МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра системного анализа и
автоматического управления

# РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ПРИЁМА ЛЕКАРСТВ С АВТОМАТИЧЕСКИМ РАСПОЗНАВАНИЕМ ПРЕПАРАТОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Селимова Сейфуллаха Селимовича

Научный руководитель
Доцент, к. ф.-м. н. _____ М. В. Корнилов

Заведующий кафедрой
к. ф.-м. н., доцент _____ И. Е. Тананко

Саратов 2026

# ВВЕДЕНИЕ

**Актуальность темы.** Каждый день миллионы людей принимают лекарства по назначению врача или для поддержания здоровья. Но в повседневных делах легко забыть вовремя выпить таблетку. Пропуск приёма может сделать лечение менее эффективным или даже привести к неприятным последствиям. Чтобы этого избежать, нужен инструмент, который будет напоминать о графике и помогать следить за курсом лечения.

Существует несколько мобильных приложений с похожими функциями, но многие из них требуют постоянного доступа к интернету, показывают рекламу или перегружены ненужными настройками. Поэтому разработка простого, удобного и полностью офлайн-приложения для контроля приёма лекарств остаётся актуальной задачей.

**Цель бакалаврской работы** — разработать мобильное приложение для контроля приёма лекарств с функциями добавления лекарств, создания напоминаний и отображения статистики.

Для достижения поставленной цели были определены следующие задачи:

1. обосновать выбор технологий разработки;
2. спроектировать базу данных для хранения информации о лекарствах и напоминаниях;
3. реализовать модуль добавления лекарств (ручной ввод и сканер);
4. реализовать модуль создания напоминаний (одиночные и групповые);
5. настроить отправку уведомлений с кнопками «Принял» и «Пропустить»;
6. разработать экран статистики для отображения прогресса по месяцам;
7. выполнить тестирование работоспособности приложения.

**Методологические основы** разработки мобильных приложений для платформы Android представлены в работах S. Braehler [1]. Проектирование баз данных рассмотрено в трудах К. Дж. Дейта [2]. Нейросетевые методы распознавания текста (OCR) базируются на работах Р. Смита [3] и С. Вика [4]. Вопросы реализации пользовательского интерфейса и хранения данных описаны в официальной документации Google [5; 6; 7]. Использование OCR-библиотек Tesseract и ML Kit описано в руководствах [8; 9].

**Практическая значимость бакалаврской работы**. В ходе выполнения выпускной квалификационной работы разработано мобильное прило-

жение для контроля приёма лекарств, не требующее подключения к интернету. Приложение включает ручное добавление и сканер упаковки с автоматическим распознаванием текста, гибкую систему напоминаний (одиночные и групповые курсы), уведомления с кнопками действий, а также экран статистики. Приложение может использоваться пациентами для соблюдения графика лечения в повседневной жизни.

**Структура и объем работы**. Бакалаврская работа состоит из введения, 3 разделов, заключения, списка использованных источников и одного приложения. Общий объем работы — 72 страницы, из них 53 страницы — основное содержание, включая 16 рисунков и 0 таблиц,цифровой носитель в качестве приложения, список использованных источников информации — 20 наименований.

## КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

**Первый раздел «Выбор технологий и архитектурные решения»** посвящен обоснованию выбора платформы разработки, языка программирования и архитектуры приложения.

В подразделе 1.1 приведено описание платформы Android, её архитектуры, системы разрешений и преимуществ перед iOS. Обоснован выбор языка Kotlin как официально рекомендуемого для Android-разработки.

Подраздел 1.2 посвящен архитектуре MVVM (Model-View-ViewModel). Рассмотрены функции модели, представления и ViewModel, а также их взаимодействие в разработанном приложении.

В подразделе 1.3 приводится краткий обзор используемых библиотек: Room для работы с базой данных, Jetpack Compose для построения интерфейса, WorkManager для планирования уведомлений, ML Kit и Tesseract для распознавания текста, Navigation Component для навигации между экранами.

В подразделе 1.4 описана структура исходного кода проекта и приведены сведения о размещении файлов на USB-накопителе.

**Вывод по первому разделу:** выбрана платформа Android, язык Kotlin, архитектура MVVM. Определены основные библиотеки: Room для базы данных, Jetpack Compose для интерфейса, WorkManager для уведомлений, ML Kit и Tesseract для распознавания текста.

