

Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
Федерального государственного бюджетного образовательного учреждения высшего
образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет _____ ИТР
Кафедра _____ ПИИ _____

КУРСОВАЯ РАБОТА

По Разработка приложений для мобильных операционных систем
Тема АИС «Доставка кофе»

Руководитель

Колпаков А.А.

(фамилия, инициалы)

(подпись)

(дата)

Студент ПИН - 121

(группа)

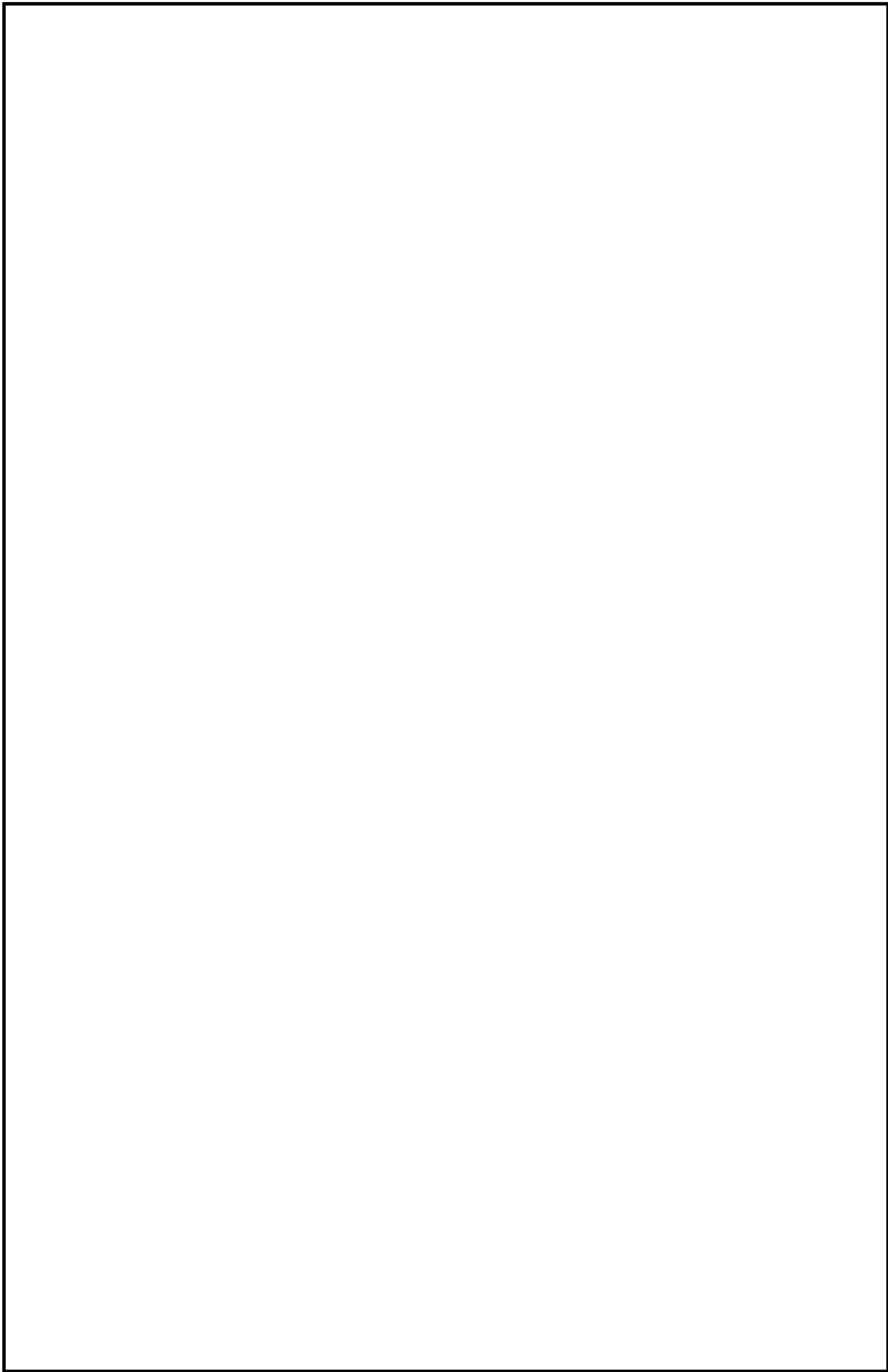
Мочалин Н.А.

