УДК 004

РАЗРАБОТКА МОБИЛЬНОГО КЛИЕНТА НА ЯЗЫКЕ KOTLIN ДЛЯ АВТОМАТИЗАЦИИ ЛОГИСТИЧЕСКИХ ПРОЦЕССОВ ТРАНСПОРТНОГО ПРЕДПРИЯТИЯ

Ващенко Г.А., студент гр. ИТб-221, IV курс
Научный руководитель: Николаев П.И. доцент (к.н.) кафедры ИиАПС
Кузбасский государственный технический университет имени Т. Ф. Горбачева, г. Кемерово

В статье рассматривается автоматизация бизнес-процесса «Работа по перевозке» для логистического предприятия. Предложено решение на базе клиент-серверной архитектуры с использованием PostgreSQL и нативного Android-клиента на Kotlin, позволяющее сократить затраты на координацию водителей и повысить точность планирования поставок.

ООО «Сибирская Традиция» предоставляет логистические услуги и осуществляет поставку продуктов питания для розничных сетей. Анализ деятельности предприятия выявил базовый процесс, требующий автоматизации — «Работа по перевозке».

В настоящее время данный процесс не автоматизирован в части взаимодействия с водителями. Диспетчеры вынуждены использовать телефонные звонки и мессенджеры для информирования водителей о новых заказах, что приводит к человеческим ошибкам и потере актуальности данных.

Для выявления задач, которые необходимо решить для достижения поставленной цели, был проведен подробный анализ процесса выполнения базового бизнес-процесса и смежных с ним бизнес-процессов. Были декомпозированы бизнес-процессы, выявлены объекты данных, используемые для выполнения бизнес-процессов, выявлены элементарные деятельности (процессы), которые необходимо автоматизировать для достижения поставленной цели.

Цель разработки: сокращение затрат на координацию водителей, уменьшение простоев транспорта, повышение точности планирования за счёт автоматизации распределения заказов.

Проблема: взаимодействие с водителями осуществляется через телефонные звонки и мессенджеры, что приводит к ошибкам и потере актуальности данных.

Было принято решение о разработке android приложения, которое обеспечивало бы удобную работу с заказами для водителей.

Автоматизируемые подпроцессы:

- Распределение заказов (назначение перевозчиков);
- Ведение заявок (просмотр, принятие, отмена заказов).

Выбор технологии

Проведён анализ платформ: кроссплатформенные решения (Flutter, React Native), нативная iOS (Swift), нативная Android (Kotlin, Java).

Выбрано: нативная разработка для Android на Kotlin.

Обоснование:

- Распространённость Android-устройств среди сотрудников;
- Поддержка асинхронности через корутины;

Система построена по клиент-серверной архитектуре. Серверная часть использует СУБД **PostgreSQL 18**.