**Второй раздел «Проектирование и реализация базы данных»**

посвящен описанию структуры базы данных разработанного приложения, а также реализации DAO и репозиториев для доступа к данным.

В подразделе 2.1 описываются реляционные базы данных и SQLite как основа для хранения данных в мобильных приложениях. Рассматриваются основные принципы организации данных, язык SQL и особенности SQLite (встраиваемость, компактность, поддержка транзакций).

Подраздел 2.2 посвящен библиотеке Room от Google. Рассматривается архитектура Room (Entity, DAO, Database) и её преимущества перед чистым SQLite: проверка SQL-запросов на этапе компиляции, минимальное количество шаблонного кода, встроенная поддержка Flow и корутин.

В подразделе 2.3 описана структура базы данных приложения представленная на рис. 1.

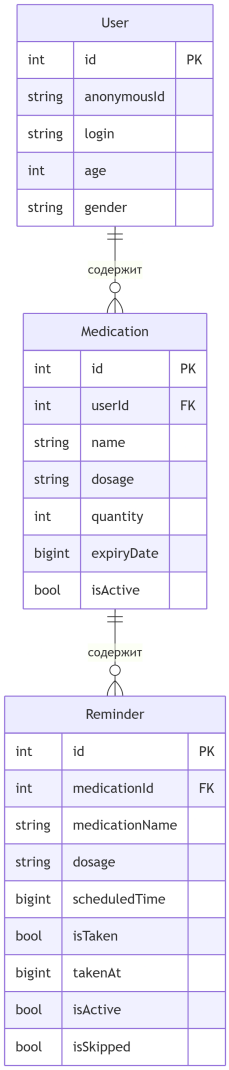


Рисунок 1 – Схема базы данных приложения

База данных состоит из трёх основных сущностей: *User* (пользователь), *Medication* (лекарство) и *Reminder* (напоминание). Связи между таблицами организованы с помощью внешних ключей. Для сохранения статистики за прошлые периоды используется механизм мягкого удаления — поле *isActive*. При удалении лекарства запись не исчезает, а только помечается как неактивная, что позволяет сохранить историю приёма.

В подразделе 2.4 описывается реализация DAO (Data Access Object). Приводится фрагмент интерфейса *ReminderDao* с основными методами.

```
@Dao
interface ReminderDao {
  @Insert
  suspend fun insertReminder(reminder: Reminder): Long

  @Query("SELECT * FROM reminders
  WHERE scheduledTime BETWEEN :start AND :end")
  fun getRemindersBetween(start: Long, end: Long): Flow<List<Reminder>>

  @Query("UPDATE reminders SET isTaken = 1 WHERE id = :id")
  suspend fun markAsTaken(id: Int)
}
```

Метод *getRemindersBetween* возвращает *Flow*, что позволяет UI автоматически реагировать на изменения в базе данных.

В подразделе 2.5 описываются репозитории как прослойка между DAO и ViewModel. Репозиторий скрывает от ViewModel источник данных и позволяет при необходимости добавить кэширование или синхронизацию с сервером.

```
class ReminderRepository(private val reminderDao: ReminderDao) {
  fun getRemindersBetween(start: Long, end: Long): Flow<List<Reminder>> =
  reminderDao.getRemindersBetween(start, end)

  suspend fun addReminder(reminder: Reminder): Long =
  reminderDao.insertReminder(reminder)
}
```

Всего в приложении создано три репозитория — для пользователей, лекарств и напоминаний.

**Вывод по второму разделу:** спроектирована база данных из трёх таблиц (User, Medication, Reminder) с внешними ключами. Реализованы DAO и репозитории.

**Третий раздел «Реализация пользовательского интерфейса и**

**основных модулей»** посвящен практической разработке экранов и модулей мобильного приложения для контроля приёма лекарств и тестированию их работоспособности, результаты которого зафиксированы на скриншотах в 3 разделе.

В подразделе 3.1 описываются общие принципы построения экранов на Jetpack Compose: декларативный подход, рекомпозиция, использование библиотеки Material Design 3. Взаимодействие между интерфейсом и данными организовано через ViewModel, которая хранит состояние экрана в виде StateFlow.

В подразделе 3.2 представлена навигация между экранами с помощью Navigation Component. Описаны три основных маршрута (аптечка, напоминания, статистика) и реализация нижней навигационной панели.

