

## ВВЕДЕНИЕ

В современных экономических условиях управление личными финансами и повышение финансовой грамотности населения приобретают первостепенное значение. Одним из ключевых макроэкономических показателей, напрямую влияющих на благосостояние каждого гражданина, является инфляция. В Российской Федерации официальный расчет индекса потребительских цен (ИПЦ), осуществляемый Федеральной службой государственной статистики (Росстат), служит основным ориентиром для оценки инфляционных процессов в стране.

Однако для многих граждан официальные данные об инфляции не всегда совпадают с личным восприятием роста цен. Этот феномен подтверждается исследованиями Центрального Банка РФ, согласно которым субъективная оценка инфляции населением, как правило, в 1,5–2 раза превышает официальные данные. Так, по данным на декабрь 2024 года, граждане оценивали годовой рост цен на уровне 15,9%, в то время как Росстат зафиксировал инфляцию на уровне 9,52%. Эта разница обусловлена тем, что официальный ИПЦ рассчитывается на основе усредненной потребительской корзины, которая может существенно отличаться от структуры реальных расходов конкретного человека или семьи. Такие факторы, как индивидуальные потребительские привычки, фокус внимания на ценах товаров-маркеров (продукты питания, бензин, услуги ЖКХ) и изменение ассортимента покупок, приводят к формированию уникального, **личного уровня инфляции**.

На фоне этого наблюдается устойчивый рост финансовой культуры россиян. По данным Аналитического центра НАФИ на 2024 год, более 70% взрослого населения России имеют средний или высокий уровень финансовой грамотности. Растет и спрос на цифровые инструменты для управления финансами: почти 75% россиян активно пользуются мобильными

банковскими приложениями, а среди ключевых ожиданий от финтех-сервисов 39% пользователей выделяют функцию контроля доходов и расходов.

Несмотря на это, на российском рынке мобильных приложений практически отсутствуют специализированные решения, сфокусированные на расчете и анализе персональной инфляции. Таким образом, возникает **проблема**: при наличии высокого запроса на качественное управление личным бюджетом и явном расхождении между официальной и ощущаемой инфляцией, у граждан отсутствует удобный инструмент для объективной оценки изменения стоимости их собственной жизни. Решением данной проблемы может стать создание специализированного мобильного приложения, сфокусированного на решении именно этой задачи.

**Целью данной выпускной квалификационной работы** является разработка Android-приложения для мониторинга и анализа личной инфляции, предоставляющего пользователям удобный инструмент для объективной оценки изменения стоимости их жизни на основе реальных данных о покупках.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Провести анализ предметной области, включая изучение существующих на российском рынке мобильных приложений для управления финансами, а также методов расчета потребительской инфляции и возможностей их адаптации для анализа личных данных.
2. Исследовать современные технологические стеки, языки программирования и архитектурные подходы, применяемые при разработке мобильных приложений под платформу Android.
3. На основе проведенного анализа сформулировать и обосновать функциональные и нефункциональные требования к разрабатываемому приложению.

4. Спроектировать архитектуру приложения, структуру базы данных для хранения информации о покупках и пользовательский интерфейс (UI/UX).
5. Реализовать основные программные модули системы на языке Kotlin в соответствии со спроектированной архитектурой, включая модуль ввода данных, модуль расчета инфляции и модуль визуализации результатов.
6. Провести тестирование разработанного приложения для проверки его функциональности, производительности и удобства использования.

**Объектом исследования** является процесс отслеживания и анализа персональных финансовых данных с помощью мобильного приложения.

**Предметом исследования** выступают методы расчета, визуализации и анализа личной инфляции в рамках Android-приложения.

Результатом работы станет готовое Android-приложение, позволяющее пользователю рассчитывать свой персональный индекс инфляции и наглядно отслеживать его динамику, что будет способствовать более осознанному управлению личным бюджетом.

# ГЛАВА 1. АНАЛИЗ МЕТОДОВ И СРЕДСТВ РАЗРАБОТКИ

## ПРИЛОЖЕНИЯ ДЛЯ МОНИТОРИНГА ЛИЧНОЙ ИНФЛЯЦИИ

### 1.1. АНАЛИЗ СУЩЕСТВУЮЩИХ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

#### ДЛЯ УЧЕТА ЛИЧНЫХ ФИНАНСОВ

Целью данного подраздела является исследование популярных на российском рынке мобильных приложений для управления личными финансами. Анализ направлен на выявление их ключевых функциональных возможностей, преимуществ и недостатков, а также на оценку их пригодности для решения задачи мониторинга и анализа персональной инфляции. Для исследования были выбраны приложения, занимающие высокие позиции в категории "Финансы" в магазине Google Play: «Дзен-мани», «Быстрый Бюджет», «Cashew» и «Monefy».