(фамилия, инициалы)

(подпись)

(дата)

Муром 2024



В рамках данной курсовой работы требовалось разработать приложение для доставки кофе. Проект реализован на языке Kotlin, в качестве базы данных используется Firebase.

As part of this course work, it was necessary to develop an application for coffee delivery. The project is implemented in Kotlin, and Firebase is used as the database.

Содержание

Введение.....	6
1. Анализ технического задания.....	7
2. Разработка алгоритмов.....	8
3. Руководство программиста.....	13
4. Руководство пользователя.....	15
Заключение.....	16
Список используемой литературы.....	17
Приложение 1.....	18
Приложение 2.....	18



					МИВУ 09.03.04 - 18					
Изм.	Лист	№ докум.	Подпись	Дата	АИС «Доставка кофе»			Лит.	Лист	Листов
Разраб.		Мочалин Н.А.								
Провер.		Колпаков А.А.							3	21
Реценз.								МИ ВлГУ ПИН-121		
Н. Контр.										
Утверд.										

Введение

Автоматизированная информационная система (АИС) «Доставка кофе» представляет собой решение, которое объединяет в себе функции заказа, оплаты и отслеживания статуса доставки кофе. Это приложение может быть полезно как для клиентов, так и для владельцев кофеен и служб доставки.

Актуальность темы курсовой работы обусловлена растущим спросом на услуги доставки кофе, а также необходимостью разработки эффективных и удобных в использовании мобильных приложений для данной сферы бизнеса. АИС «Доставка кофе» может стать важным инструментом для повышения конкурентоспособности кофеен и улучшения качества обслуживания клиентов.

Цель данной работы — разработка мобильного приложения для Android, которое будет включать в себя основные функции АИС «Доставка кофе». Для достижения поставленной цели необходимо решить следующие задачи:

Изучить существующие решения в области доставки кофе и выделить основные требования к функциональности АИС. Разработать концепцию и дизайн мобильного приложения.

Реализовать основные функции АИС с использованием Firebase. Протестировать разработанное приложение и внести необходимые улучшения.

Практическая значимость работы заключается в возможности использования разработанного мобильного приложения в реальных условиях для заказа и доставки кофе. Результаты исследования могут быть полезны как для разработчиков мобильных приложений, так и для владельцев кофеен, желающих улучшить качество обслуживания клиентов.

					МИВУ 09.03.04 – 18.001	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

1. Анализ технического задания.

1. Функционал:

- Просмотр меню
- Добавление в корзину
- Просмотр корзины
- Добавление новых кофе

2. Операционная система и языки программирования:

Приложение будет разработано для платформы Android, что обеспечит доступ к широкой аудитории любителей кофе. Использование языка Kotlin позволит создать современное, безопасное и удобное приложение с лаконичным и читаемым кодом.

3. Документация:

Документация приложения будет создаваться с использованием инструмента Dokka. Это обеспечит структурированный формат документации, облегчающий дальнейшую поддержку и развитие приложения.

4. Система контроля версий:

Исходный код приложения будет размещен в репозитории GitHub. Это позволит централизованно управлять версиями, проводить командную разработку и обеспечивать легкий доступ к истории изменений.

					МИВУ 09.03.04 – 18.001	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

2. Разработка алгоритмов

2.1. Общая структура приложения

Приложение использует архитектуру MVVM (Model-View-ViewModel), которая является расширением архитектуры MVC. MVVM обеспечивает четкое разделение логики приложения, управления данными и пользовательского интерфейса, что упрощает поддержку и масштабирование приложения.