Подраздел 3.3 содержит описание модуля регистрации и входа. При первом запуске пользователь вводит имя, возраст и выбирает пол. После сохранения создаётся запись в таблице пользователей, и при последующих запусках регистрация не показывается.

В подразделе 3.4 описан экран аптечки — главный экран приложения после регистрации. Отображается список всех активных лекарств в виде карточек. Каждая карточка содержит название, дозировку, количество и срок годности. Реализовано мягкое удаление (лекарство помечается как неактивное, но остаётся в базе данных для сохранения статистики).

В подразделе 3.4.1 описано ручное добавление лекарства: форма с полями (название, дозировка, количество, срок годности), автозаполнение при вводе названия из ранее добавленных препаратов, валидация срока годности.

Подразделы 3.4.2 – 3.4.4 посвящены OCR (оптическому распознаванию символов). Описана архитектура CRNN (свёрточно-рекуррентная нейронная сеть), математический аппарат LSTM, а также настройка используемых библиотек — Tesseract (для распознавания русского текста) и ML Kit (для распознавания цифр и дат).

В подразделе 3.4.5 описана реализация сканера лекарств. Пользователь делает два снимка упаковки, изображения передаются в Tesseract и ML Kit для распознавания. После распознавания открывается экран редактирования. При сохранении выполняется проверка срока годности.

В подразделе 3.5 В верхней части расположен горизонтальный список

дней недели, основную часть занимает список напоминаний в виде карточек. Справа отображается иконка или переключатель активности. Ниже приведён фрагмент кода, реализующий выбор даты и реактивное обновление списка.

```
private val _selectedDate = MutableStateFlow(getToday())
val selectedDate: StateFlow<Long> = _selectedDate.asStateFlow()

val remindersForSelectedDate = _selectedDate
.flatMapLatest { date ->
  reminderRepository.getRemindersBetween(getStartOfDay(date), getEndOfDay(date)
  )
}
.stateIn(
scope = viewModelScope,
started = SharingStarted.WhileSubscribed(5000),
initialValue = emptyList()
)
```

В подразделе 3.6 описано создание напоминаний. При нажатии на плавающую кнопку открывается диалог выбора типа: одиночное или групповое. При создании одиночного напоминания пользователь выбирает лекарство и время, при групповом дополнительно задаются интервал в часах и количество дней. Система автоматически рассчитывает все приёмы. Количество таблеток уменьшается на общее число созданных напоминаний.

Подраздел 3.7 посвящён уведомлениям и планированию через WorkManager. WorkManager гарантирует выполнение задач даже после перезагрузки устройства. При создании напоминания рассчитывается задержка и формируется задача. Ниже приведён фрагмент кода, демонстрирующий планирование уведомления.

```
fun scheduleReminder(reminder: Reminder) {
  val delay = reminder.scheduledTime - System.currentTimeMillis()
  if (delay <= 0) return
  val workRequest = OneTimeWorkRequestBuilder<ReminderWorker>()
  .setInitialDelay(delay, TimeUnit.MILLISECONDS)
  .setInputData(Data.Builder().putString("title", reminder.medicationName)
  .putString("text", "Time to take pill ${reminder.medicationName} -
      ${reminder.dosage}").putInt("reminderId", reminder.id)
  .build()).addTag("reminder_${reminder.id}").build()
  WorkManager.getInstance(app).enqueue(workRequest)
}
```

При срабатывании задачи показывается уведомление с кнопками «Принял» и «Пропустить». При нажатии на кнопку срабатывает BroadcastReceiver,

обновляющий статус напоминания в базе данных.

В подразделе 3.8 описан экран статистики. Пользователь может переключать месяцы кнопками назад / вперёд (будущие месяцы недоступны). Круговая диаграмма показывает соотношение принятых и пропущенных таблеток. Карточка с цифрами отображает общее количество, принятые и пропущенные приёмы, а также процент принятых. Дополнительная карточка показывает точность приёма: сколько раз пользователь принял лекарство сразу (в первые 5 минут), а сколько с опозданием, и среднее время опоздания. Внизу — список лекарств, принимаемых в выбранном месяце.

**Вывод по третьему разделу:** разработаны все модули приложения — регистрация, аптечка, сканер с OCR, напоминания (одиночные и групповые), уведомления через WorkManager, статистика. Приложение протестировано и работает корректно.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы были успешно решены все поставленные задачи, что позволило достичь основной цели исследования — разработки мобильного приложения для контроля приёма лекарств с поддержкой ручного добавления и автоматическим распознаванием препаратов.