#### 1.1.1. ОБЗОР АНАЛИЗИРУЕМЫХ ПРИЛОЖЕНИЙ

«Дзен-мани» — одно из наиболее функциональных приложений на российском рынке, ключевой особенностью которого является автоматизация учета. Система интегрируется с крупнейшими российскими банками (Сбербанк, Тинькофф, Альфа-Банк и др.), автоматически загружая транзакции и распределяя их по категориям. Приложение предлагает мощные инструменты для бюджетирования, анализа трат по категориям и ведения семейного бюджета. Однако его фокус — это **транзакционный учет**. Система оперирует итоговыми суммами операций, но не предполагает детализации чека до уровня отдельных товарных позиций (SKU), их количества и цены за единицу, что делает невозможным отслеживание динамики цен на конкретные товары.

«Быстрый Бюджет» — приложение, ориентированное на простоту и скорость ручного ввода данных. Его основная задача — помочь пользователю контролировать расходы и не выходить за рамки установленных бюджетов по категориям. Интерфейс интуитивно понятен, а отчеты в виде диаграмм

наглядно показывают структуру трат. Как и «Дзен-мани», приложение работает на уровне транзакций («Продукты — 1500 руб.»), а не отдельных товаров. Инструментарий для отслеживания изменения цены на конкретный товар во времени отсутствует.

«Cashew» — позиционируется как минималистичный и удобный инструмент для совместного ведения бюджета. Приложение позволяет быстро фиксировать доходы и расходы, отслеживать долги и формировать общие бюджеты. Упор сделан на простоту интерфейса и социальное взаимодействие (совместный учет). Детализация расходов до отдельных товаров в чеке не является основной функцией, и, как следствие, в приложении нет механизмов для анализа ценовой динамики.

«Monefy» — популярное приложение, известное своим наглядным интерфейсом на главном экране в виде круговой диаграммы. Оно предназначено для максимально быстрого ручного добавления расходов. Пользователь выбирает категорию и вводит сумму, что занимает буквально несколько секунд. Приложение идеально подходит для общего понимания структуры трат, но не обладает необходимой для расчета инфляции гранулярностью данных.

**1.1.2. СРАВНИТЕЛЬНЫЙ АНАЛИЗ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ**

Для наглядного сопоставления ключевых характеристик приложений сведем их в сравнительную таблицу (Таблица 1.1).

Таблица 1.1.

Сравнение общих функциональных возможностей приложений

Критерий	Дзен-мани	Быстрый Бюджет	Cashew	Monefy
Основной метод ввода	Автоматический (синхронизация с банками)	Ручной	Ручной	Ручной

<b>Бюджетирование</b>	Да, продвинутое	Да, по категориям	Да, совместное	Да, базовое
<b>Отчетность</b>	Продвинутая (графики, диаграммы)	Базовая (диаграмма)	Базовая (диаграмма)	Наглядная (графики)
<b>Сканирование чеков</b>	Да (распознавание QR)	Нет	Нет	Нет
<b>Модель монетизации</b>	Платная подписка	Бесплатно (с Pro-версией)	Платная подписка	Бесплатно (с Pro-версией)

### 1.1.3. АНАЛИЗ ПРИГОДНОСТИ ДЛЯ РАСЧЕТА ЛИЧНОЙ ИНФЛЯЦИИ

Несмотря на различия в функционале, для цели данной работы ключевым является анализ специализированных возможностей, необходимых для расчета персонального индекса цен. Результаты этого анализа представлены в Таблице 1.2.

Таблица 1.2.

