. Нажать кнопку: «Сделать снимок»  (рисунок 5.4);  5. Нажать кнопу выбрать сделанный снимок. | Ожидаемый результат:  Выбранный снимок загрузился на сервер и отобразился в приложении. |
| Фактический результат: результат соответствует ожидаемому. Все условия соблюдены. Результат установленного аватара, согласно рисунка 5.5. |



Рисунок 5.4 – Снимок с камеры

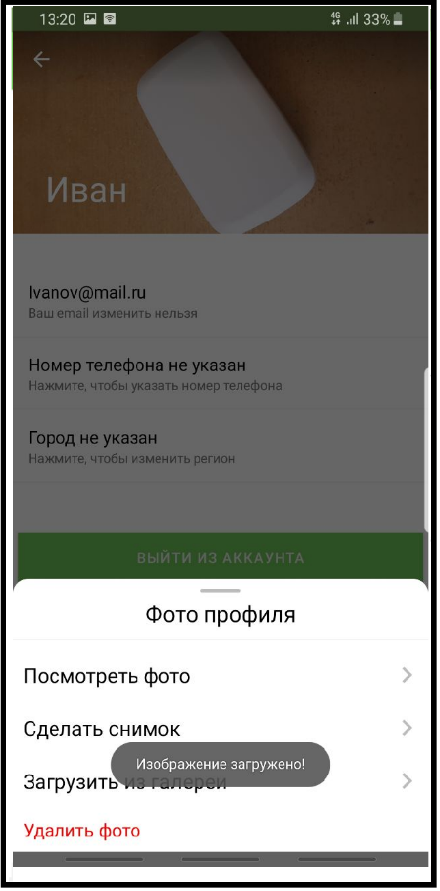


Рисунок 5.5 – Результат установки аватара

В таблице 5.4 представлен тест-кейс для добавления аватара из галереи.

Таблица 5.4 – Test-case для добавления аватара из галереи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип теста | Описание | Шаги воспроизведения | Результат |
| 004 | Позитивный | Установка аватара пользователя с галереи | 1. Запустить приложение;  2. На панели навигации выбрать: «Иван»;  3. Нажать на изображение;  4. Нажать на кнопку «Загрузить из галереи», (рисунок 5.6);  5. Выбрать необходимое фото. | Ожидаемый результат:  Загрузка фото на сервер, отображение фото в приложении. |
| Фактический результат: результат соответствует ожидаемому. Все условия соблюдены. Результат установки аватара, согласно рисунка 5.7. |

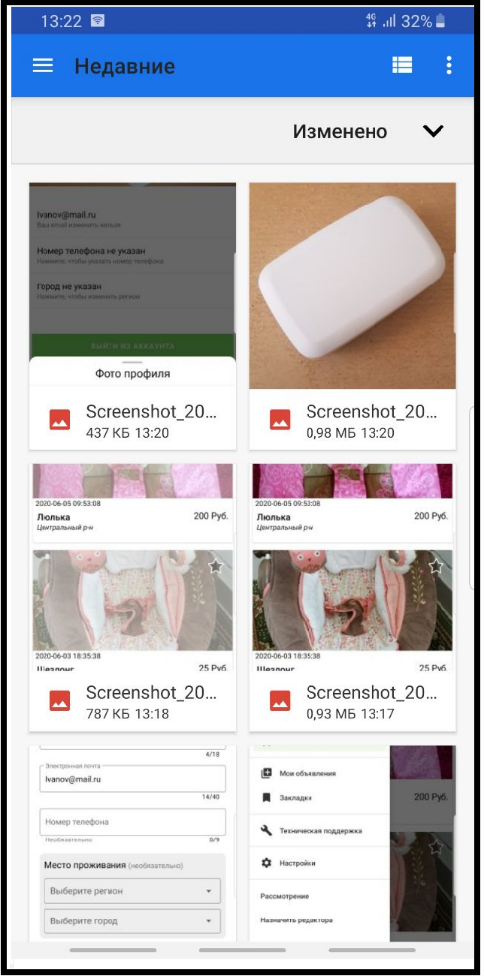


Рисунок 5.6 – Выбор фото из галереи

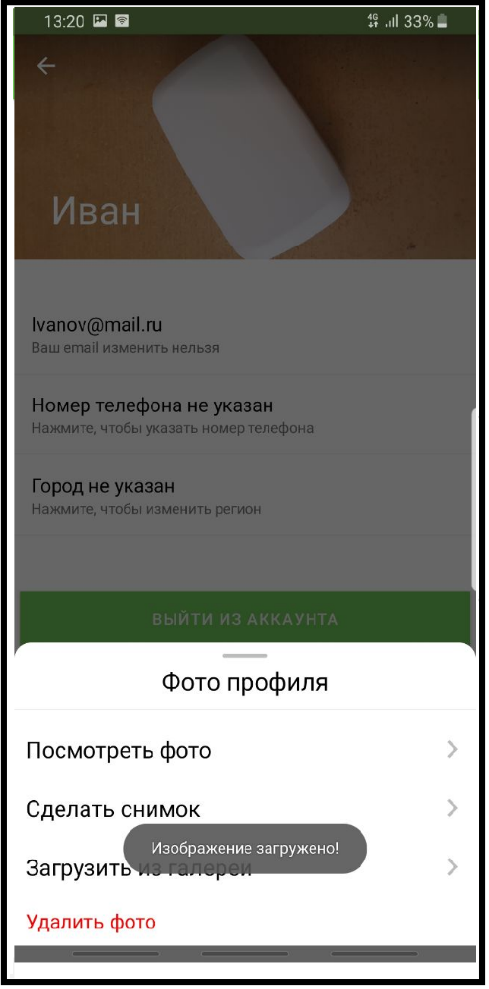


Рисунок 5.7 – Результат установки аватара

В таблице 5.5 представлен тест-кейс для добавления аватара, превышающего допустимый размер.

Таблица 5.5 – Test-case для добавления аватара, превышающего допустимый размер

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип теста | Описание | Шаги воспроизведения | Результат |
| 005 | Позитивный | Добавление на аватар фото, превышающего максимально допустимый размер | 1.Запустить приложение;  2.На панели навигации выбрать: «Иван»;  3. Нажать на фото;  4. Нажать кнопку: «Загрузить из галереи»;  5. Выбрать фото, размер которого больше 5 мб. | Ожидаемый результат:  Предупреждение о загрузке фото, превышающего максимально допустимый размер. Фото не загружено на сервер и не отображено в приложении. |
| Фактический результат: результат соответствует ожидаемому. Все условия соблюдены. Результат добавления фото, превышающего допустимый размер, согласно рисунка 5.8. |

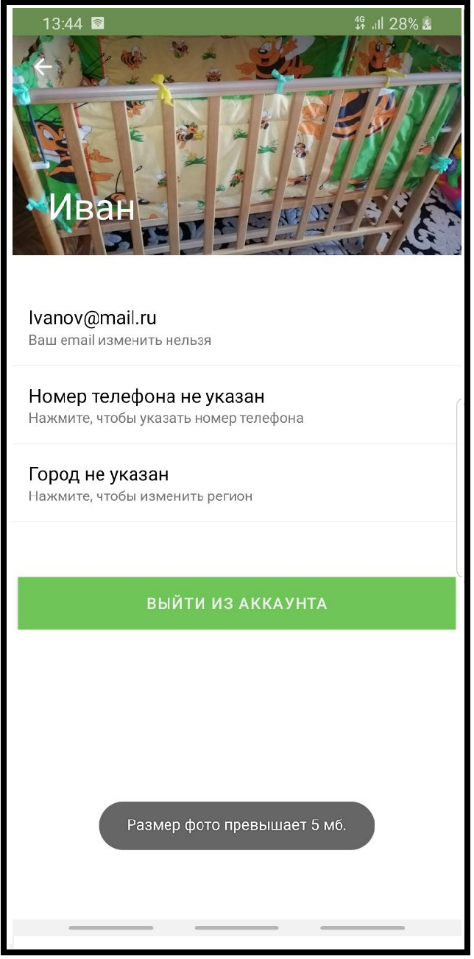


Рисунок 5.8 – Загрузка некорректного аватара

В таблице 5.6 представлен тест-кейс для создания нового объявления пользователем.

Таблица 5.6 – Test-case для создания объявления пользователем

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип теста | Описание | Шаги воспроизведения | Результат |
| 006 | Позитивный | Создание нового объявления | 1. Запустить приложение;  2. На главном фрагменте нажать на кнопку создания нового объявления;  3. Заполнить поля:  Добавить хотя бы 1 фото не превышающее 3 мб, заголовок – «Кроватка»,  описание – «Кроватка в хорошем состоянии», стоимость – «50 руб»;  5. Нажать на кнопку «Создать объявление». | Ожидаемый результат:  Загрузка данных на сервер, отправление объявления на модерацию, предупреждение о созданном объявлении. |
| Фактический результат: результат соответствует ожидаемому. Все условия соблюдены. Результат создания нового объявления, согласно рисунка 5.9. |

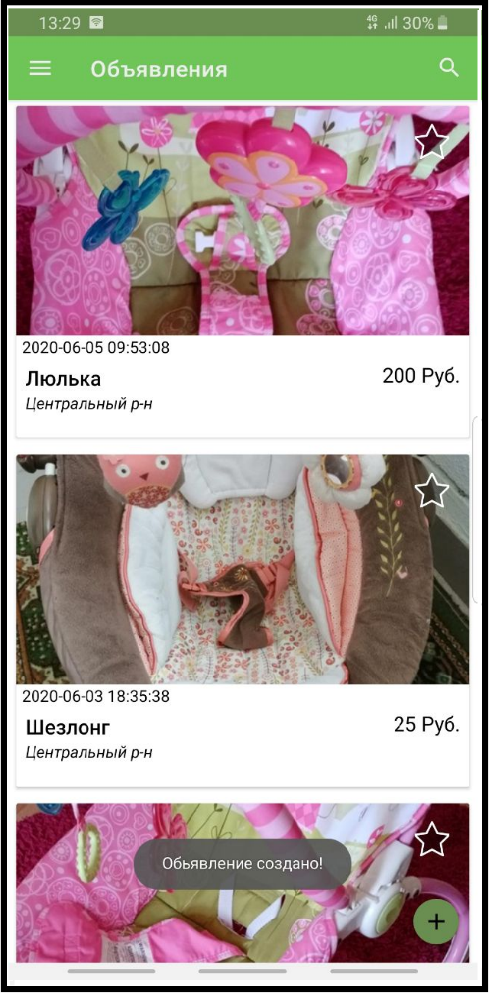


Рисунок 5.9 – Отправление нового объявления на модерацию

# Применение

## Назначение программы

Мобильное приложение предназначено для упрощения процесса покупки и продажи товаров для мам и детей бывшего употребления. С помощью этого приложения пользователь сможет найти необходимый товар без каких-либо проблем, так как в приложении реализован поиск.

С целью улучшения качества контента в приложении реализован отбор объявлений. Эту роль выполняет администратор, его задачей является одобрение или же отклонение объявлений после их создания. При отклонении администратор обязан указать причину отклонения, а также по желанию может написать комментарий к отклонению.

При возникновении каких-либо проблем пользователь может перейти во вкладку «Техническая поддержка» в меню приложения и перейдя по ссылке описать свою проблему.

Разработка мобильного приложения ведется только под систему Android ввиду того, что для разработки такого же приложения для операционной системы IOS необходимо использовать абсолютно другой язык программирования и придерживаться других принципов разработки.

# Охрана труда и окружающей среды

## Правовые, нормативные, социально-экономические и организационные вопросы охраны труда

С 1 ноября 2005 г. В республике введены в действие стандарты СТБ 18001-2005 и СТБ 18002-2005 по СУОТ, требования которых гармонизированы с международными и национальными стандартами. Внедрение этих документов позволит оценивать и управлять рисками и тем самым повышать эффективность работы предприятия.

В СУОТ, как и в любой другой системе управления, определяются основные функции и задачи, структура информационных и управленческих связей, формы учетных и отчетных документов и тд.

Функционирование СУОТ обеспечена непосредственно директором организации. Обязанности по ОТ грамотно распределены между работниками. Проводится периодичное усовершенствований работы системы охраны труда.

Управление охраной труда в ООО «Сервис-парнёр» осуществляется первым заместителем директора, в структурных подразделениях - руководители структурных подразделений.

Полномочия и ответственность руководителей структурных подразделений организации по осуществлению контроля за соблюдением законодательства об охране труда определяются положениями об отделе.

В организации «Сервис-парнёр» ответственный за охрану труда занимает должность первого заместителя директора. Раз в три года он должен проходить проверку знаний в комиссии по вопросам охраны труда Минского городского исполнительного комитета.