Были изучены и выбраны технологии разработки: платформа Android, язык Kotlin, архитектура MVVM,такие библиотеки, как Room, Jetpack Compose, WorkManager, ML Kit и Tesseract. Такой выбор обеспечил создание полностью автономного приложения с современным интерфейсом и надёжной системой напоминаний.

Спроектирована и реализована база данных на основе Room, состоящая из трёх таблиц: пользователи, лекарства и напоминания. Связи между таблицами организованы с помощью внешних ключей, а для сохранения статистики используется механизм мягкого удаления. Разработаны DAO и репозитории, обеспечивающие асинхронный доступ к данным.

Реализованы все ключевые модули приложения. Модуль добавления лекарств поддерживаете ручной ввод и автоматическое распознавание препаратов с фотографий упаковки на основе комбинации Tesseract (для русского языка) и ML Kit (для цифр и дат). Модуль напоминаний позволяет создавать одиночные и групповые напоминания с заданным интервалом. Настройка уведомлений через WorkManager гарантирует их отправку даже по-

сле перезагрузки устройства, а кнопки «Принял» и «Пропустить» в самом уведомлении позволяют оперативно обновлять статус. Для обработки случайно пропущенных уведомлений реализован механизм повторного опроса с помощью диалогов.

Разработан экран статистики, отображающий круговую диаграмму с соотношением принятых и пропущенных таблеток, карточку с числовыми показателями, точность реакции на уведомления (сразу или с опозданием) и список лекарств, принимаемых в выбранном месяце.

Все реализованные функции протестированы и работают корректно. Пользовательский интерфейс построен с использованием Jetpack Compose и Material Design 3, что обеспечивает единый стиль и удобство использования.

Практическая значимость работы подтверждается возможностью использования приложения в повседневной жизни для контроля приёма лекарств. Приложение полностью автономно, не требует доступа к интернету и сохраняет все данные на устройстве пользователя.

Таким образом, в результате проведённой работы создано полнофункциональное приложение, сочетающее удобный интерфейс, надёжную систему напоминаний и автоматическое распознавание препаратов с фотографий упаковки.

**Основные источники информации:**


1. Braehler, S. Android Architecture [Электронный ресурс] / S. Braehler // Institute for Telematics, University Karlsruhe (TH). – 2010. – URL: `https://os.itec.kit.edu/downloads/sa_2010_braehler-stefan_android-architecture.pdf` (дата обращения: 04.06.2026). – Загл. с экрана. – Яз. англ.
2. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт. – 6-е изд. – М. : Вильямс, 2016. – 1328 с.
3. Smith, R. An Overview of the Tesseract OCR Engine [Электронный ресурс] / R. Smith // Ninth International Conference on Document Analysis and Recognition (ICDAR 2007). – Curitiba, 2007. – URL: `https://ieeexplore.ieee.org/document/4376991` (дата обращения: 04.06.2026). – Загл. с экрана. – Яз. англ.
4. Wick, C. Improving OCR Accuracy on Early Printed Books using Deep Convolutional Networks [Электронный ресурс] / C. Wick, C. Reul, F.

Puppe // arXiv preprint. – 2018. – URL: https://arxiv.org/abs/1802.10033 (дата обращения: 04.06.2026). – Загл. с экрана. – Яз. англ.
5. Жемеров, Д. И. Kotlin в действии / Д. И. Жемеров, С. С. Исакова. – М. : ДМК Пресс, 2022. – 454 с.
6. Android Developers. Room persistence library [Электронный ресурс]. – URL: https://developer.android.com/training/data-storage/room (дата обращения: 04.06.2026). – Загл. с экрана. – Яз. англ.
7. Android Developers. WorkManager [Электронный ресурс]. – URL: https://developer.android.com/topic/libraries/architecture/workmanager (дата обращения: 04.06.2026). – Загл. с экрана. – Яз. англ.
8. Android Developers. Why adopt Compose [Электронный ресурс]. – URL: https://developer.android.com/develop/ui/compose/why-adopt (дата обращения: 04.06.2026). – Загл. с экрана. – Яз. англ.
9. Google ML Kit. Text Recognition guide [Электронный ресурс]. – URL: https://developers.google.com/ml-kit/vision/text-recognition (дата обращения: 04.06.2026). – Загл. с экрана. – Яз. англ.