Сравнение приложений на предмет пригодности для анализа инфляции

<b>Критерий</b>	<b>Дзен-мани</b>	<b>Быстрый Бюджет</b>	<b>Cashew</b>	<b>Monefy</b>	<b>Разрабатываемое приложение</b>
<b>Учет на уровне товаров (SKU)</b>	Нет	Нет	Нет	Нет	Да (ключевая функция)
<b>Отслеживание цены за единицу товара</b>	Нет	Нет	Нет	Нет	Да (ключевая функция)
<b>Механизм сравнения</b>	Отсутствует	Отсутствует	Отсутствует	Отсутствует	Да (адаптированы)

цен во времени					й индекс Ласпейреса)
Встроенный модуль расчета инфляции	Отсутствует	Отсутствует	Отсутствует	Отсутствует	Да (основная цель)

**Вывод по результатам анализа:** Проведенное исследование позволяет сделать однозначный вывод: существующие на российском рынке популярные приложения для управления финансами ориентированы на **учет транзакций и управление бюджетом, а не на анализ цен**. Они эффективно отвечают на вопрос «На что я трачу деньги?», но не предоставляют инструментария для ответа на вопрос «Насколько для меня выросли цены?». Отсутствие детализации данных до уровня отдельных товарных позиций и их цен делает невозможным применение методологии ценовых индексов.

Это подтверждает наличие незанятой ниши и высокую практическую значимость разработки специализированного мобильного приложения, основной функцией которого станет именно расчет, анализ и визуализация персональной инфляции.

### **1.2.1 АНАЛИЗ МЕТОДОВ И АЛГОРИТМОВ РАСЧЕТА**

#### **ПОТРЕБИТЕЛЬСКОЙ ИНФЛЯЦИИ И ИХ АДАПТАЦИЯ ДЛЯ ЦЕЛЕЙ ПЕРСОНАЛЬНОГО ФИНАНСОВОГО АНАЛИЗА**

Для объективной оценки изменения стоимости жизни используются индексы цен, измеряющие изменение среднего уровня цен на фиксированный набор товаров и услуг — потребительскую корзину. Ключевая задача при разработке приложения для анализа личной инфляции — выбрать и адаптировать такой метод расчета, который был бы одновременно научно корректным и удобным для конечного пользователя.

### 1.2.2. ИНДЕКС ЛАСПЕЙРЕСА И УЧЕТ ВЕСА ТОВАРОВ В ПОТРЕБИТЕЛЬСКОЙ КОРЗИНЕ

Основным методом, выбранным для реализации в приложении, является **индекс цен Ласпейреса**. Его фундаментальное преимущество — использование для сравнения структуры потребления (набора и количества товаров) в **базовом периоде**.

Формула индекса Ласпейреса:

$$I_L = \frac{\sum(P_t \cdot Q_0)}{\sum(P_0 \cdot Q_0)} \cdot 100\%$$

- $P_t$ : Цена товара в текущем периоде.
- $P_0$ : Цена товара в базисном периоде.
- $Q_0$ : Объём продаж товара в базисном периоде.

Важнейшей особенностью данной формулы является **автоматический учет веса каждого товара в корзине**. Вес товара определяется его долей в общих расходах базового периода. Товар, на который пользователь потратил значительную часть своего бюджета (например, бензин или аренда жилья), будет иметь высокий вес. И наоборот, товар с незначительными затратами (например, соль или спички) будет иметь низкий вес.

Математически это заложено в самой структуре формулы. Изменение цены на товар с высокой исходной стоимостью затрат внесет гораздо больший вклад в итоговое значение индекса, чем аналогичное процентное изменение цены на "дешевый" товар. Таким образом, требование об учете веса товара удовлетворяется самой математической моделью индекса Ласпейреса без необходимости введения дополнительных сложных коэффициентов. Приложение, реализующее этот алгоритм, будет корректно отражать инфляционное давление: рост цен на ключевые статьи расходов окажет на личную инфляцию пользователя гораздо большее влияние, чем подорожание второстепенных товаров.

### 1.2.3. СРАВНИТЕЛЬНЫЙ АНАЛИЗ АГРЕГАТНЫХ ИНДЕКСОВ ЦЕН И ИХ ПРИМЕНИМОСТЬ

Помимо индекса Ласпейреса, в экономической теории и статистической практике существуют и другие агрегатные индексы цен, в частности индекс Пааше и индекс Фишера. Для обоснованного выбора методологии для приложения необходимо провести их сравнительный анализ.