Компоненты архитектуры MVVM:

1. Model (Модель)

Описание:

- Отвечает за работу с данными, включая запросы к базе данных Firebase.
- Модель содержит классы CategoryModel и ItemsModel, которые представляют категории и товары.
- Получение данных из Firebase происходит в MainViewModel через ValueEventListener, который обрабатывает данные из DataSnapshot.

Пример:

```
val list = childSnapshot.getValue(CategoryModel::class.java)
```

Данные загружаются и сохраняются в MutableLiveData, что позволяет обновлять интерфейс автоматически

2. View (Представление)

Описание:

- Представляет пользовательский интерфейс (UI) и отвечает за отображение данных.
- В приложении используются XML-макеты (ActivityMainBinding, ViewholderPopularBinding) и классы Activity, такие как MainActivity и DetailsActivity.
- RecyclerView с адаптерами (PopularAdapter, CategoryAdapter) динамически обновляет списки товаров, акций и категорий.

Пример:

```
binding.recyclerViewPopular.adapter = PopularAdapter(it)
```

3. ViewModel (Модель представления)

Описание:

					МИВУ 09.03.04 – 18.001	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

- Является посредником между Model и View.
- ViewModel получает данные от модели (Firebase) и предоставляет их View через LiveData.
- MainViewModel обрабатывает бизнес-логику и хранит данные категорий, популярных товаров и предложений.

Пример:

```
viewModel.popular.observe(this, Observer {
    binding.recyclerViewPopular.adapter = PopularAdapter(it)
})
```

ViewModel также инициирует загрузку данных через функции loadCategory(), loadPopular(), loadOffers().

Связь компонентов:

1. View (MainActivity) — наблюдает за данными через LiveData и обновляет интерфейс, когда данные изменяются.
2. ViewModel (MainViewModel) — запрашивает данные у Model и сохраняет их в LiveData.
3. Model — получает данные из Firebase и передает их ViewModel.

2.2 Алгоритм отображения информации.

1. Инициализация и запуск приложения

При запуске приложения загружается MainActivity.

В методе onCreate() вызываются функции initCategory(), initPopular(), initOffers(), отвечающие за отображение категорий, популярных товаров и специальных предложений.

2. Запрос данных из Firebase (ViewModel)

MainViewModel выполняет запрос к базе данных Firebase:

loadCategory() — загружает категории кофе.

loadPopular() — получает список популярных товаров.

loadOffers() — получает текущие акции и предложения.

Для получения данных используется FirebaseDatabase и ValueEventListener.

3. Обновление данных через LiveData

Данные, полученные из Firebase, добавляются в MutableLiveData:

					МИВУ 09.03.04 – 18.001	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

```
_category.value = lists
```

Затем LiveData обновляет интерфейс в MainActivity, используя Observer.

4. Обновление интерфейса (RecyclerView)

В initPopular(), initOffers(), initCategory() данные наблюдаются через viewModel:

```
viewModel.popular.observe(this, Observer {  
    binding.recyclerViewPopular.layoutManager =  
        LinearLayoutManager(this@MainActivity, LinearLayoutManager.HORIZONTAL,  
false)  
    binding.recyclerViewPopular.adapter = PopularAdapter(it)  
    binding.progressBarPopular.visibility = View.GONE  
})
```

Адаптер PopularAdapter берет данные и создает элементы интерфейса для RecyclerView.

5. Создание элементов (Адаптер)

PopularAdapter использует макет ViewHolderPopularBinding, чтобы создать карточку с информацией о кофе:

```
holder.binding.titleText.text = items[position].title  
holder.binding.priceText.text = "$" + items[position].price.toString()  
holder.binding.ratingBar.rating = items[position].rating.toFloat()  
Изображение кофе загружается с помощью библиотеки Glide:  
Glide.with(holder.itemView.context)  
    .load(items[position].picUrl[0])  
    .into(holder.binding.pic)
```

6. Отображение деталей товара (DetailsActivity)

При клике на товар происходит переход в DetailsActivity, где отображается подробная информация о выбранном кофе:

```
val intent = Intent(holder.itemView.context, DetailsActivity::class.java)  
intent.putExtra("object", items[position])  
holder.itemView.context.startActivity(intent)
```

2.3 Алгоритм работы корзины.