Ключевые сущности: couriers — данные водителей, orders — информация о заказах

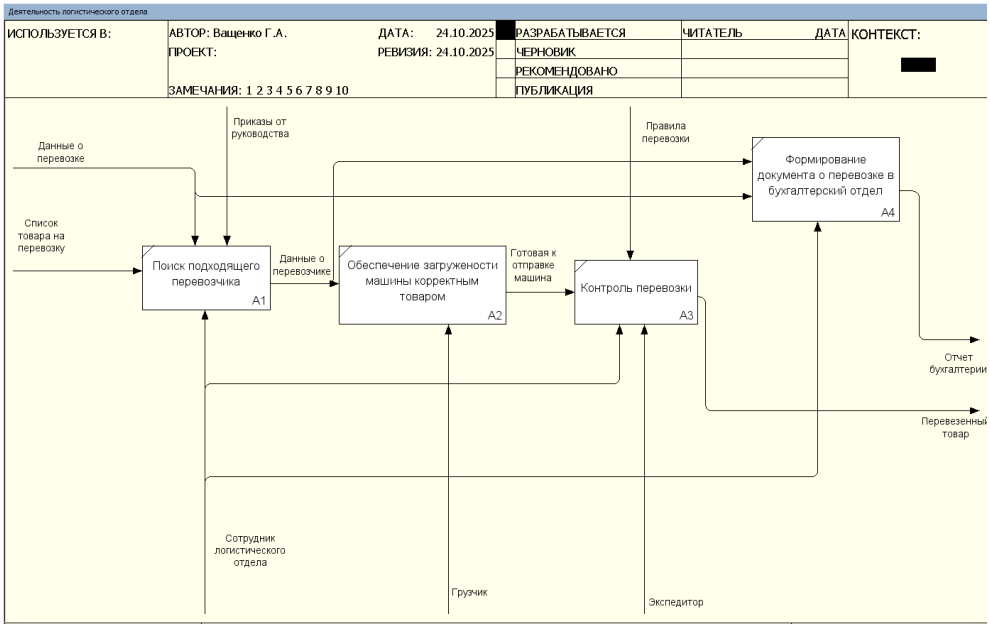

Рисунок 1 - Диаграмма декомпозиции бизнес-процесса

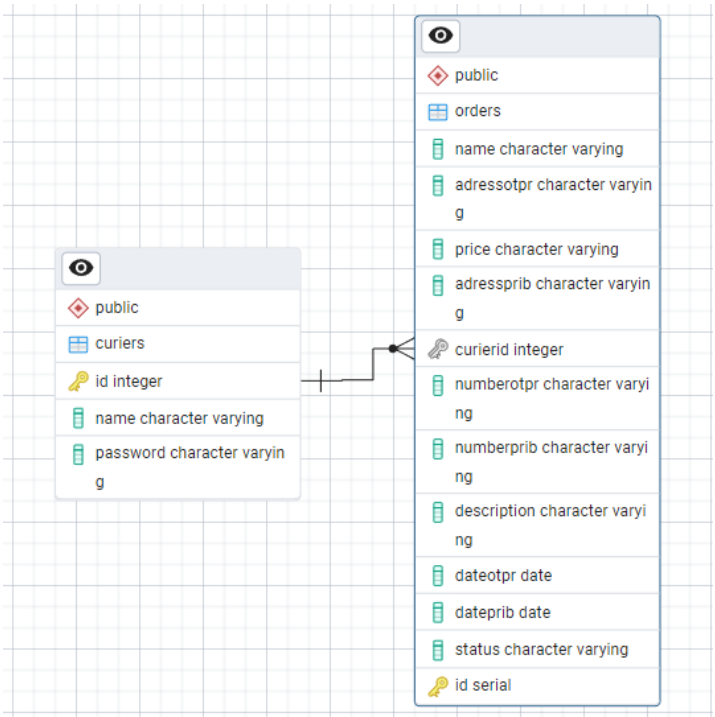

Рисунок 2 - ER-диаграмма базы данных

Выявленные методы для работы с хранимыми классами-сущностями реализованы в виде механизмов сервера (хранимых процедур, триггеров, функций). Использование механизмов позволило упростить разработку клиентской части, повысило производительность работы системы.

Реализация клиентской части - сценарий работы приложения, обрабатывающего данные от сервера базы данных, представлен на рисунке 3.