- **Индекс Пааше (Paasche Price Index)** в качестве весов (количества товаров  $q$ ) использует данные не базового, а **текущего периода**. Он отвечает на вопрос: «Насколько изменилась стоимость *сегодняшней* корзины по сравнению с тем, сколько бы она стоила в прошлом?». Основной недостаток этого метода — необходимость каждый раз собирать данные не только о ценах, но и о количестве купленных товаров, что полностью противоречит цели создания удобного приложения. Кроме того, поскольку веса меняются в каждом периоде, результаты становятся несопоставимыми в долгосрочной динамике.
- **Индекс Фишера (Fisher Price Index)** считается «идеальным», так как является средним геометрическим из индексов Ласпейреса и Пааше, что нивелирует их разнонаправленные смещения. Однако его расчет требует еще большего объема данных, чем для индекса Пааше, что делает его абсолютно непрактичным для реализации в пользовательском приложении.

Таблица 1.3.

Сравнительный анализ индексов цен

Критерий	Индекс Ласпейреса	Индекс Пааше
База для весов ( $q$ )	Базовый период ( $q_0$ )	Текущий период ( $q_t$ )
Практичность сбора данных	Высокая (нужны только текущие цены)	Низкая (нужны текущие цены и количества)
Основное смещение	Склонен к завышению инфляции (не учитывает эффект замещения)	Склонен к занижению инфляции (учитывает переход на более дешевые товары)

<b>Применимость для приложения</b>	<b>Оптимальная.</b> Минимальные требования к вводу данных.	<b>Неприменим.</b> Высокие требования к вводу данных.
------------------------------------	--	---

**Вывод:** Сравнительный анализ показывает, что для целей мобильного приложения, ориентированного на удобство пользователя, индекс Ласпейреса является единственным методологически корректным и практически реализуемым подходом. Он закладывает стабильный фундамент для отслеживания динамики стоимости жизни, на основе которого можно строить более сложные гибридные модели. Разработка гибридного метода актуализации цен для баланса точности и удобства использования

Ключевой вызов при адаптации индекса Ласпейреса для личного использования — это проблема неполных данных. Пользователь не совершает покупки всех товаров из своей базовой корзины каждый месяц. Это ставит перед разработчиком фундаментальную дилемму, для решения которой можно рассмотреть три основных подхода:

1. Наивный подход (допущение о неизменности цены): Использовать цену базового периода ( $p_0$ ) для всех не купленных товаров. Этот метод максимально прост, но приводит к систематическому искусственному занижению инфляции и лишает приложение его основной ценности — объективности.
2. Подход усеченной корзины: Исключать не купленные товары из расчета. Этот метод еще более некорректен, так как он нарушает главный принцип индексного метода — стабильность корзины, делая результаты разных месяцев абсолютно несопоставимыми.
3. Гибридный интеллектуальный метод: Разработать систему, которая будет автоматически рассчитывать (экстраполировать) наиболее вероятную цену отсутствующих товаров на основе имеющихся у пользователя данных.

Для решения дилеммы между точностью и удобством в рамках данной работы предлагается реализация именно третьего подхода, основанного на

многоуровневой иерархии источников данных и статистической экстраполяции. Он позволяет минимизировать усилия пользователя, сохраняя при этом высокий уровень репрезентативности расчетов.

### **Гибридный интеллектуальный метод:**

Алгоритм определения текущей цены ( $p_t$ ) для каждого товара из базовой корзины строится по следующему приоритету:

Уровень 1: Прямые пользовательские данные (наивысший приоритет)

- Цена из фактической покупки: Если товар был приобретен в отчетном периоде, его цена используется как наиболее достоверный источник.
- Цена из ручного мониторинга: Приложение предоставляет пользователю возможность добровольно актуализировать цену на товар без фиксации факта покупки. Эти данные также считаются абсолютно точными.

Уровень 2: Расчетные данные на основе статистической экстраполяции (автоматический расчет).

Если прямые данные для товара отсутствуют, система не игнорирует его и не использует устаревшие данные базового периода. Вместо этого она экстраполирует наиболее вероятную текущую цену:

- Экстраполяция по категории: Система анализирует все товары той же категории, по которым имеются данные Уровня 1, и вычисляет средний темп инфляции для данной категории за текущий месяц. Этот процент применяется к последней известной цене искомого товара для получения его расчетной текущей цены.
- Экстраполяция по общему индексу: В случае, если в категории нет ни одного товара с обновленной ценой, система в качестве базы для экстраполяции использует общий процент личной инфляции, рассчитанный по всем доступным данным в отчетном месяце.