1. Инициализация корзины

					МИВУ 09.03.04 – 18.001	Лист
Изм.	Лист	№ докум.	Подпись	Дата		10

- Создается экземпляр ManagementCart, который принимает Context приложения для работы с локальным хранилищем (TinyDB).
- Корзина представлена в виде списка объектов ItemsModel, который сохраняется в TinyDB.

2. Добавление товара в корзину (insertItems)

1. Получить текущий список товаров из корзины (getListCart).
2. Проверить, есть ли уже товар с таким же названием (title).
3. Если товар есть, обновить его количество (numberInCart).
4. Если товара нет в корзине, добавить новый товар в список.
5. Сохранить обновленный список в TinyDB.
6. Показать уведомление (Toast) о добавлении товара.

Ключевые моменты:

- Проверка осуществляется с помощью: `val existAlready = listItem.any { it.title == item.title }`

- Сохранение обновленного списка:

`tinyDB.putListObject("CartList", listItem)`

3. Получение списка товаров в корзине (getListCart)

1. Получить список товаров из TinyDB.
2. Если список пустой или отсутствует, вернуть пустой ArrayList: `return tinyDB.getListObject("CartList") ?: arrayListOf()`

4. Уменьшение количества товара (minusItem)

1. Проверить количество единиц товара в позиции position.
2. Если товар в количестве 1, удалить его из корзины.
3. Если количество больше 1, уменьшить на единицу.
4. Сохранить обновленный список в TinyDB.
5. Вызвать onChanged() у слушателя, чтобы обновить интерфейс.

5. Увеличение количества товара (plusItem)

1. Увеличить количество единиц товара на 1 для позиции position.
2. Сохранить обновленный список в TinyDB.
3. Вызвать onChanged() для обновления UI.

6. Подсчет общей стоимости корзины (getTotalFee)

					МИВУ 09.03.04 – 18.001	Лист 11
Изм.	Лист	№ докум.	Подпись	Дата		

1. Получить список всех товаров из корзины.
 2. Пройтись по каждому товару и умножить цену на количество (price * numberInCart).
 3. Суммировать все стоимости.
 4. Вернуть итоговую сумму как Double.
- fee += item.price * item.numberInCart

					МИВУ 09.03.04 – 18.001	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

3. Руководство программиста

Приложение имеет несколько слоёв:

1. Презентационный слой (UI Layer):

Описание:

Этот слой отвечает за отображение данных и взаимодействие с пользователем. Он содержит экраны, элементы пользовательского интерфейса и адаптеры для списков.

- Основная задача – получение данных и их отображение на экране.
- Обрабатывает пользовательские действия и передает их на следующий слой.

Пример классов:

- MainActivity – главный экран приложения.
- PopularAdapter – адаптер для отображения списка популярных товаров.

Взаимодействие:

- MainActivity получает данные из MainViewModel и обновляет интерфейс.
- PopularAdapter отвечает за отображение списка товаров на главном экране.

2. Слой ViewModel (ViewModel Layer):

Описание:

Служит посредником между пользовательским интерфейсом и бизнес-логикой. Содержит данные и управляет их состоянием, обеспечивая их сохранение при изменении конфигурации.

Пример классов:

- MainViewModel – хранит данные для MainActivity и управляет их обновлением.

Взаимодействие:

- MainViewModel получает список товаров из бизнес-логики (ManagmentCart).
- При изменении данных MainViewModel уведомляет MainActivity для обновления интерфейса.

3. Модельный слой (Model Layer):

Описание:

					МИВУ 09.03.04 – 18.001	Лист
Изм.	Лист	№ докум.	Подпись	Дата		13

Определяет структуру данных, используемых в приложении. Этот слой описывает, как должны выглядеть объекты (например, товары или элементы меню).

Пример классов:

- ItemsModel – модель данных для товара (название, цена, изображение).

Взаимодействие:

- ItemsModel передается между слоями MainViewModel и PopularAdapter для отображения данных.