Функции ответственного за охрану труда в организации:

* выявление опасных и вредных производственных факторов на рабочих местах;
* проведение анализа состояния условий и охраны труда, причин нарушения законодательства о труде и охране труда, производственного травматизма, профессиональной и профессионально обусловленной заболеваемости;
* оказание помощи подразделениям предприятия в организации и проведении замеров параметров опасных и вредных производственных факторов, аттестации и сертификации рабочих мест и производственного оборудования на соответствие требованиям охраны труда;
* информирование работающих от лица работодателя о состоянии условий труда на рабочих местах, о причинах возникновения профессиональных заболеваний и мероприятиях по их предупреждению, о принятых мерах по защите от опасных и вредных производственных факторов;
* участие в подготовке документов на выплату возмещения вреда, причиненного здоровью работающих в результате несчастного случая на производстве или профессионального заболевания;
* проведение проверок, обследований (или участие в проверках, обследованиях) технического состояния зданий, сооружений, оборудования, машин и механизмов на соответствие их нормативным правовым актам по охране труда, эффективности работы вентиляционных систем, состояния санитарно-технических устройств, санитарно-бытовых помещений, средств коллективной и индивидуальной защиты;
* разработка совместно с руководителями подразделений и другими службами предприятия мероприятий по предупреждению несчастных случаев на производстве и профессиональных заболеваний, по улучшению условий труда и доведению их до требований нормативных правовых актов по охране труда, а также оказание организационной помощи по выполнению запланированных мероприятий;
* участие в составлении раздела «Охрана труда» коллективного договора, соглашения по охране труда предприятия;
* участие в работе комиссии по приемке в эксплуатацию законченных строительством или реконструированных объектов производственного назначения, по приемке из ремонта установок, агрегатов, станков и другого оборудования;
* составление (при участии руководителей подразделений и соответствующих служб предприятия) перечней профессий и видов работ, на которые должны быть разработаны инструкции по охране труда;
* оказание методической помощи руководителям подразделений предприятия при разработке и пересмотре инструкций по охране труда для работников, стандартов предприятия по безопасности труда;
* разработка программы и проведение вводного инструктажа по охране труда со всеми вновь принимаемыми на работу, командированными, учащимися и студентами, прибывшими на производственную практику или обучение;
* участие в работе комиссий по проверке знаний по охране труда у работников предприятия;
* составление отчетности по охране труда по установленным формам и в соответствующие сроки и др.

Порядок подготовки, переподготовки, стажировки, инструктажа, повышения квалификации и проверки знаний, работающих по вопросам охраны труда установлен инструкцией о порядке подготовки, переподготовки, стажировки, инструктажа, повышения квалификации и проверки знаний, работающих по вопросам охраны труда, утвержденной постановлением Министерства труда и социальной защиты Республики Беларусь 28.11.2008 № 175.

По характеру и времени проведения инструктажи по охране труда подразделяются на:

* вводный проводят: со всеми работниками, которые впервые принимаются на постоянную или временную работу, независимо от их образования, трудового стажа, стажа работы по этой профессии, специальности, должности, а также с командированными при участии их в производственном процессе или выполнении или выполнении работ на территории организации, учащимися и студентами, прибывшими на производственное обучение или практику. Проводит инженер по охране труда либо специалист, на которого возложены эти обязанности, а с учениками в учебном заведении – преподаватель либо мастер производственного обучения;
* первичный на рабочем месте проводится до начала производственной деятельности с работниками, принятыми на работу. Проводится с каждым работником индивидуально, с практическим показом безопасных приемов и методов работы;
* повторный проходят все работники независимо от квалификации, образования, стажа и характера выполняемой работы не реже одного раза в полугодие. Он проводится либо с группой работников, обслуживающих однотипное оборудование и в пределах общего рабочего места по программе первичного инструктажа на рабочем месте, либо в объеме инструкций по охране труда на рабочем месте;
* внеплановый проводится при введении в действие новых либо переработанных нормативных актов по охране труда или внесении изменений в них, а также при изменении технологического процесса, замене или модернизации оборудования, приборов и инструментов, сырья, материалов и иных факторов, воздействующих на охрану труда. Проводится индивидуально либо с группой работников одной профессии;
* целевой проводится при ликвидации последствий аварий, стихийных бедствий и катастроф, а также при производстве работ, на которые оформляется наряд-допуск или разрешение. Фиксируется в наряде-допуске, разрешении и ином документе.

Порядок расследования несчастных случаев на производстве определен Правилами расследования несчастных случаев на производстве и профессиональных заболеваний, утвержденными постановлением Совета Министров Республики Беларусь от 15 января 2004 г. № 30.

Целями расследования несчастных случаев являются:

* установление причин, вызвавших несчастный случай;
* установление виновного;
* разработка плана мероприятий, направленных на предотвращение подобных случаев.

Основными видами контроля являются:

* контроль за соблюдением законодательства об охране труда, осуществляемый руководителями и специалистами организации в соответствии с их должностными обязанностями;
* общественный контроль за соблюдением законодательства об охране труда, осуществляемый профсоюзами;
* периодический контроль за соблюдением законодательства об охране труда, осуществляемый представителями нанимателя с участием общественных инспекторов профсоюзов по охране труда (уполномоченных лиц по охране труда работников);

Периодический контроль функционирования СУОТ проводится нескольких видов, а именно:

* ежедневный – осуществляется в структурных подразделениях их руководителями с участием общественного инспектора профсоюза по охране труда;
* ежемесячный – осуществляется в отделах главой отдела с участием общественного инспектора профсоюза по охране труда, руководителей отдела и ответственных за охрану труда;
* ежеквартальный – осуществляется в организации в целом руководителем организации с участием руководителей служб общественного инспектора профсоюза по охране труда;

## Организация труда и отдыха подростков

Подростково-юношеский возраст (14... 18 лет) характеризуется рядом анатомо-физиологических особенностей. Организм подростков сильнее реагирует на действие вредных веществ, пониженных и по­вышенных температур воздуха (в связи с менее совершенной системой терморегуляции), на шум, высокую физическую нагрузку, одинако­вую со взрослыми работу подростки выполняют ценой больших энер­гетических затрат, мышечная выносливость у них на 20...30% ниже.

Для работающей молодежи законодательство предусматривает ряд льгот и ограничений.

На постоянную работу разрешено принимать лиц не моложе 16 лет, в исключительных случаях по согласию одного из родителей с 14 лет (ст. 272 ТК). Школьников, учащихся профтехучилищ, средних специальных учебных заведений, достигших 14-летнего возраста, мож­но по их желанию и с согласия одного из родителей принимать на легкую работу как в период каникул, так и в течение всего учебного года в свободное от занятий время.

Все лица моложе 18 лет принимаются на работу лишь после предварительного медицинского осмотра и в дальнейшем, до достиже­ния 18 лет, ежегодно подлежат обязательному медицинскому осмотру. Им не устанавливается испытание при приеме на работу, с ними не заключаются договоры о полной материальной ответственности, еже­годный отпуск предоставляется до истечения шести месяцев непрерыв­ной работы в летнее время или, по их желанию, в любое другое время года продолжительностью не менее одного календарного месяца.

Запрещено применение труда несовершеннолетних на тяжелых работах и на работах с вредными и опасными условиями труда (ст. 274 ТК), также привлекать их к ночным и сверхурочным работам, к работам в государственные праздники, выходные и праздничные дни.

Кроме того, в РБ предусмотрены нормы подъема и перемещения тяжести вручную подростками от 14 до 18 лет, утвержденные поста­новлением Министерства труда РБ от 18.12.1997 г. №116, и являются обязательными для применения при разработке проектной докумен­тации вновь строящихся и реконструируемых объектов (табл. 2.1).

На отдельных видах работ запрещается применение труда лиц до 18 лет по всем профессиям рабочих, например, связанных с обслу­живанием технологического оборудования при производстве синтети­ческого каучука и продуктов нефтехимии, асбеетных изделий и др.

Выпускникам государственных высших, средних специальных и профессионально-технических учебных заведений, а также военно­служащим срочной службы, уволенным из Вооруженных Сил РБ, га­рантируется предоставление первого рабочего места.

У подростков моложе 18 лет организм отличается повышенной чувствительностью к неблагоприятным производственным факторам. Трудовое законодательство предусматривает льготные условия и ог­раничения, направленные на улучшение условий и охраны труда мо­лодежи (ст. 272—282 ТК РБ).

Трудовое законодательство предусматривает льготные условия и ограничения, направленные на улучшение условий и охраны труда молодежи. Не допускается прием на работу лиц моложе 16 лет. В ис­ключительных случаях молодежь в возрасте от 15 до 16 лет прини­мают только по согласованию с профсоюзом, как правило, только для производственного обучения. Лица моложе 18 лет принимаются на работу после предварительного медицинского осмотра и в дальней­шем до достижения 18-летнего возраста обязательно проходят профи­лактические медицинские осмотры.

Продолжительность рабочей недели для лиц в возрасте 16...18 лет установлена 36 ч, для лиц от 15 до 16 лет — 24 ч. Их запрещено привлекать к ночным и сверхурочным работам, а также к работам в выходные дни. Этим лицам предоставляется ежегодный отпуск в 1 календарный месяц предпочтительно летом или в любое время года по их выбору.

Увольнение по инициативе администрации допускается лишь в исключительных случаях с согласия профсоюза и комиссии по де­лам несовершеннолетних при исполкоме, при этом предлагается обя­зательное трудоустройство увольняемого на другом предприятии.

Существует ограничение на применение труда подростков в ра­ботах по переноске тяжестей (масса груза не должна превышать 16,4 кг).

Контроль за выполнением руководителями предприятий меро­приятий по охране труда подростков осуществляет врач по гигиене.

Нормы предельно допустимых величин подъема и перемещения тяжестей вручную подростками от 14 до 18 лет

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Подростки,  возраст | Подъем и перемещение груза вручную в течение смены, кг | | Суммарная масса груза, подымаемого и перемещае­мого в течение смены, кг | |
| постоянно, более 2 раз в час | при чередовании с другой работой, до 2 раз в час | при подъеме с рабочей поверхности | при  подъеме с пола |
| Подростки женского пола | | | | |
| 14 | 3 | 4 | 180 | 90 |
| 15 | 4 | 5 | 200 | 100 |
| 16 | 5 | 7 | 400 | 200 |
| 17 | 6 | 8 | 500 | 250 |
| Подростки мужского пола | | | | |
| 14 | 6 | 10 | 400 | 200 |
| 15 | 7 | 12 | 500 | 250 |
| 16 | 10 | 16 | 900 | 450 |
| 17 | 12 | 18 | 1400 | 700 |

Порядок и условия предоставления первого рабочего места ука­занным лицам определен Положением, утвержденным постановле­нием СМ РБ от 27.03.1998 г. №487 (с изм. и доп., внесенными поста­новлением СМ РБ от 28.04.2000 г. №597).

Аспиранты, завершившие обучение в очной аспирантуре и на­правленные по договорам (заявкам) к нанимателям, обеспечиваются работой в соответствии с заключенными договорами (заявками).

Нормы выработки для работников моложе восемнадцати лет ус­танавливаются исходя из норм выработки для взрослых работников пропорционально сокращенной продолжительности рабочего времени, предусмотренной законодательством для данной категории работни­ков: в возрасте от 16 до 18 лет— не более 36 ч в неделю, от 14 до 16 — 24 ч в неделю.

Заработная плата несовершеннолетним, несмотря на сокращен­ную продолжительность рабочего времени, выплачивается в таком же размере, как работникам соответствующих категорий при полной продолжительности ежедневной работы. Труд работников моложе во­семнадцати лет, допущенных к сдельным работам, оплачивается по сдельным расценкам, установленным для взрослых работников. За время, на которое продолжительность их ежедневной работы сокра­щается по сравнению с продолжительностью ежедневной работы взрос­лых работников, предусмотрена доплата по тарифной ставке.

Дополнительные гарантии предусмотрены для работников мо­ложе 18 лет при расторжении трудового договора по инициативе на­нимателя. Расторжение трудового договора возможно при ликвида­ции предприятия, несоответствии работника занимаемой должности, вследствие недостаточной квалификации или состояния здоровья, пре­пятствующего выполнению данной работы (пп. 1...3 и 6 ст. 42 ТК).

Нарушение трудовой дисциплины по пп. 4, 5, 7 и 9 ст. 42 ТК (неисполнение обязанностей, предусмотренных трудовым договором, прогулы, появление на работе в состоянии алкогольного, токсического или наркотического опьянения) также дают право нанимателю рас­торгнуть трудовой договор после предварительного, не менее чем за две недели, уведомления районной (городской) комиссии по делам несовершеннолетних.

## Пожарная безопасность

В ООО «Сервис-парнёр» согласно ТКП 474-2013, помещение в котором будет создаваться, использоваться программный продукт, входит в категорию помещений В1-В4, пожароопасные, помещении есть горючие вещества и материалы, способные при взаимодействии с кислородом или воздуха гореть.

Согласно ПУЭ, помещение классифицируется на зоны. Помещение, в котором создается дипломный проект относится к зоне П-IIa, т.к зона, расположена в помещении, в котором обращаются твёрдые горючие вещества.