Ключевым элементом данного подхода является полная прозрачность для пользователя. В интерфейсе приложения рядом с ценой каждого товара будет отображаться индикатор, поясняющий источник данных (например, "из

покупки", "введено вручную" или "рассчитано автоматически на основе инфляции по категории X"). Это не только повышает доверие к системе, но и мотивирует пользователя при желании уточнить расчетные данные, повысив тем самым общую точность своего личного индекса инфляции.

Данный подход является оптимальным решением поставленной дилеммы, так как он:

- Автоматизирует процесс: Снимает с пользователя основную нагрузку по сбору данных.
- Поддерживает точность: Избегает грубых допущений и использует релевантные статистические данные самого пользователя для прогнозирования.
- Вовлекает и расширяет возможности пользователя: Превращает рутинный процесс в интерактивное взаимодействие, где пользователь контролирует точность и понимает логику расчетов.

Таким образом, предложенный гибридный интеллектуальный метод, построенный на базе индекса Ласпейреса, является инновационным решением, адаптированным под специфику мобильных пользовательских приложений. Он сочетает в себе строгую экономическую логику и прагматичный, ориентированный на пользователя подход, что является ключевым фактором для успешной реализации проекта.

### **1.3. АНАЛИЗ СОВРЕМЕННЫХ АРХИТЕКТУРНЫХ ПОДХОДОВ И ТЕХНОЛОГИЧЕСКОГО СТЕКА ДЛЯ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ПОД ПЛАТФОРМУ ANDROID**

Выбор технологического стека и архитектуры является определяющим фактором для успешной разработки, поддержки и дальнейшего развития мобильного приложения. Современная Android-разработка предлагает широкий спектр инструментов и методологий, выбор которых должен основываться на критериях эффективности, надежности, масштабируемости и удобства сопровождения кода. В данном разделе будет проведен анализ

ключевых компонентов разработки для обоснования выбора оптимального набора технологий для проекта.

### 1.3.1. ЯЗЫК ПРОГРАММИРОВАНИЯ: KOTLIN ПРОТИВ JAVA

Исторически основным языком для Android-разработки — Java. Однако с 2019 года компания Google официально объявила **Kotlin** приоритетным языком для платформы Android. На сегодняшний день Kotlin стал отраслевым стандартом для создания новых приложений.

- **Java** является зрелым, надежным языком с огромной экосистемой, однако для него характерна многословность (boilerplate code) и проблемы с безопасностью, в частности, ошибки NullPointerException.
- **Kotlin** предлагает более лаконичный и выразительный синтаксис, что сокращает объем кода и время разработки. Ключевым его преимуществом является встроенная **null-безопасность**, которая устраняет целый класс потенциальных ошибок на этапе компиляции. Кроме того, Kotlin предоставляет современные языковые возможности, такие как корутины (coroutines) для упрощенной асинхронной работы, что критически важно для отзывчивости интерфейса приложения.

**Вывод:** Учитывая официальную поддержку Google, повышенную безопасность, лаконичность кода и современные возможности для асинхронного программирования, **Kotlin** является безальтернативным выбором для разработки нового Android-приложения.

### 1.3.2. ПРИНЦИПЫ ЧИСТОЙ АРХИТЕКТУРЫ (CLEAN ARCHITECTURE) КАК ОСНОВА ПРОЕКТА

Помимо выбора паттерна для слоя представления (Presentation), для создания надежного и масштабируемого приложения необходимо определить общую архитектурную стратегию. В современной мобильной разработке стандартом де-факто является многоуровневый подход, наиболее известной реализацией которого является "**Чистая архитектура**" (Clean Architecture), предложенная Робертом Мартином.

Основная идея этого подхода — строгое **разделение ответственностей (Separation of Concerns)** между слоями приложения. Классическая реализация для Android-приложения включает три основных слоя:

1. **Presentation (Слой представления):** Отвечает за всё, что связано с UI. Включает в себя компоненты UI (экраны, созданные с помощью Jetpack Compose) и ViewModel, которая подготавливает данные для отображения и обрабатывает действия пользователя. Этот слой зависит от Android SDK.
2. **Domain (Слой бизнес-логики):** Ядро приложения. Этот слой содержит всю бизнес-логику, не зависящую от деталей реализации. Здесь находятся бизнес-модели и **"Use Cases"** (или Interactors) — классы, инкапсулирующие конкретные бизнес-правила (например, «Рассчитать личную инфляцию», «Актуализировать цену товара»). Этот слой является чистым Kotlin/Java модулем и **ничего не знает об Android SDK, базе данных или сети.**
3. **Data (Слой данных):** Отвечает за предоставление данных для Domain-слоя. Включает в себя реализации **репозитория (Repositories)**, которые являются единым источником данных для приложения, и конкретные источники данных (**Data Sources**), такие как локальная база данных (Room) или удаленный API.