4. Бизнес-логика и управление данными (Domain Layer):

Описание:

Содержит основную бизнес-логику приложения. Отвечает за обработку данных, управление корзиной и выполнение операций.

Пример классов:

- ManagmentCart – класс для работы с корзиной (добавление и удаление товаров, подсчет итоговой суммы).

Взаимодействие:

- ManagmentCart получает команды от MainViewModel и обновляет данные корзины.
- Данные передаются в MainActivity для отображения изменений.

5. Слой данных (Data Layer):

Описание:

Этот слой отвечает за работу с данными: их хранение, загрузку и обновление.

Пример классов:

- TinyDB – класс для локального хранения данных корзины.

Взаимодействие:

- ManagmentCart сохраняет данные корзины в TinyDB.
- При запуске приложения корзина загружается из TinyDB в MainViewModel.

4. Руководство пользователя

Вступительная страница (IntroActivity)

Это страница создаёт первое впечатление о приложении и на ней нет ничего кроме кнопки продолжить

Главная страница (MainActivity)

На этой страницы представлены категории, популярные товары и акции. По товарам можно кликнуть для отображения подробной информации. Также есть кнопка для перехода в корзину, где отображены товары добавленные в корзину

Подробная информация (DetailsActivity)

Страница с подробной информацией содержит описание товара, а также предоставляет возможность добавлять товар в корзину.

Корзина (CartActivity)

В корзине можно увеличивать и уменьшать количество уже добавленных товаров. Вводить промокод и оформить заказ.

					МИВУ 09.03.04 – 18.001	Лист
Изм.	Лист	№ докум.	Подпись	Дата		15

Заключение

В ходе выполнения курсовой работы было разработано мобильное приложение, предназначенное для упрощения и автоматизации процесса заказа кофе и напитков. Приложение охватывает основные потребности пользователей, позволяя быстро выбирать товары, добавлять их в корзину и оформлять заказ в несколько кликов.

Проект был реализован с применением современных технологий разработки Android-приложений, включая архитектурный паттерн MVVM, что обеспечило четкое разделение логики приложения на слои и повысило читаемость и масштабируемость кода. Для хранения данных и управления корзиной использовалась библиотека TinyDB, что позволило оптимизировать локальное сохранение информации о заказах. В качестве основной базы данных для хранения информации о кофе использовалась Firebase Realtime Database, что обеспечило синхронизацию данных в реальном времени.

Основные результаты работы включают:

- Создание интуитивно понятного и простого в использовании интерфейса.
- Реализацию ключевых функций: отображение списка товаров, управление корзиной, оформление заказа.
- Оптимизацию производительности и стабильности приложения.

В процессе разработки были приобретены навыки работы с Kotlin и адаптерами RecyclerView. Особое внимание было уделено пользовательскому опыту и быстродействию приложения.

Перспективы развития включают добавление функции онлайн-оплаты, интеграцию с картами для отслеживания доставки и внедрение системы лояльности для постоянных клиентов.

Разработка данного приложения стала ценным опытом для освоения навыков мобильной разработки и позволила получить практическое представление о процессе создания коммерческих приложений.

					МИВУ 09.03.04 – 18.001	Лист
Изм.	Лист	№ докум.	Подпись	Дата		16

Список литературы:

1. Введение в разработку приложений для смартфонов на ОС Android / А.Семакова – М.: Национальный открытый Университет «ИНТУИТ», 2016
2. Колисниченко Д.Н. Программирование для Android 5. Самоучитель. — СПб.: БХВПетербург, 2015. — 303 с.
3. Дейтел П., Дейтел Х., Уолд А. Android для разработчиков. 3-е изд. — СПб.: Питер, 2016.
4. Гриффитс Дэвид, Гриффитс Дон Head First. Программирование для Android. 2-е изд. — СПб.: Питер, 2018. — 912 с

					МИВУ 09.03.04 – 18.001	Лист
						17
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение

Приложение 1. Программный код продукта

<https://github.com/B4N4N41C/CoffeeGoAndroid> - ссылка на репозиторий с исходным кодом