При пожаре строительные конструкции имеют следующие предел огнестойкости и класс пожарной опасности:

* несущие элементы здания (R 45-K1) – предел огнестойкости 45 минут - по потере несущей способности, мало пожароопасные;
* самонесущие стены (RE 30-K1) – предел огнестойкости 30 минут - по потере несущей способности и потере целостности независимо от того, какое из двух предельных состояний наступит ранее, мало пожароопасные;
* наружные несущие стены (E 15-K2) – предел огнестойкости 15 минут - по потере целостности, умеренно пожароопасные;
* перекрытия междуэтажные (REI 45-K1) – предел огнестойкости 45 минут - по потере несущей способности, целостности и теплоизолирующей способности независимо от того, какое из трех предельных состояний наступит ранее, мало пожароопасные;
* элементы покрытий без чердаков – настилы, в том числе с утеплителем (RE 15-K1) - предел огнестойкости 15 минут – по потере несущей способности и потере целостности независимо от того, какое из двух предельных состояний наступит ранее, мало пожароопасные;
* элементы покрытий без чердаков – фермы, балки, прогоны (R 15-K1) - предел огнестойкости 15 минут - по потере несущей способности, мало пожароопасные;
* лестничные клетки - внутренние стены (REI 60-KO) - предел огнестойкости 60 минут - по потере несущей способности, целостности и теплоизолирующей способности независимо от того, какое из тех предельных состояний наступит ранее, не пожароопасные;
* лестничные клетки - марши и площадки лестниц (R 45-KO) - предел огнестойкости 45 минут – по потере несущей способности, не пожароопасные.

В качестве первичных средств пожаротушений, в организации используются порошковые, углекислотные огнетушители марки ОП-5 и ОУ-10 соответственно, так как они наибольшим образом подходят для тушения электронной техники и близлежащих объектов, также имеются пожарные краны. Огнетушители размещаются на видных и легкодоступных местах для сотрудников, для быстрой ликвидации в случаи возгорания, в каждом помещении с сотрудниками, не препятствуют безопасной эвакуации людей. Расстояние от возможного очага пожара, до места размещения первичного средства пожаротушения должно быть не более 20 метров. Пожарные краны, марки Ду50 размещены на коридоре в специально оборудованных полках с закрывающейся дверью, механизмом, снабжены пломбой.

В организации отсутствуют средства автоматического пожаротушения. В качестве средств и устройств об извещении о пожаре используются дымоуловители и пожарные извещатели. Дымоуловители равномерно распложены по помещениям и располагаются на потолке. Пожарные извещатели расположены в помещениях на стенках в легко доступных местах.

Добровольная пожарная дружина, как это обязательно установлено на каждом предприятии, входит в обязательный численный состав организации. Численный состав дружины составляет 10 человек, поскольку численность самой организации не превышает 100 человек. Командиры, старшие расчетов, назначены руководителем организации. В помещениях на видном месте вывешен табель действия при пожарах. Все члены дружины своевременно проходят контроль знаний.

Основными задачами добровольной пожарной дружины являются:

* контроль за соблюдением противопожарного режима;
* проведение разъяснительной работы среди работников по соблюдению противопожарного режима на рабочих местах и правил осторожного обращения с огнем в быту;
* надзор за исправностью средств пожаротушения и их укомплектованностью;
* вызов пожарной службы в случае возникновения пожара, принятие мер по его тушению имеющимися средствами и т.п.

Члены добровольной пожарной дружины принимают участие в локализации и ликвидации загораний, эвакуации людей и материальных ценностей из горящих помещений.

При эвакуации здания, сотрудники покидают офисные помещения, направляются к лестнице, по которой осуществляют спуск, затем выходят из здания. Пути эвакуации отражены на плане эвакуации. Сам план содержит схему занимаемых помещений организацией, с обозначениями. Каждый сотрудник обязательно ознакамливается с планом эвакуации.

## Охрана окружающей среды

В ООО «Сервис-парнёр» согласно санитарным нормам в помещении должен быть доступ к свежему воздуху. Доступ к свежему воздуху осуществляется через форточное проветривание или же кондиционирование, ионизацию воздуха.

В основном используется форточное проветривание. Оно закреплено графиком, согласно которому минимум два раза в день помещение должно проветриваться, при этом сотрудники не должны присутствовать в помещении. Кондиционер используется для поддержания благоприятной температуры в помещении.

План очистки воздуха:

- оповестить сотрудников в занимаемом помещении;

- открыть форточку в режим проветривания;

- освободить помещение;

- закрыть помещение на время проветривания.

Используемая вода в организации, сточная вода, направляется в канализацию, для дальнейшей приемки водоочистительными сооружениями. Стоит заметить, что вода, должна приниматься согласно условиям местных исполнительных органов, по согласованию с органами Министерства природных ресурсов и охраны окружающей среды Республики Беларусь. Данные требования также в самом общем виде определяются в Правилах пользования централизованными системами водоснабжения, водоотведения (канализации) в населенных пунктах. Способы очистки сточных вод:

* механический – размельчение осадков и отложений различными инструментами и механизмами;
* гидромеханический – подача струи воды под давлением для размыва и выноса отложений и грязи;
* физико-химический – циркуляция специально подобранных растворителей в аппаратуре без ее вскрытия;
* выжигание отложений в струе воздуха без вскрытия аппаратов и др.

Вывозом и утилизацией бытового мусора, занимается сторонняя организация «Remondis», согласно нормам, в существующем законодательстве. Сама утилизация производится термически. Термическая утилизация, подразумевает собой переработку отходов, путем сжигания. Способы утилизации бытового мусора:

* захоронение – производственный мусор, который не может быть использован вторично, подвергается хранению на специальных полигонах и свалках. Сбор, переработка и складирование радиоактивных отходов — большая проблема для любого государства. Их хранят в могильниках, которые должны быть расположены как можно дальше от грунтовых вод, в сейсмически безопасных зонах;
* компостирование – естественное разложение тяжёлых бытовых отходов. В результате процесса получают компост, который эффективно используется в сельском хозяйстве. В Беларуси способ утилизации не получил широкого распространения;
* термическая переработка – превращение мусора в топливо, путём его сжигания;
* плазменная переработка – процесс представляет собой разложение ТБО без доступа воздуха. Мусор нагревается и расщепляется на вещества с более низкой молекулярной массой. Пиролиз имеет больше преимуществ, чем сжигание: не загрязняется окружающая среда, тяжелые металлы превращаются в золу, в остатках отходов не содержится токсичных веществ.

В организации используются лампы, как люминесцентные, так и ртутные. По истечению срока использования ламп прибытию их в негодность, нужна их утилизация. Утилизация ламп осуществляется путем оформления заявки в организации «БелВторСырье», затем вывоз и утилизация.

# Экономический раздел

## Технико-экономическое обоснование разработки программного средства

Необходимо рассчитать экономическую эффективность разработки программного средства.

В технико-экономическом обосновании будут рассмотрены следующие вопросы:

* составление плана по разработке программного средства;
* расчет стоимости разработки;
* экономическая эффективность разработки.

В первой части технико-экономического обоснования необходимо рассчитать срок разработки по созданию программного средства для управления задачами в ООО «Сервис-парнёр»

Во второй части технико-экономического обоснования необходимо рассчитать стоимость разработки программного средства с момента получения первого варианта технического задания и заканчивая оформлением документации и сдачей разработки.

В третьей части необходимо оценить экономическую эффективность программного средства.

## Составление плана по разработке программного средства

План разработки программного средства представлен в таблице 8.1.

Таблица 8.1 – План разработки программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов и видов работ | Исполнитель | Количество исполнителей | Трудоемкость, человеко-дни |
| Подготовительный | техник-программист | 1 | 3 |
| Выбор методов и средств | техник-программист | 1 | 6 |
| Разработка алгоритмов и программ | техник-программист | 1 | 10 |
| Отладка программ и анализ результатов | техник-программист | 1 | 17 |
| Оформление документации и подготовка к сдаче разработки | техник-программист | 1 | 4 |

## Определение цены программного средства

В условиях рыночных отношений научно-техническая продукция также является товаром. Поэтому узловым вопросом технико-экономического обоснования выступает определение цены основного результата дипломного проекта.

Вместе с тем следует иметь в виду, что в отраслевых рекомендациях по установлению цены на научно-техническую продукцию фактически реализован вариант механической аналогии с материальным производством (издержек или сметной стоимости) и учет минимального уровня рентабельности.

Расчет цены основного результата дипломного проекта осуществляется в определенной последовательности.

Определяются материальные затраты на выполнение работ по теме, включая стоимость покупных комплектующих изделий и полуфабрикатов на изготовление макетов и опытных образцов. Расчет осуществляется по формуле (8.1)

, (8.1)

где – коэффициент, учитывающий транспортно-заготовительные расходы, ≈от 1,05 до 1,10;

– норма расхода i-го вида материалов на макет или опытный образец (кг, м, и т.д.);

– действующая отпускная цена за единицу i-го вида материала, руб.;

– возвратные отходы i-го вида материала (кг, м, и т.д.);

– цена за единицу возвращенных отходов i-го вида материала, руб.;

– количество применяемых видов материалов.

Расчет целесообразно представить в таблице 8.2.

Таблица 8.2 – Расчет затрат на материалы, покупные полуфабрикаты и комплектующие изделия

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование материалов покупных полуфабрикатов и комплектующих изделий | Единица измерения | Количество | Цена приобретения без НДС, руб. | НДС, руб. | Цена с НДС, руб. |
| Бумага | пачка | 1 | 7,22 | 1,44 | 8,66 |
| Ручка шариковая | штука | 1 | 0,73 | 0,15 | 0,88 |
| Папка-скоросшиватель | штука | 1 | 0,83 | 0,17 | 1,00 |
| Диск | штука | 1 | 0,95 | 0,19 | 1,14 |
| Всего расходов | | | | | 11,68 |
| Всего с транспортно-заготовительными расходами | | | | | 12,26 |

Затраты на электроэнергию находятся исходя из продолжительности периода разработки программного обеспечения, количества кВт/ч, затраченных на его проектирование и тарифа за 1 кВт/ч. по следующей формуле

, (8.2)

где– стоимость одного кВт/ч, руб.;

– количество кВт/ч.

Тарифы на электроэнергию применяются согласно приложения к Декларации *«*Об уровне тарифов на электроэнергию, отпускаемую РУП Электроэнергетики ГПО «Белэнерго» для юридических лиц и ИП» на соответствующий период времени, когда разрабатывается программное обеспечение.

Базовый тариф для прочих потребителей с 01.01.2020 г. составляет 0,33048 руб. за 1 кВт/ч. Время реализации проекта 40 дней, среднее потребление энергии в месяц составило 75 кВт/ч, то есть было потреблено 143 кВт/ч. Исходя из вышеизложенного получаем, что на электроэнергию было затрачено

Рэ = 143 × 0,33048 = 47,26 руб.

Определяется основная заработная плата научно-технического персонала, непосредственно занятого выполнением работ.

К этой статье относятся основная заработная плата работников, а также премии, входящие в фонд заработной платы. Среднее количество рабочих дней в месяце равно 21, а средняя продолжительность рабочего дня составляет восемь часов. Следовательно, часовая заработная плата определяется делением размера оклада на количество рабочих часов в месяце (т.е. на 168 часов).

Тарифная ставка первого разряда на предприятии составляет 66 руб. Трудоемкость определяется исходя из данных, представленных в таблице 8.1.

Произведение трудоемкости на сумму часовой заработной платы определяет затраты по зарплате для работника на все время разработки.

Расчет основной заработной платы по теме приведен в таблице 8.3.

Таблица 8.3 – Расчет затрат на основную заработную плату научно-производственного персонала

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование этапов работ | Исполнитель | Разряд | Тарифный  коэффициент | Часовая тарифная ставка, руб. | Трудоемкость,  чел/час | Затраты по заработной плате, руб. |
| Подготовительный | техник-программист | 7 | 2,03 | 0,81 | 24 | 19,44 |
| Выбор методов и средств | техник-  программист | 7 | 2,03 | 0,81 | 48 | 38,88 |
| Разработка алгоритмов и программ | техник-  программист | 7 | 2,03 | 0,81 | 88 | 64,80 |
| Отладка программ и анализ результатов | техник-  программист | 7 | 2,03 | 0,81 | 128 | 110,16 |
| Оформление документации и подготовка к сдаче разработки | техник-  программист | 7 | 2,03 | 0,81 | 32 | 25,92 |
| Всего | | | | | | 259,20 |
| Всего с коэффициентом премий | | | | | | 311,04 |

Определяется дополнительная заработная плата исполнителей, включающая разнообразные предусмотренные трудовым законодательством выплаты, по формуле (8.3)

, (8.3)

где Ндз – норматив дополнительной заработной платы, Ндз ≈от 10 до 25%.

Рдз= 311,04 × = 46,66 руб.

Рассчитываются отчисления органам социальной защиты по формуле (8.4)

, (8.4)

где Нос – норма отчислений на социальную защиту, Нос=34%.

Рос= (311,04+ 46,66) × = 121,62 руб.