Ключевым принципом является **"Правило зависимостей" (The Dependency Rule):** все зависимости направлены внутрь, к центральному Domain-слою. Presentation и Data слои зависят от Domain, но Domain не зависит ни от кого.

#### **Обоснование выбора Clean Architecture:**

- **Тестируемость:** Бизнес-логика в Domain-слое может быть покрыта простыми юнит-тестами без необходимости запускать эмулятор или использовать специфичные для Android библиотеки.
- **Независимость от фреймворков:** UI и детали хранения данных могут быть заменены с минимальным влиянием на бизнес-логику.

- **Масштабируемость и поддержка:** Четкое разделение на слои делает код более понятным, упрощает командную работу и добавление нового функционала.

Таким образом, применение принципов Clean Architecture заложит прочный фундамент для создания качественного и поддерживаемого приложения.

### 1.3.3. АНАЛИЗ И ВЫБОР АРХИТЕКТУРНОГО ПАТТЕРНА

#### PRESENTATION-СЛОЯ: MVVM И MVI

В рамках Clean Architecture слой представления (Presentation Layer) также требует выбора конкретного паттерна для организации взаимодействия между UI и логикой. Наиболее актуальными на сегодняшний день являются MVVM и MVI.

- **MVVM (Model-View-ViewModel)** — паттерн, официально рекомендуемый Google и входящий в состав Android Jetpack. Он предполагает, что ViewModel предоставляет один или несколько потоков данных (StateFlow, LiveData), на которые подписывается View (UI). View реагирует на изменения в этих потоках и перерисовывает себя. Данные текут от ViewModel к View, а события от пользователя — от View к ViewModel.
- **MVI (Model-View-Intent)** — более строгий паттерн, построенный на принципах **однонаправленного потока данных (Unidirectional Data Flow - UDF)**.
  - **Intent:** Действия пользователя упаковываются в объекты-намерения и отправляются в ViewModel.
  - **Model:** ViewModel обрабатывает Intent, изменяет на его основе свое состояние и эмитирует новый, единый и неизменяемый **объект состояния (State)**.
  - **View:** View подписывается на единственный поток этих состояний и просто отображает («рендерит») текущее состояние.

Таблица 1.4.

## Сравнение паттернов MVVM и MVI

Критерий	MVVM	MVI
Управление состоянием	Множественные, независимые потоки данных (StateFlow).	Единый, неизменяемый объект состояния (State).
Поток данных	Реактивный, но может быть разнонаправленным (события и данные идут по разным каналам).	Строго однонаправленный и циклический.
Предсказуемость	Высокая, но при большом количестве потоков отслеживание может усложняться.	Максимальная. Состояние экрана всегда является результатом предыдущего состояния и действия.
Сложность и Boilerplate	Меньше шаблонного кода для простых экранов.	Требует больше кода для описания всех состояний и намерений, что может быть избыточно для простых случаев.

Паттерн MVI предоставляет максимальную предсказуемость и упрощает отладку для сложных экранов с множеством асинхронных операций. Однако для приложения по анализу инфляции, где экраны имеют умеренную сложность (списки, формы ввода, графики), его строгость может привести к избыточному усложнению кода.

Паттерн MVVM является "золотой серединой": он обеспечивает отличное разделение логики и UI, прекрасно интегрируется с Jetpack Compose и Kotlin Coroutines, и требует меньше шаблонного кода для реализации функционала данного проекта. Поэтому в качестве архитектурного паттерна для слоя представления **будет использован MVVM**.

## **1.4. РАЗРАБОТКА ТРЕБОВАНИЙ К СИСТЕМЕ**

На основе проведенного анализа предметной области и технологического стека были сформулированы функциональные и нефункциональные требования к разрабатываемому приложению. Требования определяют ключевые возможности системы и критерии ее качества, формируя базис для дальнейшего проектирования и реализации.