Приложение 2. Снимки окон программы

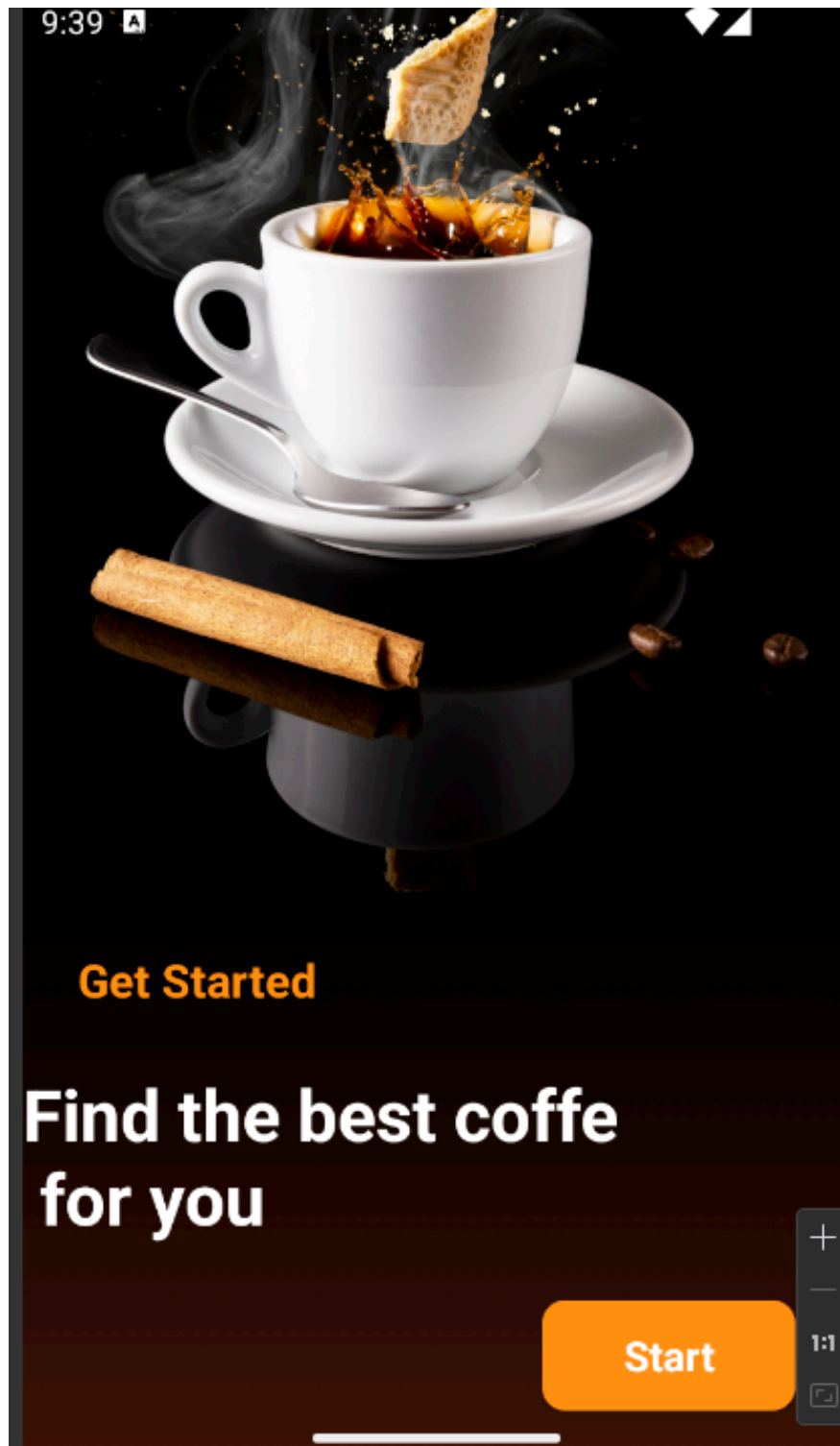


Рисунок 1 - стартовый экран

					МИВУ 09.03.04 – 18.001	Лист
						18