Также рассчитываются отчисления на страхование от несчастных случаев на производстве и профессиональных заболеваний (Остр) по ставке действующего законодательства (Нбгс принимается равным от 0,3 до 0,9%). Для расчетов среднее значение Нбгс  принимается равным 0,6%. Остр рассчитывается по формуле (8.5)

 (8.5)

Остр= (311,04+ 46,66)× = 2,15 руб.

Определяются прочие прямые расходы, связанные с амортизационными отчислениями на полное восстановление основных производственных фондов, арендная плата и лизинговые платежи, компенсация за износ (амортизацию) использованного в процессе создания научно-технической продукции оборудования по договоренности. Рпр рассчитывается по формуле (8.6)

, (8.6)

где Нпр– норматив прямых расходов, Нпр≈от 10 до 20%.

Рпр= 311,04×= 49,77 руб.

Исчисляются косвенные (накладные) расходы по формуле (8.7)

, (8.7)

где Нкос – норматив косвенных расходов, Нкос≈от 50 до 100%.

Ркос= 311,04 ×= 217,73 руб.

Определяется полная себестоимость научно-технической продукции как сумма всех групп затрат по формуле (8.8)

 (8.8)

Сп= 12,26 + 47,26 + 311,04 + 46,66 + 121,62 + 2,15 + 49,77 + 217,73 = 808,49 руб.

По среднему уровню рентабельности в процентах от полной себестоимости определяется плановая прибыль единицы научно-технической продукции по формуле (8.9)

, (8.9)

где Ур – средний уровень рентабельности, Ур ≈от 10 до 30%.

П= 808,49 ×=161,70 руб.

Определяется приближенная (ориентировочная) отпускная цена научно-технической продукции по формуле (8.10)

 (8.10)

Цотп= 808,49 + 161,70 = 970,19 руб.

Определяется налог на добавленную стоимость (НДС) по формуле (8.11)

, (8.11)

где ННДС – ставка налога НДС, ННДС=20%.

НДС= 970,19 × =194,04 руб.

Определяется цена научно-технической продукции с учетом НДС по формуле (8.12)

 (8.12)

Цотп= 970,19 + 194,04 = 1164,23 руб.

Все приведенные выше расчеты целесообразно объединить в сводную таблицу 8.4.

Таблица 8.4 – Расчет ориентировочной цены научно-технической продукции

|  |  |  |
| --- | --- | --- |
| Статьи затрат | Условные обозначения | Сумма, руб. |
| Материалы, покупные полуфабрикаты и комплектующие изделия | Рм | 12,26 |
| Электроэнергия | Рэ | 47,26 |
| Основная заработная плата научно-производственного персонала | Роз | 311,04 |
| Дополнительная заработная плата научно-производственного персонала | Рдз | 46,66 |
| Отчисления на социальную защиту | Рос | 121,62 |
| Отчисления на страхование от несчастных случаев | Остр | 2,15 |
| Прочие прямые расходы | Рпр | 49,77 |
| Накладные расходы | Ркос | 217,73 |
| Полная себестоимость | Сп | 808,49 |
| Плановые накопления (прибыль) | П | 161,70 |
| Отпускная цена (без НДС) | Ц | 970,19 |
| Налог на добавленную стоимость | РНДС | 194,04 |
| Отпускная цена с НДС | Цотп | 1164,23 |

## Экономическая эффективность разработки

В данном разделе представлено экономическое обоснование для дипломного проекта по теме программное средство для управления задачами в ООО «Сервис-парнёр».

В технико-экономическом обосновании были рассмотрены следующие вопросы:

* составление плана по разработке программы;
* расчет стоимости разработки.

В первой части технико-экономического обоснования был рассчитан срок разработки по созданию программного средства. Он составил 320 часов.

Во второй части технико-экономического обоснования была рассчитана стоимость разработки программного продукта, которая составила 1164,23 рублей.

Основное преимущество разработки программы состоит в пользе для людей. Так как можно не только продать товар в котором человеку нет нужды, но и купить необходимую вещь по невысокой цене.

Разработка программы актуальна в современных условиях, так как всегда были и будут вещи, купленные и неиспользуемые людьми или же те вещи, которые использовались ранее. А также идея продажи ненужных вещей за деньги не может не радовать людей. Таким образом можно понять, что это приложение является очень полезным инструментом в наше время.

# Заключение

В рамках дипломного проектирования был реализован следующий функционал: регистрация и авторизация пользователей, лента объявлений с постраничной загрузкой данных в необходимом количестве, просмотр полной информации об объявлении, его продавце и всех активных объявлениях данного продавца, поиск объявлений по критериями, фильтр объявлений по категориям для более удобной сортировки объявлений, возможность зарегистрированных пользователей добавлять объявления в закладки, создавать новые объявления, иметь доступ к своему аккаунту с последующим редактированием и настройками личной информации, просматривать свои объявления в отсортированном виде, добавлен раздел для администратора с одобрением и отклонением объявлений на модерации.

В процессе разработки мобильного приложения применены самые современные стандарты написания мобильных приложений в визуальном и функциональном плане. Мобильное приложение упрощает поиск, просмотр, продажу и покупку товаров бывших в употреблении. Возникающие в процессе разработки мобильного приложения проблемы впоследствии решались либо путем более глубокого изучения, либо путем поиска новых путей решения.

Первоочередной задачей являлась разработка серверной части мобильного приложения. Изучение этого вопроса привело к выбору скриптового языка программирования PHP, так как работа с ним была наиболее эффективной. В процессе выбора хоста для базы данных и скриптов стало понятно, что лучшим вариантом является хостинг «000webhost», его главными особенностями являются бесплатность, время отклика и количество выделяемой памяти.

Одной из главных задач, поставленных при разработке мобильного приложения, являлось создание удобной ленты с объявлениями. Первоначально разработанные варианты не удовлетворяли современным требованиям к мобильному приложению. После проработки существующих вариантов определено, что лучшим решением является реализация ленты с применением новой технологии – библиотеки «Paging Library», разработанной компанией Google в 2017 году. Благодаря указанной библиотеке качество ленты объявлений существенно возросло: сократилось количество кода, также упростилась работа с постраничным выводом объявлений с сервера в необходимом количестве. Процесс обновления ленты теперь занимает гораздо меньше времени. Наличие готовых решений привело к упрощению процесса разработки и сокращению нежелательных ситуаций в процессе эксплуатации ленты, что и позволило безоговорочно выбрать данную технологию.

Для реализации эстетичной и интуитивно понятной визуализации мобильного приложения выбрано использование стиля графического дизайна Material Design, данный стиль расширяет перечень цветов, иконой, шрифтов, анимаций и кнопок. Благодаря Material Design приложение приобрело более естественный, эстетичный и интуитивно понятный вид.

Мобильное приложение планируется выпустить в магазин Android приложений Play Store, в данный момент его нельзя скачать в интернете. Установка мобильного приложения не занимает много времени, так как все данные хранятся на сервере. Размер APK файла не превышает 5.5 Мбайт.

В разделе охраны труда и окружающей среды были рассмотрены вопросы:

* правовые, нормативные, социально – экономические и организационные вопросы охраны труда;
* разработка мер по нормализации и защиты от ультрафиолетового излучения;
* пожарная безопасность;
* охрана окружающей среды.

В технико-экономическом обосновании были рассмотрены следующие вопросы:

* составление плана по разработке приложения;
* расчет стоимости разработки.

В первой части технико-экономического обоснования был рассчитан срок разработки по созданию мобильного приложения. Он составил 320 часов.

Во второй части технико-экономического обоснования была рассчитана стоимость разработки приложения, которая составила 1164,23 рублей.

Поставленная задача выполнена, все разделы приложения созданы и функционируют согласно техническому заданию. Все проектные решения принимались исходя из технического задания и успешно реализованы. Разработанное мобильное приложение полностью соответствует разработанному техническому заданию.

# Список информационных источников

1. Багласова, Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий» / Т.Г. Багласова. – Минск : КБП, 2017. – 30 c.
2. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск : КБП, 2013. – 29 c.
3. Бондарь, А.Г. Microsoft SQL Server 2012 / А.Г. Бондарь. – СПб. : БХВ-Петербург, 2013. – 608 с.
4. Жемеров Д. Котлин в действии / Д. Женеров, С. Исакова. – ДМК Пресс, 2018 г.
5. Мишель Е. Дэвис Изучаем PHP и MySQL / Дэвис Е. Мишель. – Символ-Плюс, 2013 г.
6. Шилдт, Герберт. Ш57 Java8. Полное руководство, 9-е изд.:Пер. с англ. – М.:ООО«И.Д. Вильямс», 2015. – 1376 с. :ил. – Парал. тит. англ.
7. Орлов, С. А. Технологии разработки программного обеспечения: Учебник для вузов. 4-е изд. / С. А. Орлов, Б. Я. Цилькер. – СПб. : Питер, 2012. – 608 с.
8. Михнюк, Т.Ф. Охрана труда / Т.Ф. Михнюк. – Минск : ИВЦ Минфина, 2009. – 365 с.
9. Экономика предприятия. Практикум / Э. В. Крум [и др.] ; под ред. Э. В. Крум. – Минск : Издательство Гревцова, 2009. – 355 с.
10. Общие требования к тестовым документам : ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.
11. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с.
12. Текст программы. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.
13. Microsoft Office [Электронный ресурс] / Microsoft Word: правка документов и общий доступ. – Google, 2020. – Режим доступа : https://play.google.com/store/apps/details?id=com. microsoft.office.word&hl=ru. – Дата доступа : 20.05.2020.
14. UMLet – Free UML Tools for fast UML diagrams [Электронный ресурс] / UMLet 14.3. – Режим доступа: https://www.umlet.com. – Дата доступа : 04.05.2020.
15. Обзор обновлений и новых функций Windows 10 [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : https://www.microsoft.com/ru-ru/windows/features. – Дата доступа : 22.05.2020.

# Приложение А

(обязательное)

Текст программы

import android.app.Activity;

import android.app.Dialog;

import android.content.Intent;

import android.content.SharedPreferences;

import android.graphics.Color;

import android.graphics.drawable.ColorDrawable;

import android.os.Bundle;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import androidx.annotation.NonNull;

import androidx.appcompat.widget.AppCompatButton;

import androidx.appcompat.widget.Toolbar;

import androidx.fragment.app.Fragment;

import androidx.fragment.app.FragmentTransaction;

import android.os.Handler;

import android.view.View;

import androidx.core.view.GravityCompat;

import android.view.MenuItem;

import com.google.android.material.navigation.NavigationView;

import com.squareup.picasso.MemoryPolicy;

import com.squareup.picasso.NetworkPolicy;

import com.squareup.picasso.Picasso;

import androidx.drawerlayout.widget.DrawerLayout;

import androidx.appcompat.app.AppCompatActivity;

import android.view.Menu;

import android.view.inputmethod.InputMethodManager;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

import java.util.ArrayList;

import java.util.List;

import ragalik.baraxolka.other\_logic.account.Account;

import ragalik.baraxolka.other\_logic.activities.SettingsActivity;

import ragalik.baraxolka.other\_logic.ad\_creator.AdCreator;

import ragalik.baraxolka.other\_logic.ad\_creator.AdCreatorActivity;

import ragalik.baraxolka.paging\_feed.search.SearchActivity;

import ragalik.baraxolka.other\_logic.entrance.LogIn;

import ragalik.baraxolka.other\_logic.entrance.SignIn;

import ragalik.baraxolka.network.ApiClient;

import ragalik.baraxolka.network.entities.AdsCount;

import ragalik.baraxolka.paging\_feed.ads.ADS;

import ragalik.baraxolka.paging\_feed.moderator.AdModerator;

import ragalik.baraxolka.other\_logic.main\_fragments.Administrator;

import ragalik.baraxolka.paging\_feed.favourites.FAVOURITES;

import ragalik.baraxolka.other\_logic.main\_fragments.RULES;

import ragalik.baraxolka.other\_logic.main\_fragments.TechnicalSUPPORT;

import ragalik.baraxolka.paging\_feed.my\_ads.MyADS;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class MainActivity extends AppCompatActivity

implements NavigationView.OnNavigationItemSelectedListener {

public static final String SERVER\_URL = "https://imvaper.000webhostapp.com/scripts/";

private LogIn logInFragment;

public static ADS adsFragment;

private SignIn signInFragment;

private Dialog myDialog;

public static SharedPreferences sp;

private static List<String> listItem;

public static FloatingActionButton fab;

private static TextView moderator\_count;

public static Activity activity;

public static NavigationView navigationView;

private Toolbar toolbar;

private int entrance\_count = 0;

public static Boolean isActualFragment = false;

public static DrawerLayout drawer;

private boolean doubleBackToExitPressedOnce;

public static boolean isEntranceFromDialog = false;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

sp = getPreferences(MODE\_PRIVATE);

// toolbar = findViewById(R.id.toolbar);

// setSupportActionBar(toolbar);

activity = this;

listItem = new ArrayList<>();

myDialog = new Dialog(this);

logInFragment = new LogIn();

adsFragment = new ADS();

moderator\_count = new TextView(this);

SharedPreferences.Editor editor = sp.edit();

editor.putInt("entrance\_counter", sp.getInt("entrance\_counter", 0) + 1);

editor.apply();