### **1.4.1. РОЛИ ПОЛЬЗОВАТЕЛЕЙ**

В системе определена единственная роль — **Пользователь**. Это владелец устройства, который обладает полным набором прав на выполнение всех функций приложения в рамках своего локального профиля данных, включая ввод, редактирование и удаление информации, а также просмотр аналитических отчетов.

### **1.4.2. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ**

#### **FR.1. Управление данными о покупках**

**FR.1.1. Ввод одиночной покупки:** Система должна предоставлять пользователю возможность быстро зафиксировать одну покупку с указанием обязательных атрибутов: "Сумма" и "Наименование товара". Необязательные атрибуты: "Категория", "Дата", "Комментарий".

**FR.1.2. Ввод списка покупок (чека):** Система должна предоставлять интерфейс для удобного последовательного ввода нескольких товаров в рамках одной покупки (чека), с указанием общей информации (название магазина, дата) и данных по каждой позиции (наименование, цена, количество).

**FR.1.3. Просмотр истории покупок:** Система должна отображать все введенные покупки в виде хронологического списка, сгруппированного по датам или чекам, для обеспечения удобной навигации и доступа к деталям.

**FR.1.4. Редактирование и удаление данных:** Пользователь должен иметь возможность редактировать атрибуты любой ранее введенной покупки или полностью удалить запись о покупке (одиночной или чеке) из системы.

## **FR.2. Расчет и анализ инфляции**

### **FR.2.1. Формирование личной потребительской корзины (ЛПК):**

Система должна автоматически формировать ЛПК на основе уникальных товаров, приобретенных пользователем в базовом периоде (предыдущий календарный месяц).

**FR.2.2. Реализация гибридного метода актуализации цен:** Для товаров из ЛПК, не купленных в отчетном периоде, система должна применять гибридный метод расчета их наиболее вероятной текущей цены, как это было обосновано в подразделе 1.2.3.

**FR.2.3. Расчет индекса личной инфляции:** Система должна ежемесячно рассчитывать персональный индекс инфляции за месяц (Month-over-Month, MoM) по адаптированной формуле Ласпейреса.

**FR.2.4. Расчет производных показателей:** При наличии достаточного объема данных система должна рассчитывать инфляцию с начала года (Year-to-Date, YTD) и годовую инфляцию (Year-over-Year, YoY).

## **FR.3. Визуализация данных и отчетность**

**FR.3.1. Отображение ключевых показателей:** Главный экран приложения должен отображать итоговый показатель личной инфляции (MoM и YoY/YTD) в наглядном виде.

**FR.3.2. График динамики инфляции:** Система должна предоставлять линейный график, отображающий динамику месячной инфляции (MoM) пользователя во времени.

**FR.3.3. Отчет "Драйверы инфляции":** Система должна формировать сортированный список товаров и/или категорий, внесших наибольший вклад в изменение общего индекса цен за отчетный период.

**FR.3.4. Диаграмма структуры расходов:** Система должна отображать структуру расходов пользователя за выбранный период в виде кольцевой диаграммы.

## **FR.4. Пользовательская кастомизация и управление**

**FR.4.1. Управление категориями расходов:** Пользователь должен иметь возможность создавать новые, переименовывать и удалять существующие категории расходов.

**FR.4.2. Экспорт данных:** Система должна предоставлять функцию экспорта всех данных о покупках в универсальном формате CSV для целей резервного копирования или анализа во внешних системах.

### **1.4.3. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ**

#### **NFR.1. Производительность:**

- Интерфейс приложения должен оставаться отзывчивым (время отклика на действия пользователя не более 200 мс).
- Процессы расчета инфляции и формирования отчетов должны выполняться в фоновом потоке, не блокируя UI.

#### **NFR.2. Удобство использования (Usability):**

- Интерфейс должен быть интуитивно понятным, следовать гайдлайнам Material Design для платформы Android.
- Процесс ввода одиночной покупки должен занимать минимальное количество действий (не более 3-5 касаний не считая ввода).

#### **NFR.3. Надежность:**

- Все введенные пользователем данные должны корректно и атомарно сохраняться в локальной базе данных.
- Приложение должно корректно обрабатывать ошибки ввода (например, ввод текста в числовое поле).

#### **NFR.4. Хранение данных и безопасность:**

- Все данные пользователя (покупки, категории) должны храниться **исключительно локально** на устройстве.
- Приложение не должно требовать регистрации на сервере или передавать финансовые данные пользователя на внешние ресурсы.