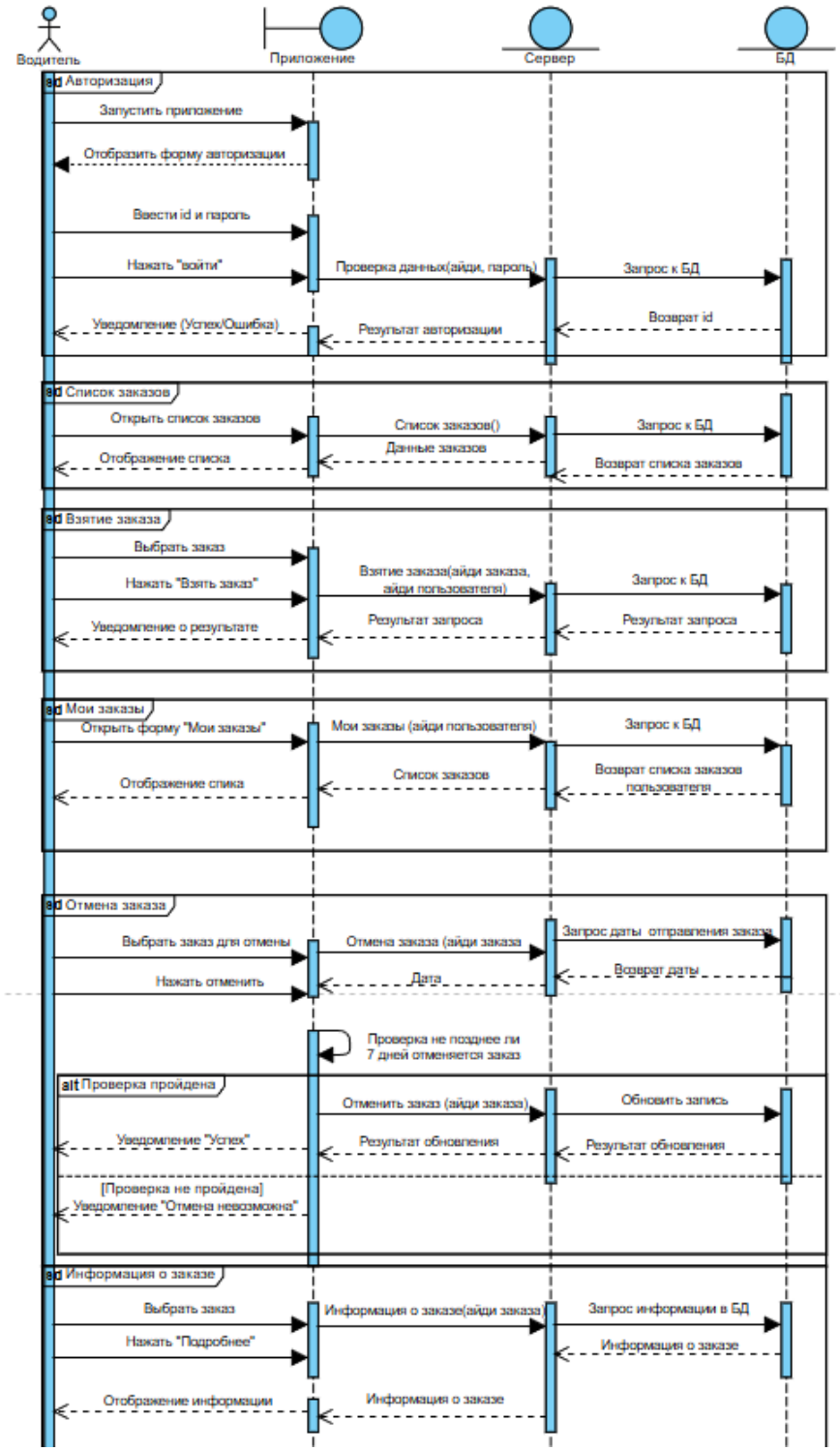

Рисунок 3 - Диаграмма последовательности взаимодействия клиента с сервером

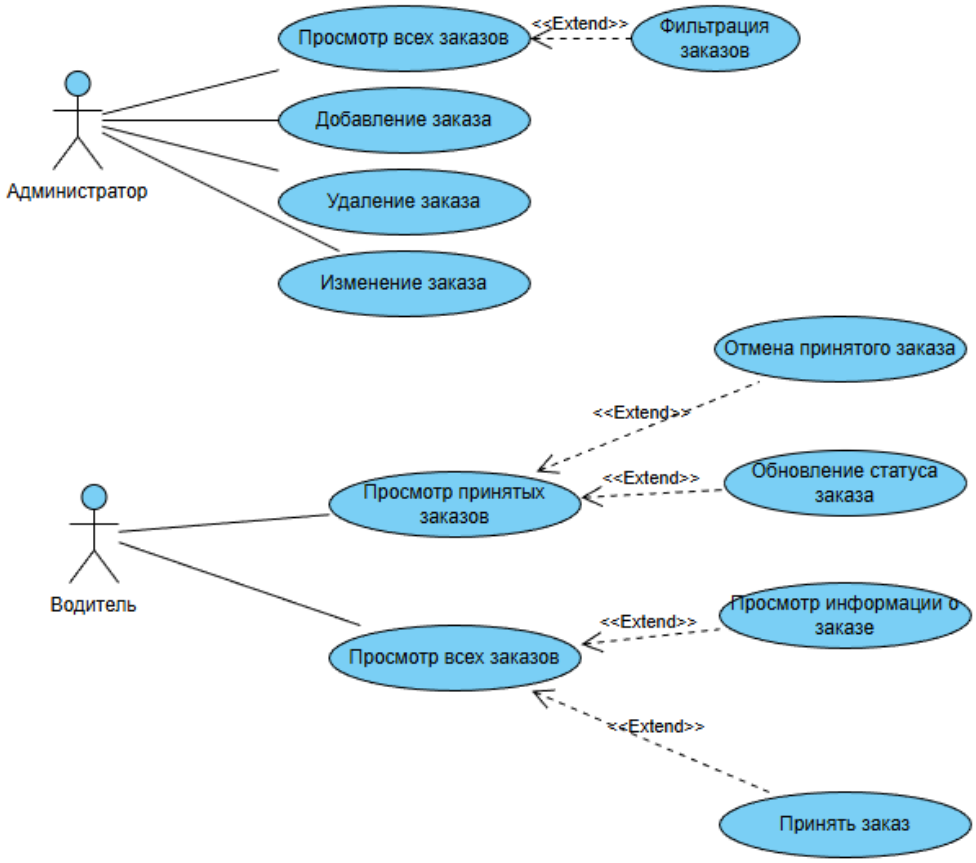

Рисунок 4 - Диаграмма вариантов использования

Программная реализация

Ниже приведены примеры реализации основных функций системы

```
data class Order(
    val id: Int,
    val name: String,
    val addressotpr: String,
    val addressprib: String,
    val price: String,
    val numberotpr: String,
    val numberprib: String,
    val description: String,
    val dateotpr: LocalDateTime,
    val dateprib: LocalDateTime,
    val status: String,
    val curierid: Int
)
```

Рисунок 5 – Модель данных заказа

```
class OrderService(private val repository: OrderRepository) {
    fun cancelOrder(
        orderId: Int,
        currentTime: LocalDateTime = LocalDateTime.now()
    ): CancelResult {

        val order = repository.findById(orderId)
            ?: return CancelResult.OrderNotFound

        if (order.status.equals("Новый", ignoreCase = true)) {
            return CancelResult.AlreadyCancelled
        }

        if (order.status.equals("Доставлен", ignoreCase = true)
            return CancelResult.OrderAlreadyShipped
        }

        val daysUntilShipping = ChronoUnit.DAYS.between(currentTime, order.dateotpr)
        if (daysUntilShipping < 7) {
            return CancelResult.TooLateToCancel
        }

        val updated = order.copy(
            status = "Новый",
            curierid = 0
        )

        return if (repository.update(updated)) {
            CancelResult.Success
        } else {
            CancelResult.DatabaseError
        }
    }
}
```

Рисунок 6 - Ограничение на отмену заказа (бизнес-правило)

Правило реализовано на клиенте для мгновенной реакции и дублируется на сервере для обеспечения целостности.

```
fun acceptOrderByCurier(
    orderId: Int,
    curierId: Int,
    repository: OrderRepository
): AcceptResult {

    val order = repository.findById(orderId)
        ?: return AcceptResult.OrderNotFound

    if (order.curierid != 0) return AcceptResult.AlreadyAssigned
    if (order.status == "Отменен") return AcceptResult.OrderCancelled

    val updatedOrder = order.copy(curierid = curierId, status = "Принят")

    return if (repository.update(updatedOrder)) {
        AcceptResult.Success
    } else {
        AcceptResult.DatabaseError
    }
}
```

Рисунок 7 - Бизнес-логика принятия заказа

Интерфейс пользователя

Интерфейс адаптирован под мобильные устройства: крупные элементы управления, минималистичная навигация.