SharedPreferences sharedPreferences = getSharedPreferences("MenuItems", MODE\_PRIVATE);

String resultString = sharedPreferences.getString("Menu", "[]");

resultString = resultString.substring(1, resultString.length() - 1);

if (!resultString.isEmpty()) {

if (resultString.contains(",")) {

String[] items = resultString.split(",");

for (String item : items) {

listItem.add(item.trim());

}

} else {

listItem.add(resultString);

}

}

fab = findViewById(R.id.fab1);

fab.setOnClickListener(view -> {

Intent myIntent = new Intent(MainActivity.this, AdCreatorActivity.class);

startActivity(myIntent);

});

drawer = findViewById(R.id.drawer\_layout);

drawer.addDrawerListener(new DrawerLayout.DrawerListener() {

@Override

public void onDrawerSlide(@NonNull View drawerView, float slideOffset) {

if(getCurrentFocus() != null)

{

InputMethodManager inputMethodManager = (InputMethodManager) getSystemService(Activity.INPUT\_METHOD\_SERVICE);

inputMethodManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(), 0);

}

updateNotify();

}

@Override

public void onDrawerOpened(@NonNull View drawerView) {}

@Override

public void onDrawerClosed(@NonNull View drawerView) {}

@Override

public void onDrawerStateChanged(int newState) {}

});

navigationView = findViewById(R.id.nav\_view);

navigationView.setNavigationItemSelectedListener(this);

navigationView.setCheckedItem(R.id.ADS);

entrance\_count = sp.getInt("entrance\_counter", 0);

if (entrance\_count == 1 && !isEntered()) {

myDialog.setContentView(R.layout.start\_login\_window);

myDialog.show();

myDialog.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));

AppCompatButton log = myDialog.findViewById(R.id.logButton);

AppCompatButton sign = myDialog.findViewById(R.id.signButton);

this.setTitle("Все объявления");

myDialog.setOnCancelListener(dialog -> {

newTransaction(adsFragment, "Все объявления");

myDialog.dismiss();

});

log.setOnClickListener(v -> {

isEntranceFromDialog = true;

myDialog.dismiss();

newTransaction(logInFragment, "Вход");

});

sign.setOnClickListener(v -> {

isEntranceFromDialog = true;

signInFragment = new SignIn();

myDialog.dismiss();

newTransaction(signInFragment, "Регистрация");

});

hideItemsNavigationDrawer(R.id.MY\_ADS, R.id.FAVOURITES);

} else if (isEntered()) {

newTransaction(adsFragment, "Все объявления");

showItemsNavigationDrawer(R.id.MY\_ADS, R.id.FAVOURITES);

} else if (entrance\_count > 1) {

if (entrance\_count == 10) {

editor.putInt("entrance\_counter", 0);

editor.apply();

}

newTransaction(adsFragment, "Все объявления");

hideItemsNavigationDrawer(R.id.MY\_ADS, R.id.FAVOURITES);

}

}

private boolean isEntered () {

return (!sp.getString("nickname", "").equals(""));

}

public void onClickImage(View v){

if (!isEntered()) {

newTransaction(logInFragment, "Вход");

} else {

Intent intent = new Intent(this, Account.class);

startActivity(intent);

}

DrawerLayout drawer = findViewById(R.id.drawer\_layout);

drawer.closeDrawer(GravityCompat.START);

}

public static void createUserData(int id, String nickname, String email, String phoneNumber, String region, String town, String statusName) {

SharedPreferences.Editor editor = sp.edit();

editor.putInt("id", id);

editor.putString("nickname", nickname);

editor.putString("email", email);

editor.putString("phoneNumber", phoneNumber);

editor.putString("region", region);

editor.putString("town", town);

editor.putString("status\_name", statusName);

editor.apply();

}

public static void invalidateSearchMenu() {

activity.invalidateOptionsMenu();

}

@Override

public boolean onPrepareOptionsMenu(Menu menu) {

MenuItem item = menu.findItem(R.id.option\_search);

if (isActualFragment) {

item.setVisible(true);

} else {

item.setVisible(false);

}

return super.onPrepareOptionsMenu(menu);

}

public static void setNavHeaderText (Activity activity) {

TextView data = activity.findViewById(R.id.nickname);

data.setText(sp.getString("nickname", ""));

}

private void newTransaction (Fragment fragment, String fragmentName) {

FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();

fragmentTransaction.setCustomAnimations(R.anim.enter\_from\_up, R.anim.exit\_to\_up);

fragmentTransaction.replace(R.id.constrLayout, fragment).commit();

this.setTitle(fragmentName);

}

private void readData () {

TextView nickname = findViewById(R.id.nickname);

if (isEntered()) {

nickname.setText(sp.getString("nickname", ""));

}

}

public static void removeData (Activity activity) {

SharedPreferences.Editor editor = sp.edit();

editor.putInt("id", 0);

editor.putString("nickname", "");

editor.putString("email", "");

editor.putString("phoneNumber", "");

editor.putString("image", "");

editor.putString("status\_name", "");

TextView nickname = activity.findViewById(R.id.nickname);

nickname.setText("Войти/Регистрация");

editor.apply();

}

public static void removeSignOut (Activity activity) {

applyChanges(activity);

}

private static void applyChanges (Activity activity) {

SharedPreferences sharedPreferences = activity.getSharedPreferences("MenuItems", MODE\_PRIVATE);

SharedPreferences.Editor editor = sharedPreferences.edit();

editor.putString("Menu", listItem.toString());

editor.apply();

activity.invalidateOptionsMenu();

}

@Override

public void onBackPressed() {

// DrawerLayout drawer = findViewById(R.id.drawer\_layout);

// if (getSupportFragmentManager().getBackStackEntryCount() > 0) {

// getSupportFragmentManager().popBackStack();

// } else {

// super.onBackPressed();

// }

DrawerLayout drawer = findViewById(R.id.drawer\_layout);

if (doubleBackToExitPressedOnce) {

super.onBackPressed();

return;

} else {

if (drawer.isDrawerOpen(GravityCompat.START)) {

drawer.closeDrawer(GravityCompat.START);

}

}

this.doubleBackToExitPressedOnce = true;

Toast.makeText(this, "Нажмите 'назад' еще раз чтобы выйти", Toast.LENGTH\_SHORT).show();

new Handler().postDelayed(new Runnable() {

@Override

public void run() {

doubleBackToExitPressedOnce = false;

}

}, 2000);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.main, menu);

readData();

ImageView navigationPhoto = this.findViewById(R.id.navigationDrawerPhoto);

if (!MainActivity.sp.getString("image", "").equals("null") && !MainActivity.sp.getString("image", "").equals("")) {

String temp = MainActivity.sp.getString("image", "");

Picasso.get().invalidate(temp);

Picasso.get().load(temp).networkPolicy(NetworkPolicy.NO\_CACHE).memoryPolicy(MemoryPolicy.NO\_CACHE).into(navigationPhoto);

} else {

navigationPhoto.setImageResource(R.drawable.gradient\_navigation);

}

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.option\_search:

Intent myIntent = new Intent(MainActivity.this, SearchActivity.class);

startActivity(myIntent);

return true;

default:

return super.onOptionsItemSelected(item);

}

}

@Override

public boolean onNavigationItemSelected(MenuItem item) {

FragmentTransaction fragmentTransaction = this.getSupportFragmentManager().beginTransaction();

// Toolbar toolbar = findViewById(R.id.toolbar);

int id = item.getItemId();

if (id == R.id.ADS) {

fragmentTransaction.replace(R.id.constrLayout, new ADS()).commit();

// toolbar.setTitle("Объявления");

} else if (id == R.id.MY\_ADS) {

fragmentTransaction.replace(R.id.constrLayout, new MyADS()).commit(); //Нужно раскомментить при готовых май адс!

//toolbar.setTitle("Мои объявления");

} else if (id == R.id.FAVOURITES) {

fragmentTransaction.replace(R.id.constrLayout, new FAVOURITES()).commit();

// toolbar.setTitle("Закладки");

}

// else if (id == R.id.RULES) {

// fragmentTransaction.replace(R.id.constrLayout, new RULES()).commit();

// // toolbar.setTitle("Правила");

// }

else if (id == R.id.Technical\_SUPPORT) {

fragmentTransaction.replace(R.id.constrLayout, new TechnicalSUPPORT()).commit();

// toolbar.setTitle("Техническая поддержка");

} else if (id == R.id.SETTINGS) {

Intent myIntent = new Intent(MainActivity.this, SettingsActivity.class);

startActivity(myIntent);

} else if (id == R.id.MODERATOR) {

fragmentTransaction.replace(R.id.constrLayout, new AdModerator()).commit();

} else if (id == R.id.ADMIN) {

fragmentTransaction.replace(R.id.constrLayout, new Administrator()).commit();

}

DrawerLayout drawer = findViewById(R.id.drawer\_layout);

drawer.closeDrawer(GravityCompat.START);

return true;

}

public static void checkUserStatus() {

if (sp.getString("status\_name", "").equals("АДМИНИСТРАТОР")) {

createAdminField();

} else if (sp.getString("status\_name", "").equals("РЕДАКТОР")) {

createEditorField();

}

}

public static void removeGroupFromNV(int id, Activity activity) {

NavigationView navigationView = activity.findViewById(R.id.nav\_view);

Menu menu = navigationView.getMenu();

menu.removeGroup(id);

}

private static void createAdminField() {

NavigationView navigationView = MainActivity.activity.findViewById(R.id.nav\_view);

Menu menu = navigationView.getMenu();

MenuItem administrator = menu.findItem(R.id.ADMIN);

administrator.setVisible(true);

createEditorField();

}

private static void createEditorField() {

NavigationView navigationView = MainActivity.activity.findViewById(R.id.nav\_view);

Menu menu = navigationView.getMenu();

MenuItem moderator = menu.findItem(R.id.MODERATOR);

moderator.setVisible(true);

moderator\_count = (TextView) moderator.getActionView();

}

public static void hideItemsNavigationDrawer(int... items) {

Menu menu = navigationView.getMenu();

for (int item : items) {

menu.findItem(item).setVisible(false);

}

}

public static void showItemsNavigationDrawer(int... items) {

Menu menu = navigationView.getMenu();

for (int item : items) {

menu.findItem(item).setVisible(true);

}

}

private void updateNotify () {

Call<AdsCount> call = ApiClient.getApi().getAdsCount(3);

call.enqueue(new Callback<AdsCount>() {

@Override

public void onResponse(Call<AdsCount> call, Response<AdsCount> response) {

if (response.body() != null) {

int count = response.body().getCount();

moderator\_count.setText("");

if (count != 0) {

moderator\_count.setText(String.valueOf(count));

}

}

}

@Override

public void onFailure(Call<AdsCount> call, Throwable t) {

//Toast.makeText(getApplicationContext(), "Ошибка в рассмотрение", Toast.LENGTH\_LONG).show();

}

});

}

}

package ragalik.baraxolka.paging\_feed.ads

import androidx.core.view.isVisible

import androidx.paging.PageKeyedDataSource

import ragalik.baraxolka.MainActivity

import ragalik.baraxolka.network.ApiClient

import ragalik.baraxolka.paging\_feed.Ad

import ragalik.baraxolka.paging\_feed.AdResponse

import retrofit2.Call

import retrofit2.Callback

import retrofit2.Response

class AdDataSource : PageKeyedDataSource<Int, Ad>() {

override fun loadInitial(params: LoadInitialParams<Int>, callback: LoadInitialCallback<Int, Ad>) {

val call = ApiClient.getApi().getAds(FIRST\_PAGE, 1, MainActivity.sp.getInt("id", 0))

call.enqueue(object : Callback<AdResponse>{

override fun onFailure(call: Call<AdResponse>, t: Throwable) {

}

override fun onResponse(call: Call<AdResponse>, response: Response<AdResponse>) {

if (response.isSuccessful) {

val apiResponse = response.body()!!

val responseItems = apiResponse.ads

ADS.progressBar.isVisible = false

responseItems?.let {

callback.onResult(responseItems, null, FIRST\_PAGE + 4)

}

}

}

})

}

override fun loadAfter(params: LoadParams<Int>, callback: LoadCallback<Int, Ad>) {

val call = ApiClient.getApi().getAds(params.key, 1, MainActivity.sp.getInt("id", 0))

call.enqueue(object : Callback<AdResponse> {

override fun onFailure(call: Call<AdResponse>, t: Throwable) {}

override fun onResponse(call: Call<AdResponse>, response: Response<AdResponse>) {

if (response.isSuccessful) {

val apiResponse = response.body()!!

val responseItems = apiResponse.ads

val key = params.key + 4

responseItems?.let {

callback.onResult(responseItems, key)

}

}

}

})

}

override fun loadBefore(params: LoadParams<Int>, callback: LoadCallback<Int, Ad>) {

val call = ApiClient.getApi().getAds(params.key, 1, MainActivity.sp.getInt("id", 0))

call.enqueue(object : Callback<AdResponse> {

override fun onFailure(call: Call<AdResponse>, t: Throwable) {

}

override fun onResponse(call: Call<AdResponse>, response: Response<AdResponse>) {

if (response.isSuccessful) {

val apiResponse = response.body()!!