#### **NFR.5. Совместимость:**

- Приложение должно корректно функционировать на устройствах под управлением ОС Android версии 8.0 (Oreo, API 26) и выше.

## **NFR.6. Требования к удобству использования (Usability):**

- **NFR.6.1. Навигация:** Навигация между основными разделами приложения («Главная», «История», «Аналитика», «Настройки») должна быть реализована с помощью панели навигации (Navigation Bar) в нижней части экрана, обеспечивая быстрый доступ к ключевым функциям в одно касание.
- **NFR.6.2. Обратная связь:** Система должна предоставлять пользователю своевременную и понятную обратную связь на его действия. Успешные операции (например, сохранение покупки) должны подтверждаться краткими уведомлениями (Toast/Snackbar). Ошибки ввода или системные сбои должны сопровождаться информативными диалоговыми окнами.
- **NFR.6.3. Единообразие интерфейса:** Все элементы управления (кнопки, поля ввода, переключатели) и экраны должны быть выполнены в едином стиле в соответствии с гайдлайнами Material Design 3. Похожие по назначению функции должны иметь схожее визуальное представление и поведение на разных экранах.
- **NFR.6.4. Минимизация ввода:** Для ускорения процесса ввода данных поля «Наименование товара» и «Категория» должны поддерживать механизм автоподсказок на основе ранее введенной информации. Поле «Дата» по умолчанию должно быть заполнено текущей датой.
- **NFR.6.5. Читаемость и восприятие:** Цветовая схема приложения должна обеспечивать достаточный контраст между текстом и фоном для комфортного чтения. Ключевые показатели на дашбордах (процент инфляции, суммы расходов) должны быть выделены крупным, легко читаемым шрифтом.

- **NFR.6.6. Язык и терминология:** Все текстовые элементы интерфейса, системные сообщения и подсказки должны быть выполнены на русском языке. Терминология (например, «Инфляция MoM», «Драйверы инфляции») должна сопровождаться краткими поясняющими подсказками при первом использовании или по требованию.

**NFR.7. Расширяемость:** Архитектура приложения должна позволять в будущем добавлять новые модули (например, синхронизацию с банками или модуль прогнозирования) с минимальными изменениями в существующем коде.

## **1.5. ВЫВОДЫ ПО ГЛАВЕ 1**

В рамках первой главы был проведен комплексный анализ предметной области и технологических основ для создания мобильного приложения по мониторингу личной инфляции. Полученные результаты позволяют сделать следующие ключевые выводы.

Во-первых, анализ существующих на российском рынке финансовых приложений показал, что, несмотря на их многообразие, ни одно из них не предоставляет специализированного функционала для удобного и методологически корректного расчета персонального индекса потребительских цен. Это подтверждает наличие свободной рыночной ниши и высокую актуальность разрабатываемого продукта.

Во-вторых, было проведено исследование классических методов расчета инфляции. В качестве базового подхода был выбран индекс Ласпейреса, как наиболее подходящий для отслеживания стоимости стабильной потребительской корзины. Была выявлена фундаментальная проблема неполных данных в пользовательских приложениях, для решения которой был разработан и обоснован инновационный **гибридный метод актуализации цен**. Данный метод, основанный на иерархии источников данных и статистической экстраполяции,

позволяет достичь оптимального баланса между точностью расчетов и удобством использования, что является ключевым конкурентным преимуществом будущего приложения.

В-третьих, анализ современных подходов к Android-разработке позволил сформировать надежный и актуальный технологический стек. Было принято решение использовать принципы **Чистой архитектуры (Clean Architecture)** для обеспечения масштабируемости и тестируемости проекта, а также архитектурный паттерн **MVVM** для слоя представления. Итоговый технологический стек будет включать язык **Kotlin**, декларативный фреймворк **Jetpack Compose** для построения UI, библиотеку **Room** для управления локальной базой данных и **Kotlin Coroutines** для асинхронных операций.

Таким образом, проведенное в первой главе исследование позволило не только подтвердить актуальность поставленной задачи, но и сформировать прочный научно-технический базис для дальнейшей работы. Результатом этого анализа стала разработка детализированных функциональных и нефункциональных требований к системе, включая требования к удобству использования. Данные требования служат формализованным техническим заданием, на основе которого в следующей главе будут выполнены этапы проектирования архитектуры и пользовательского интерфейса приложения.