Приложение имеет 3 основных экрана:

Главная (список доступных заказов): на этом экране возможно выбрать интересный заказ и принять его, а также узнать подробности о нем (описание, конкретные адреса)

Мои заказы (взятые пользователем заказы, которые еще не выполнены): на этом экране возможно завершить заказ, либо отменить его не позднее чем за неделю до отправки

Настройки

Скриншоты некоторых готовых форм мобильного приложения представлены ниже (рисунок 7, рисунок 8)

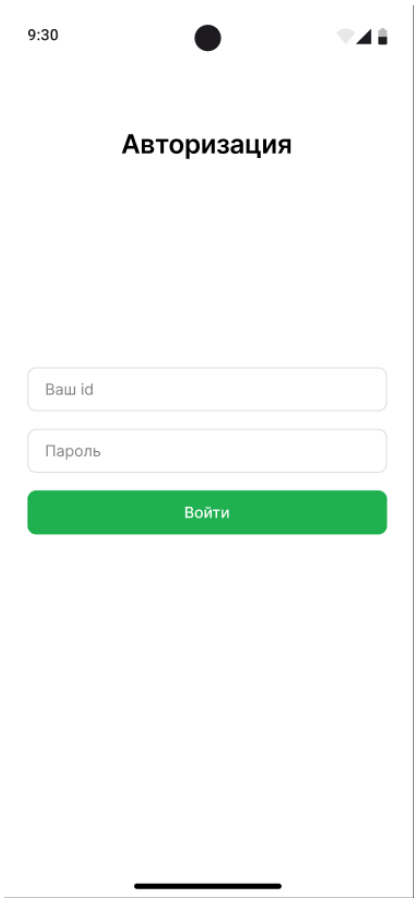

Рисунок 8 - Экран авторизации

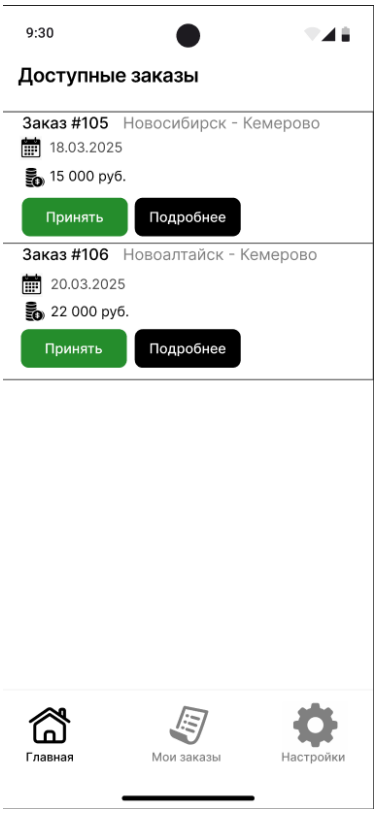

Рисунок 9 – Интерфейс главного экрана

Заключение

Было разработано мобильное приложение на Kotlin для удалённого доступа водителей к информационной системе распределения заказов, теперь заказы можно оперативно принимать самостоятельно водителям, исключив звонки и мессенджеры. Реализован удобный просмотр свободных заказов и управление статусом уже взятых. Ожидаемый эффект от внедрения - сокращение времени на координацию водителей, уменьшение ошибок при передаче заказов, повышение точности планирования. Дальнейшее развитие системы предполагает внедрение push-уведомлений о новых заказах и интеграцию с GPS-трекингом для мониторинга местоположения транспорта.

Список литературы

1. Документация по языку Kotlin. - URL [Электронный ресурс]: https://kotlinlang.org/docs/home.html (дата обращения: 20.03.2026). – Текст: электронный
2. Android Developers. [Электронный ресурс] URL: https://developer.android.com/ (дата обращения: 20.03.2026). - Текст: электронный
3. Документация PostgreSQL 18. [Электронный ресурс] URL: https://www.postgresql.org/docs/ (дата обращения: 20.03.2026). - Текст: электронный