Изм.	Лист	№ докум.	Подпись	Дата		

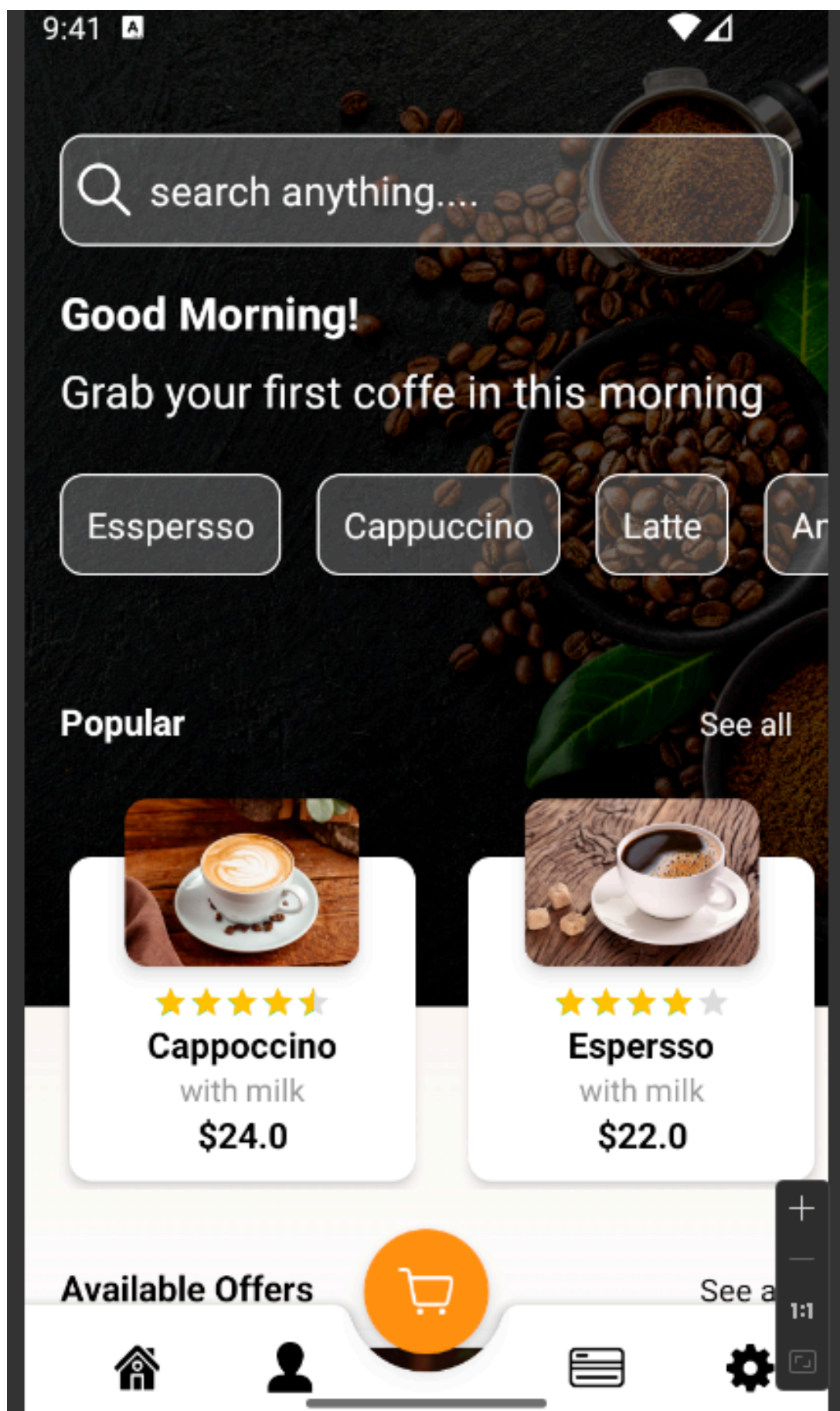


Рисунок 2 - главный экран



Рисунок 3 - экран с подробной информацией