val responseItems = apiResponse.ads

val key = if (params.key > 1) params.key - 4 else 0

responseItems?.let {

callback.onResult(responseItems, key)

}

}

}

})

}

companion object {

const val PAGE\_SIZE = 4

const val FIRST\_PAGE = 0

}

}

package ragalik.baraxolka.other\_logic.ad\_creator;

import android.app.Activity;

import android.app.ProgressDialog;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.net.Uri;

import android.os.Bundle;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AppCompatActivity;

import android.view.View;

import android.view.inputmethod.InputMethodManager;

import android.widget.ArrayAdapter;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.widget.AppCompatAutoCompleteTextView;

import androidx.appcompat.widget.AppCompatButton;

import androidx.appcompat.widget.AppCompatTextView;

import androidx.appcompat.widget.Toolbar;

import androidx.fragment.app.DialogFragment;

import androidx.fragment.app.FragmentTransaction;

import com.google.android.material.textfield.TextInputEditText;

import com.google.android.material.textfield.TextInputLayout;

import java.io.File;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.Locale;

import java.util.TimeZone;

import okhttp3.MediaType;

import okhttp3.MultipartBody;

import okhttp3.RequestBody;

import ragalik.baraxolka.R;

import ragalik.baraxolka.MainActivity;

import ragalik.baraxolka.network.ApiClient;

import ragalik.baraxolka.network.entities.CategoryResponse;

import ragalik.baraxolka.network.entities.ServerResponse;

import ragalik.baraxolka.network.entities.Subcategories;

import ragalik.baraxolka.other\_logic.account.FullImageLayout;

import ragalik.baraxolka.other\_logic.account.PathUtils;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class AdCreator extends AppCompatActivity implements View.OnClickListener {

private static final long MAX\_SIZE\_PHOTO = 3145728;

private TextInputLayout titleEditText;

private AppCompatAutoCompleteTextView categorySpinner;

private AppCompatAutoCompleteTextView subcategorySpinner;

private TextInputLayout descriptionEditText;

private TextInputLayout priceEditText;

private AppCompatButton addAdButton;

private String categoryFromSpinner = "";

private String subcategoryFromSpinner = "";

public static ArrayList<ImageView> adImages;

private int choosedIndex = 0;

public static int imageUploadCount;

public static ArrayList<File> files;

private TextInputLayout subcategoryLayout;

private TextInputLayout categoryLayout;

//private AppCompatRadioButton newState;

//private AppCompatRadioButton secondaryState;

//private String goodsState;

// private boolean state;

public static TextView uploadAmountTW;

public static AppCompatTextView mainLabel;

public static String uploadedCounterStr;

public static AppCompatActivity activity;

public static ArrayList<String> uris;

static Uri selectedImageUri;

public static Uri outputFileUri;

public static AppCompatActivity appCompatActivity;

public static File fileWithUri;

private ProgressDialog pDialog;

private static ArrayList<MultipartBody.Part> parts = new ArrayList<>();

@Override

protected void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.fragment\_ad\_creator);

appCompatActivity = this;

Toolbar toolbar = findViewById(R.id.adCreatorToolbar);

toolbar.setTitle("Создание объявления");

setSupportActionBar(toolbar);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

ArrayList<String> arr = new ArrayList<>();

arr.add("https://duckduckgo.com/?q=ima&atb=v218-5&iax=images&ia=images&iai=https%3A%2F%2Fi.ytimg.com%2Fvi%2FuBQW2310F-A%2Fmaxresdefault.jpg");

FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();

fragmentTransaction.replace(R.id.adCreatorCoordinator, new FullImageLayout(0, arr)).addToBackStack("").commit();

subcategorySpinner = findViewById(R.id.subcategorySpinner);

categorySpinner = findViewById(R.id.categorySpinner);

mainLabel = findViewById(R.id.ad\_creator\_main\_label);

subcategoryLayout = findViewById(R.id.subcategoryLayout);

categoryLayout = findViewById(R.id.categoryLayout);

activity = this;

uploadAmountTW = findViewById(R.id.uploadPhotoCounterAdCreator);

uploadedCounterStr = "Загружено 0 из 5";

adImages = new ArrayList<>();

files = new ArrayList<>();

uris = new ArrayList<>();

imageUploadCount = 0;

MultipartBody.Part image1 = null;

MultipartBody.Part image2 = null;

MultipartBody.Part image3 = null;

MultipartBody.Part image4 = null;

MultipartBody.Part image5 = null;

parts.add(image1);

parts.add(image2);

parts.add(image3);

parts.add(image4);

parts.add(image5);

adImages.add((ImageView) findViewById(R.id.adPhoto1));

adImages.add((ImageView) findViewById(R.id.adPhoto2));

adImages.add((ImageView) findViewById(R.id.adPhoto3));

adImages.add((ImageView) findViewById(R.id.adPhoto4));

adImages.add((ImageView) findViewById(R.id.adPhoto5));

adImages.get(0).setBackground(getResources().getDrawable(R.drawable.ad\_creator\_main\_photo\_dark\_theme));

adImages.get(0).setImageDrawable(getResources().getDrawable(R.drawable.ic\_circle\_add));

adImages.get(0).setOnClickListener(this);

adImages.get(1).setOnClickListener(this);

adImages.get(2).setOnClickListener(this);

adImages.get(3).setOnClickListener(this);

adImages.get(4).setOnClickListener(this);

titleEditText = findViewById(R.id.titleEditText);

createCategorySpinner();

// newState = findViewById(R.id.newState);

// secondaryState = findViewById(R.id.secondaryState);

// newState.setOnClickListener(this);

// secondaryState.setOnClickListener(this);

descriptionEditText = findViewById(R.id.descriptionEditText);

priceEditText = findViewById(R.id.priceEditText);

addAdButton = findViewById(R.id.addAdButton);

addAdButton.setOnClickListener(this);

}

public void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (resultCode == RESULT\_OK) {

if (requestCode == 1) {

if (fileWithUri == null) {

fileWithUri = new File(PathUtils.getPath(this, data.getData()));

if (data.getData() != null) {

selectedImageUri = data.getData();

}

} else {

selectedImageUri = outputFileUri;

}

ImageView imageView = new ImageView(this);

imageView.setImageDrawable(getResources().getDrawable(R.drawable.ad\_creator\_image\_style));

addImage();

}

}

}

private void createCategorySpinner() {

ApiClient.getApi().getCategoriesWithSubcategories().enqueue(new Callback<CategoryResponse>() {

@Override

public void onResponse(Call<CategoryResponse> call, Response<CategoryResponse> response) {

HashMap<String, ArrayList<Subcategories>> subcategoriesHashMap = new HashMap<>();

ArrayList<String> categories = new ArrayList<>();

categories.add("Не выбрано");

for (int i = 0; i < response.body().getCategories().size(); ++i) {

categories.add(response.body().getCategories().get(i).getCategory\_name());

subcategoriesHashMap.put(response.body().getCategories().get(i).getCategory\_name(), response.body().getCategories().get(i).getSubcategories());

}

categorySpinner.setAdapter(new ArrayAdapter<String>(getApplicationContext(), R.layout.dropdown\_text\_color, categories));

categorySpinner.setOnClickListener(v -> categorySpinner.showDropDown());

categorySpinner.setOnItemClickListener((parent, view, position, id) -> {

ArrayList<String> subcategories = new ArrayList<>();

subcategories.add("Не выбрано");

if (subcategoriesHashMap.get(parent.getItemAtPosition(position).toString()) != null) {

if (position != 0) {

subcategoryLayout.setVisibility(View.VISIBLE);

for (int i = 0; i < subcategoriesHashMap.get(parent.getItemAtPosition(position).toString()).size(); ++i) {

subcategories.add(subcategoriesHashMap.get(parent.getItemAtPosition(position).toString()).get(i).getSubcategory\_name());

}

categoryFromSpinner = parent.getItemAtPosition(position).toString();

if (categoryFromSpinner.equals("Не выбрано")) {

categoryFromSpinner = "";

}

ArrayAdapter<String> adapterSubcategory = new ArrayAdapter<String>(getApplicationContext(), R.layout.dropdown\_text\_color, subcategories);

subcategorySpinner.setAdapter(adapterSubcategory);

subcategorySpinner.setOnClickListener(v -> subcategorySpinner.showDropDown());

subcategoryFromSpinner = "";

subcategorySpinner.setText(adapterSubcategory.getItem(0), false);

if (subcategoriesHashMap.get(parent.getItemAtPosition(position).toString()).size() == 1) {

subcategoryFromSpinner = adapterSubcategory.getItem(1);

subcategorySpinner.setText(subcategoryFromSpinner, false);

}

} else {

subcategoryLayout.setVisibility(View.GONE);

}

} else {

subcategoryLayout.setVisibility(View.GONE);

}

});

subcategorySpinner.setOnItemClickListener((parent, view, position, id) -> {

if (parent.getItemAtPosition(position).toString().equals("Не выбрано")) {

subcategoryFromSpinner = "";

} else {

subcategoryFromSpinner = parent.getItemAtPosition(position).toString();

}

});

}

@Override

public void onFailure(Call<CategoryResponse> call, Throwable t) {

Toast.makeText(AdCreator.this, "Проверьте интернет соединение.", Toast.LENGTH\_SHORT).show();

}

});

}

private void addImage() {

if (fileWithUri.length() < MAX\_SIZE\_PHOTO) {

adImages.get(choosedIndex).setImageURI(selectedImageUri);

if (choosedIndex == imageUploadCount) {

++imageUploadCount;

}

uris.add(choosedIndex, selectedImageUri.toString());

files.add(choosedIndex, fileWithUri);

if (imageUploadCount < 5) {

adImages.get(imageUploadCount).setBackground(getResources().getDrawable(R.drawable.ad\_creator\_image\_style));

adImages.get(imageUploadCount).setImageDrawable(getResources().getDrawable(R.drawable.ic\_circle\_add));

if (imageUploadCount > 1) {

adImages.get(imageUploadCount - 1).setBackground(getResources().getDrawable(R.drawable.ad\_creator\_secondary\_photo));

}

} else if (imageUploadCount == 5) {

adImages.get(imageUploadCount - 1).setBackground(getResources().getDrawable(R.drawable.ad\_creator\_secondary\_photo));

}

uploadedCounterStr = uploadedCounterStr.substring(0, 10) + imageUploadCount + uploadedCounterStr.substring(11, 16);

uploadAmountTW.setText(uploadedCounterStr);

if (imageUploadCount == 1) {

mainLabel.setVisibility(View.VISIBLE);

}

fileWithUri = null;

} else {

Toast.makeText(activity, "Размер фото превышает 3 мб.", Toast.LENGTH\_SHORT).show();

fileWithUri = null;

}

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {

super.onRequestPermissionsResult(requestCode, permissions, grantResults);

if (requestCode == 1) {

if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {} else {

Toast.makeText(activity, "Отказано в доступе", Toast.LENGTH\_SHORT).show();

}

}

}

@Override

public void onClick(View v) {

boolean flag = false;

switch (v.getId()) {

// case R.id.newState :

// state = true;

// newState.setChecked(true);

// secondaryState.setChecked(false);

// goodsState = "Новое";

// break;

// case R.id.secondaryState :

// state = false;

// secondaryState.setChecked(true);

// newState.setChecked(false);

// goodsState = "Б/У";

// break;

case R.id.adPhoto1:

choosedIndex = 0;

flag = true;

break;

case R.id.adPhoto2:

if (imageUploadCount >= 1) {

choosedIndex = 1;

flag = true;

}

break;

case R.id.adPhoto3:

if (imageUploadCount >= 2) {

choosedIndex = 2;

flag = true;

}

break;

case R.id.adPhoto4:

if (imageUploadCount >= 3) {

choosedIndex = 3;

flag = true;

}

break;

case R.id.adPhoto5:

if (imageUploadCount >= 4) {

choosedIndex = 4;

flag = true;

}

break;

case R.id.addAdButton:

if (isAdCorrect()) {

insertAd();

}

InputMethodManager inputMethodManager = (InputMethodManager) getSystemService(Activity.INPUT\_METHOD\_SERVICE);

inputMethodManager.hideSoftInputFromWindow(v.getWindowToken(), 0);

break;

}

if (flag) {

AdCreatorImageMenu adCreatorImageMenu = new AdCreatorImageMenu(imageUploadCount, choosedIndex, uris);

adCreatorImageMenu.setStyle(DialogFragment.STYLE\_NORMAL, R.style.BottomSheetDialogTheme);

adCreatorImageMenu.show(getSupportFragmentManager(), "");

}

}

private boolean isAdCorrect () {

boolean isAdCorrect = true;

titleEditText.setErrorEnabled(false);

categoryLayout.setErrorEnabled(false);

subcategoryLayout.setErrorEnabled(false);

descriptionEditText.setErrorEnabled(false);

priceEditText.setErrorEnabled(false);

if (files.size() == 0) {

Toast.makeText(activity, "Отсутствуют фотографии. Добавьте хотя бы одну.", Toast.LENGTH\_LONG).show();

isAdCorrect = false;

}

if (categoryFromSpinner.equals("")) {

categoryLayout.setError("Укажите категорию");

isAdCorrect = false;

} else if (subcategoryFromSpinner.equals("")) {

subcategoryLayout.setError("Укажите подкатегорию");

isAdCorrect = false;

}

if (titleEditText.getEditText().getText().toString().equals("")) {

titleEditText.setError("Укажите заголовок");

isAdCorrect = false;

}

if (descriptionEditText.getEditText().getText().toString().equals("")) {

descriptionEditText.setError("Укажите описание");

isAdCorrect = false;

}

if (priceEditText.getEditText().getText().toString().equals("")) {

priceEditText.setError("Укажите стоимость");

isAdCorrect = false;

}

return isAdCorrect;

}

private void insertAd () {

pDialog = new ProgressDialog(activity);

pDialog.setMessage("Создание объявления");

pDialog.setIndeterminate(false);

pDialog.setCancelable(true);

pDialog.show();

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.forLanguageTag("ru"));

dateFormat.setTimeZone(TimeZone.getTimeZone("GMT+3"));

String title = titleEditText.getEditText().getText().toString();

String description = descriptionEditText.getEditText().getText().toString();

for (int i = 0; i < uris.size(); ++i) {

parts.set(i, MultipartBody.Part.createFormData("image" + (i + 1), "-" + (i + 1) + ".png", RequestBody.create(MediaType.parse("image/\*"), files.get(i))));

}

int price = Integer.parseInt(priceEditText.getEditText().getText().toString());

String nickname = MainActivity.sp.getString("nickname", "");

String datetime = dateFormat.format(new Date());

String subcategory = subcategoryFromSpinner;

ApiClient.getApi().insertAd(RequestBody.create(MediaType.parse("multipart/form-data"), title),

RequestBody.create(MediaType.parse("multipart/form-data"), description),

parts.get(0),

parts.get(1),

parts.get(2),

parts.get(3),

parts.get(4),

RequestBody.create(MediaType.parse("multipart/form-data"), String.valueOf(price)),

RequestBody.create(MediaType.parse("multipart/form-data"), nickname),

RequestBody.create(MediaType.parse("multipart/form-data"), datetime),

RequestBody.create(MediaType.parse("multipart/form-data"), subcategory)).enqueue(new Callback<ServerResponse>() {

@Override

public void onResponse(Call<ServerResponse> call, Response<ServerResponse> response) {

pDialog.dismiss();

if (response.body() != null) {

Toast.makeText(getApplicationContext(), "Обьявление создано!", Toast.LENGTH\_LONG).show();

Intent myIntent = new Intent(appCompatActivity, MainActivity.class);

appCompatActivity.startActivity(myIntent);

}

}

@Override

public void onFailure(Call<ServerResponse> call, Throwable t) {

pDialog.dismiss();

Toast.makeText(getApplicationContext(), "Произошла ошибка", Toast.LENGTH\_LONG).show();

}

});

}

}

package ragalik.baraxolka.other\_logic.account

import android.content.Intent

import android.os.Bundle

import android.view.View

import android.widget.AdapterView

import android.widget.AdapterView.OnItemClickListener

import android.widget.ArrayAdapter

import android.widget.Toast

import androidx.appcompat.app.AppCompatActivity

import com.r0adkll.slidr.Slidr

import kotlinx.android.synthetic.main.activity\_account\_edit.\*

import kotlinx.android.synthetic.main.content\_account\_edit.\*

import ragalik.baraxolka.MainActivity

import ragalik.baraxolka.R

import ragalik.baraxolka.network.ApiClient

import ragalik.baraxolka.network.entities.ServerResponse

import ragalik.baraxolka.other\_logic.entrance.SignIn

import retrofit2.Call

import retrofit2.Callback

import retrofit2.Response

class AccountEditActivity : AppCompatActivity() {

val PHONE\_NUMBER : String = "(^\\+375\\d{9})"

lateinit var activity : AppCompatActivity

companion object {

private lateinit var edit\_flag : String

private var isRegionSelected : Boolean = false

private var regionFromSpinner: String = ""

private var townFromSpinner : String = ""

}

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_account\_edit)

setSupportActionBar(toolbar\_edit\_activity)

supportActionBar?.setDisplayHomeAsUpEnabled(true)

Slidr.attach(this)

val bundle = intent.extras

bundle.let {

edit\_flag = (bundle?.get("edit\_flag") as String)

}

activity = this

when (edit\_flag) {

"phone" -> {

account\_edit\_layout.hint = "Номер телефона"

tw\_account\_edit.text = "После указания номера телефона, его НЕЛЬЗЯ будет изменить!"

title = "Добавление номера тел."

tw\_account\_edit\_title.text = "Укажите номер телефона"

edit\_region\_layout.visibility = View.GONE

}

"region" -> {

account\_edit\_layout.visibility = View.GONE

tw\_account\_edit.text = "Вы можете изменить регион в любой момент."

title = "Указание региона"

tw\_account\_edit\_title.visibility = View.GONE

EditRegionSpinner.setAdapter(ArrayAdapter.createFromResource(this, R.array.Spinner\_region\_items, R.layout.dropdown\_text\_color))

EditRegionSpinner.setOnClickListener {

EditRegionSpinner.showDropDown()

}

EditRegionSpinner.setOnItemClickListener { parent: AdapterView<\*>, \_: View?, position: Int, \_: Long ->

val text = parent.getItemAtPosition(position).toString()

isRegionSelected = true

tw\_select\_region.text = "Укажите область проживания"

EditRegionTextEdit.hint = "Выберите область"

tw\_select\_town.text = "Укажите город проживания"

EditTownTextEdit.hint = "Выберите город"

lateinit var adapterTown : ArrayAdapter<CharSequence>

when (text) {

"Не указано" -> {

adapterTown = ArrayAdapter.createFromResource(this, R.array.Nothing, R.layout.dropdown\_text\_color)

isRegionSelected = false

}

"Минск" -> {

adapterTown = ArrayAdapter.createFromResource(this, R.array.Minsk, R.layout.dropdown\_text\_color)

tw\_select\_region.text = "Укажите город проживания"

EditRegionTextEdit.hint = "Выберите город"

tw\_select\_town.text = "Укажите район проживания"

EditTownTextEdit.hint = "Выберите район"

}

"Брестская обл." -> adapterTown = ArrayAdapter.createFromResource(this, R.array.BrestRegion, R.layout.dropdown\_text\_color)

"Витебская обл." -> adapterTown = ArrayAdapter.createFromResource(this, R.array.VitebskRegion, R.layout.dropdown\_text\_color)

"Гомельская обл." -> adapterTown = ArrayAdapter.createFromResource(this, R.array.GomelRegion, R.layout.dropdown\_text\_color)

"Гродненская обл." -> adapterTown = ArrayAdapter.createFromResource(this, R.array.GrodnoRegion, R.layout.dropdown\_text\_color)

"Минская обл." -> adapterTown = ArrayAdapter.createFromResource(this, R.array.MinskRegion, R.layout.dropdown\_text\_color)

"Могилевская обл." -> adapterTown = ArrayAdapter.createFromResource(this, R.array.MogilevRegion, R.layout.dropdown\_text\_color)

else -> {}

}

regionFromSpinner = text

EditTownSpinner.setAdapter(adapterTown)

townFromSpinner = adapterTown.getItem(0).toString()

EditTownSpinner.setText(townFromSpinner, false)

EditTownSpinner.setOnClickListener {

EditTownSpinner.showDropDown()

}

EditTownSpinner.onItemClickListener = OnItemClickListener { parent: AdapterView<\*>, \_: View?, position: Int, \_: Long -> townFromSpinner = parent.getItemAtPosition(position).toString() }

}

}

}

saveEditInfoButton.setOnClickListener {

val email = MainActivity.sp.getString("email", "")

when (edit\_flag) {

"phone" -> {

val phoneNumber = account\_edit\_layout.editText?.text.toString()

if (phoneNumber.matches(PHONE\_NUMBER.toRegex())) {

val call = ApiClient.getApi().setPhoneNumber(phoneNumber, email)

call.enqueue(object : Callback<ServerResponse> {

override fun onResponse(call: Call<ServerResponse>, response: Response<ServerResponse>) {

val intent = Intent(activity, Account::class.java)

startActivity(intent)

Toast.makeText(activity, "Добавлен номер телефона!", Toast.LENGTH\_LONG).show()

}

override fun onFailure(call: Call<ServerResponse>, t: Throwable) {

Toast.makeText(activity, "Такой номер телефона уже существует!", Toast.LENGTH\_LONG).show()

}

})

} else {

Toast.makeText(activity, "Введен некорректный номер тел.", Toast.LENGTH\_LONG).show()

}

}

"region" -> {

account\_edit\_layout.visibility = View.GONE

if (!isRegionSelected) {

regionFromSpinner = "0"

townFromSpinner = "0"

}

val call = ApiClient.getApi().setRegion(regionFromSpinner, townFromSpinner, email)

call.enqueue(object : Callback<ServerResponse> {

override fun onResponse(call: Call<ServerResponse>, response: Response<ServerResponse>) {

val intent = Intent(activity, Account::class.java)

startActivity(intent)

Toast.makeText(activity, "Регион изменен", Toast.LENGTH\_LONG).show()

}

override fun onFailure(call: Call<ServerResponse>, t: Throwable) {}

})

}

}

}

}

}

package ragalik.baraxolka.network;

import java.util.List;

import okhttp3.MultipartBody;

import okhttp3.RequestBody;

import ragalik.baraxolka.network.entities.AdsCount;

import ragalik.baraxolka.network.entities.CategoryResponse;

import ragalik.baraxolka.network.entities.ReasonsResponse;

import ragalik.baraxolka.network.entities.RegisterResponse;

import ragalik.baraxolka.network.entities.ServerResponse;

import ragalik.baraxolka.network.entities.User;

import ragalik.baraxolka.network.entities.UserResponse;

import ragalik.baraxolka.paging\_feed.AdResponse;

import ragalik.baraxolka.paging\_feed.FullAdResponse;

import retrofit2.Call;

import retrofit2.http.Field;

import retrofit2.http.FormUrlEncoded;

import retrofit2.http.GET;

import retrofit2.http.Multipart;

import retrofit2.http.POST;

import retrofit2.http.Part;

import retrofit2.http.Query;

public interface IApi {

//GET---GET

@GET("/scripts/getUserInfo.php")

Call<List<User>> getUserInfo(@Query("id") int id);

@GET("/scripts/getAds.php")

Call<AdResponse> getAds(@Query("page") int page,

@Query("status\_number") int status\_number,

@Query("id") int id);

@GET("/scripts/getAdsCount.php")

Call<AdsCount> getAdsCount(@Query("status\_number") int status\_number);

@GET("/scripts/getMyAds.php")

Call<AdResponse> getMyAds(@Query("page") int page,

@Query("status\_number") int status\_number,

@Query("id") int id,

@Query("user\_id") int user\_id);

@GET("/scripts/getSearchAds.php")

Call<AdResponse> getSearchAds(@Query("page") int page,

@Query("status\_number") int status\_number,

@Query("where") String where,

@Query("sort\_field") String sort\_field,

@Query("sort\_type") String sort\_type,

@Query("user\_id") int user\_id);

@GET("/scripts/getFavouritesAds.php")

Call<AdResponse> getFavouritesAds(@Query("page") int page,

@Query("id") int id);

@GET("/scripts/acceptRejectAd.php")

Call<ServerResponse> acceptRejectAd(@Query("id") int id,

@Query("status") int status,

@Query("dateTime") String datetime,

@Query("reason\_id") int reason\_id,

@Query("reject\_message") String reject\_message);

@GET("/scripts/getFullAd.php")

Call<FullAdResponse> getFullAd(@Query("id\_ad") int id\_ad,

@Query("user\_id") int user\_id);

@GET("/scripts/authorization.php")

Call<UserResponse> auth(@Query("email\_or\_number") String email\_or\_number,

@Query("flag") String flag,

@Query("password") String password);

@GET("/scripts/getMyAdsCount.php")

Call<AdsCount> getMyAdsCount(@Query("status\_number") int status,

@Query("id") int id);

@GET("/scripts/getSearchAdsCount.php")

Call<AdsCount> getSearchAdsCount(@Query("status\_number") int status,

@Query("where") String where);

@GET("/scripts/getCategoriesWithSubcategories.php")

Call<CategoryResponse> getCategoriesWithSubcategories();

@GET("/scripts/getRejectReasons.php")

Call<ReasonsResponse> getRejectReasons();

//POST---POST

@FormUrlEncoded

@POST("/scripts/deleteUserImage.php")

Call<ServerResponse> deleteAccountImage(@Field("id") int id,

@Field("filename") String filename);

@FormUrlEncoded

@POST("/scripts/setDeleteBookmark.php")