					МИВУ 09.03.04 – 18.001	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

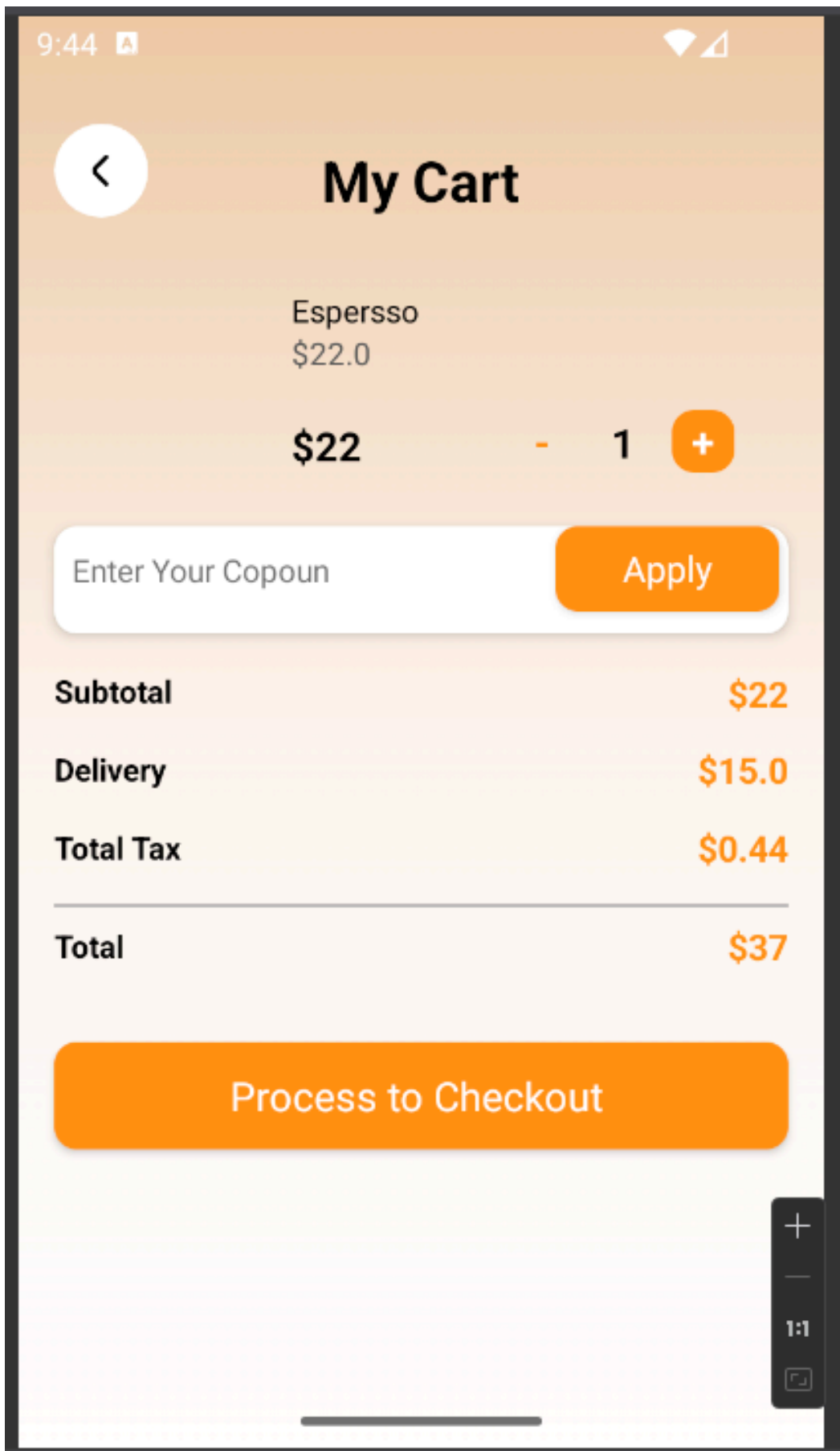


Рисунок 4 - экран корзины