Call<ServerResponse> setDeleteBookmark(@Field("ad\_id") int ad\_id,

@Field("user\_id") int user\_id,

@Field("datetime") String datetime,

@Field("flag") String flag);

@FormUrlEncoded

@POST("/scripts/setRegion.php")

Call<ServerResponse> setRegion(@Field("region") String region,

@Field("town") String town,

@Field("email") String email);

@FormUrlEncoded

@POST("/scripts/setPhoneNumber.php")

Call<ServerResponse> setPhoneNumber(@Field("phoneNumber") String phoneNumber,

@Field("email") String email);

@FormUrlEncoded

@POST("/scripts/showHidePhone.php")

Call<ServerResponse> showHidePhone(@Field("isPhoneHide") int isPhoneHide,

@Field("email") String email);

@FormUrlEncoded

@POST("/scripts/registration.php")

Call<RegisterResponse> register(@Field("nickname") String nickname,

@Field("email") String email,

@Field("phoneNumber") String phoneNumber,

@Field("region") String region,

@Field("town") String town,

@Field("password") String password);

//MULTIPART---MULTIPART

@Multipart

@POST("/scripts/uploadAccountImage.php")

Call<String> uploadAccountImage(@Part MultipartBody.Part file,

@Part("email") RequestBody email);

@Multipart

@POST("/scripts/insertAd.php")

Call<ServerResponse> insertAd(@Part("title") RequestBody title,

@Part("description") RequestBody description,

@Part MultipartBody.Part image1,

@Part MultipartBody.Part image2,

@Part MultipartBody.Part image3,

@Part MultipartBody.Part image4,

@Part MultipartBody.Part image5,

@Part("price") RequestBody price,

@Part("nickname") RequestBody nickname,

@Part("dateTime") RequestBody datetime,

@Part("subcategory") RequestBody subcategory);

}

package ragalik.baraxolka.paging\_feed

import com.google.gson.annotations.SerializedName

class Ad {

@field:SerializedName("id")

var id: Int? = null

@field:SerializedName("title")

var title: String? = null

@field:SerializedName("nickname")

var userNickname: String? = null

@field:SerializedName("userImage")

var userImage: String? = null

@field:SerializedName("dateTime")

var dateTime: String? = null

@field:SerializedName("description")

var description: String? = null

@field:SerializedName("image1")

var image1url: String? = null

@field:SerializedName("image2")

var image2url: String? = null

@field:SerializedName("image3")

var image3url: String? = null

@field:SerializedName("image4")

var image4url: String? = null

@field:SerializedName("image5")

var image5url: String? = null

@field:SerializedName("price")

var price: Int? = null

@field:SerializedName("views")

var views: Int? = null

@field:SerializedName("region")

var region: String? = null

@field:SerializedName("town")

var town: String? = null

@field:SerializedName("subcategory\_name")

var subcategoryName: String? = null

@field:SerializedName("category\_name")

var categoryName: String? = null

@field:SerializedName("id\_user")

var idUser: String? = null

@field:SerializedName("favourite")

var isFavourite: Boolean? = null

@field:SerializedName("reason\_name")

var reasonName: String? = null

@field:SerializedName("reject\_message")

var rejectMessage: String? = null

}

class AdResponse {

@field:SerializedName("ads")

var ads: List<Ad>? = null

}

class FullAdResponse {

@field:SerializedName("ad")

var ad: List<Ad>? = null

}

package ragalik.baraxolka.paging\_feed

import android.annotation.SuppressLint

import android.content.Intent

import android.text.SpannableStringBuilder

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import android.widget.AdapterView

import android.widget.ArrayAdapter

import android.widget.Toast

import androidx.core.text.bold

import androidx.core.view.isVisible

import androidx.paging.PagedListAdapter

import androidx.recyclerview.widget.DiffUtil

import androidx.recyclerview.widget.RecyclerView

import com.squareup.picasso.Picasso

import kotlinx.android.synthetic.main.item\_ad.view.\*

import ragalik.baraxolka.R

import ragalik.baraxolka.other\_logic.full\_ad.FullAdActivity

import ragalik.baraxolka.MainActivity

import ragalik.baraxolka.network.ApiClient

import ragalik.baraxolka.network.entities.ReasonsResponse

import ragalik.baraxolka.network.entities.ServerResponse

import ragalik.baraxolka.paging\_feed.favourites.SetDeleteBookmark

import ragalik.baraxolka.paging\_feed.moderator.AdModerator

import retrofit2.Call

import retrofit2.Callback

import retrofit2.Response

import java.text.SimpleDateFormat

import java.util.\*

import kotlin.collections.ArrayList

class AdAdapter(private val flag: String = "") : PagedListAdapter <Ad, AdAdapter.AdViewHolder>(AD\_COMPARATOR){

companion object {

var activity = MainActivity()

var reasonPosition = 1

private val AD\_COMPARATOR = object : DiffUtil.ItemCallback<Ad>() {

override fun areItemsTheSame(oldItem: Ad, newItem: Ad): Boolean =

oldItem.dateTime == newItem.dateTime

@SuppressLint("DiffUtilEquals")

override fun areContentsTheSame(oldItem: Ad, newItem: Ad): Boolean =

newItem == oldItem

}

}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): AdViewHolder {

val view = LayoutInflater.from(parent.context)

.inflate(R.layout.item\_ad, parent, false)

return AdViewHolder(view)

}

override fun onBindViewHolder(holder: AdViewHolder, position: Int) {

val ad = getItem(position)

val view = holder.itemView

val id = ad?.id

view.setOnClickListener {

val myIntent = Intent(MainActivity.activity, FullAdActivity::class.java)

myIntent.putExtra("adId", ad?.id)

MainActivity.activity.startActivity(myIntent)

}

view.bookmark\_button\_ads.setOnClickListener {

if (view.bookmark\_button\_ads.isChecked) {

SetDeleteBookmark.getInstance().setDeleteBookmark(id!!, it.bookmark\_button\_ads, "set", MainActivity.activity, ad)

} else {

SetDeleteBookmark.getInstance().setDeleteBookmark(id!!, it.bookmark\_button\_ads, "delete", MainActivity.activity, ad)

}

}

view.bookmark\_button\_ads.isVisible = MainActivity.sp.getInt("id", 0) != 0

if (flag == "MODERATOR") {

var rejectFlag = true

view.moderator\_layout.visibility = View.VISIBLE

view.moderator\_accept\_button.setOnClickListener {

acceptRejectAd(id!!, true)

}

view.moderator\_reject\_button.setOnClickListener {

if (rejectFlag) {

view.rejectReasonLayout.visibility = View.VISIBLE

getReasons(view)

} else {

view.rejectReasonLayout.visibility = View.GONE

}

rejectFlag = !rejectFlag

}

view.acceptRejectButton.setOnClickListener {

if (reasonPosition == 1) {

Toast.makeText(it.context, "Укажите причину", Toast.LENGTH\_LONG).show()

} else {

val rejectMessage = view.rejectMessageEditText.editText?.text.toString()

acceptRejectAd(id!!, false, rejectMessage)

view.rejectReasonLayout.visibility = View.GONE

view.rejectMessageEditText.editText?.setText("")

view.rejectReasonDropdown.editText?.setText("")

}

}

} else if (flag == "REJECTED") {

view.rejected\_ads\_layout.visibility = View.VISIBLE

view.ad\_edit\_but.visibility = View.VISIBLE

view.ad\_delete\_but.visibility = View.VISIBLE

view.ad\_edit\_but.setOnClickListener {

}

view.ad\_delete\_but.setOnClickListener {

}

val customRejectReason = SpannableStringBuilder()

.bold { append("Причина отклонения: ") }

.append("${ad?.reasonName}")

view.tw\_reject\_reason.text = customRejectReason

if (ad?.rejectMessage.isNullOrEmpty()) {

view.tw\_reject\_message.visibility = View.GONE

} else {

val customRejectMes = SpannableStringBuilder()

.bold { append("Комментарий к отклонению: ") }

.append("${ad?.rejectMessage}")

view.tw\_reject\_message.text = customRejectMes

}

} else if (flag == "ACTIVE") {

view.ad\_edit\_but.visibility = View.VISIBLE

view.ad\_deactivate\_but.visibility = View.VISIBLE

view.ad\_edit\_but.setOnClickListener{

}

view.ad\_deactivate\_but.setOnClickListener {

}

} else if (flag == "NON\_ACTIVE") {

view.ad\_delete\_but.visibility = View.VISIBLE

view.ad\_delete\_but.setOnClickListener {

}

}

ad?.let { holder.bind(ad) }

}

class AdViewHolder (view: View) : RecyclerView.ViewHolder(view) {

private val imageView = view.iv\_avatar

private val title = view.tv\_title

private val location = view.tv\_location

private val datetime = view.ad\_preview\_date\_time

private val isFavouritesButton = view.bookmark\_button\_ads

private val price = view.ad\_preview\_price

fun bind (ad: Ad) {

title.text = ad.title

when (ad.region) {

"Минск" -> {

location.text = "${ad.town} р-н"

}

"0" -> {

location.text = "Регион не указан"

}

else -> {

location.text = ad.town

}

}

datetime.text = ad.dateTime

isFavouritesButton.isChecked = ad.isFavourite!!

price.text = "${ad.price.toString()} Руб."

// Glide.with(imageView.context)

// .load(ad.image1url)

// .placeholder(R.drawable.jiga\_olegovicha)

// .into(imageView)

Picasso.get()

.load(ad.image1url)

.fit()

.centerCrop()

.into(imageView)

}

}

private fun acceptRejectAd (id: Int, accepted: Boolean, rejectMessage: String = "") {

val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.forLanguageTag("ru"))

dateFormat.timeZone = TimeZone.getTimeZone("GMT+3")

val datetime = dateFormat.format(Date())

val call = ApiClient.getApi().acceptRejectAd(id, if(accepted) 1 else 2, datetime, reasonPosition, rejectMessage)

call.enqueue(object : Callback<ServerResponse> {

override fun onResponse(call: Call<ServerResponse>, response: Response<ServerResponse>) {

if (response.body() != null) {

AdModerator.itemViewModel.liveDataSource.value!!.invalidate()

}

}

override fun onFailure(call: Call<ServerResponse>, t: Throwable) {}

})

}

private fun getReasons (view: View) {

val call = ApiClient.getApi().rejectReasons

call.enqueue(object : Callback<ReasonsResponse> {

override fun onResponse(call: Call<ReasonsResponse>, response: Response<ReasonsResponse>) {

if (response.body() != null) {

val reasons = response.body()?.reasons!!

val reasonsArr : ArrayList<String> = ArrayList()

for (i in reasons) {

i.reason\_name?.let { reasonsArr.add(it) }

}

view.rejectReasonDropdownText.setAdapter(ArrayAdapter(MainActivity.activity, R.layout.dropdown\_text\_color, reasonsArr))

view.rejectReasonDropdownText.setOnClickListener { view.rejectReasonDropdownText.showDropDown() }

view.rejectReasonDropdownText.onItemClickListener = AdapterView.OnItemClickListener { \_, \_, position, \_ -> reasonPosition = position + 2 }

}

}

override fun onFailure(call: Call<ReasonsResponse>, t: Throwable) {}

})

}

}

package ragalik.baraxolka.paging\_feed.favourites;

import android.content.Context;

import android.widget.CheckBox;

import android.widget.Toast;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Locale;

import java.util.TimeZone;

import ragalik.baraxolka.MainActivity;

import ragalik.baraxolka.network.ApiClient;

import ragalik.baraxolka.network.entities.ServerResponse;

import ragalik.baraxolka.paging\_feed.Ad;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class SetDeleteBookmark {

private static SetDeleteBookmark instance;

public static SetDeleteBookmark getInstance () {

if (instance == null) {

instance = new SetDeleteBookmark();

}

return instance;

}

public void setDeleteBookmark (int ad\_id, CheckBox bookmark, String flag, Context context, Ad ad) {

int user\_id = MainActivity.sp.getInt("id", 0);

Date date = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.forLanguageTag("ru"));

formatter.setTimeZone(TimeZone.getTimeZone("GMT+3"));

String datetime = formatter.format(date);

Call<ServerResponse> call = ApiClient.getApi().setDeleteBookmark(ad\_id, user\_id, datetime, flag);

call.enqueue(new Callback<ServerResponse>() {

@Override

public void onResponse(Call<ServerResponse> call, Response<ServerResponse> response) {

if (response.body() != null && response.body().getSuccess() == 1) {

if (flag.equals("set")) {

Toast.makeText(context, "Объявление добавлено в закладки", Toast.LENGTH\_SHORT).show();

bookmark.setChecked(true);

ad.setFavourite(true);

} else if (flag.equals("delete")) {

Toast.makeText(context, "Объявление убрано из закладок", Toast.LENGTH\_SHORT).show();

bookmark.setChecked(false);

ad.setFavourite(false);

}

}

}

@Override

public void onFailure(Call<ServerResponse> call, Throwable t) {

Toast.makeText(context, "Произошла ошибка!", Toast.LENGTH\_SHORT).show();

}

});

}

}