Министерство образования Республики Беларусь

Учреждение образования  
 БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
 ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Анализ и оптимизация бизнес-процессов

Отчет

по лабораторной работе №1

на тему:

**РЕФАКТОРИНГ ПРОГРАММНОГО КОДА**

|  |  |  |
| --- | --- | --- |
| Проверил |  | А.В. Шелест |
|  | (подпись) |  |
| зачтено |  |  |
|  | (дата защиты) |  |
|  |  |  |
| Выполнил |  | Е.А. Корсаков  гр. 114301 |
|  | (подпись) |  |

Минск, 2025

Ссылка на репозиторий GitHub:

Программное средство BUSIK (Программное средство автоматизации управления пассажирских перевозок марушрутными такси) предназначена для упрощения процесса бронирования билетов на определенный рейс, а также упрощает работу водителям и операторам. Оно будет строится на платформе мобильного приложения для всех типов пользователей на Android Studio, Kotlinи использовать архитектуру MVC. В качестве базы данных была выбрана SQLite, т.к. она является основной базой данных в мобильных приложениях.

Для наглядности проектируемой архитектуры программного средства была построена диаграмма C4.

**Диаграмма C4 в GitHub:** <https://github.com/D1le1/BUSIK/blob/main/Documentation/Code/C4/Container.png>

На рисунке 1 представлена диаграмма C4 на контейнерном уровне.

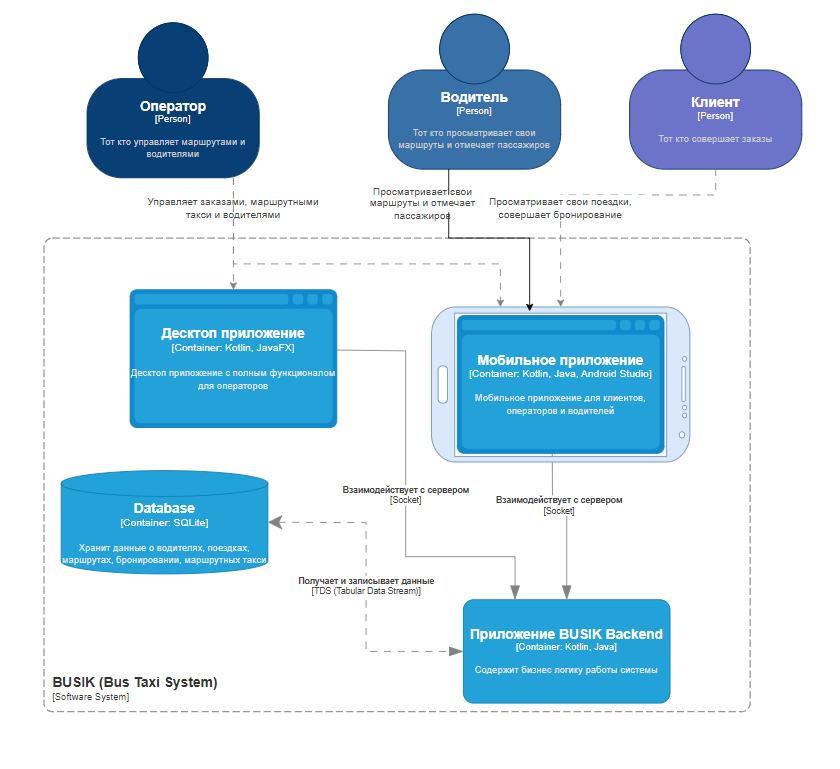


Рисунок 1 - Диаграмма С4 на контейнерном уровне

Затем была построена диаграмма С4 на компонентном уровне, представленная на рисунке 2.

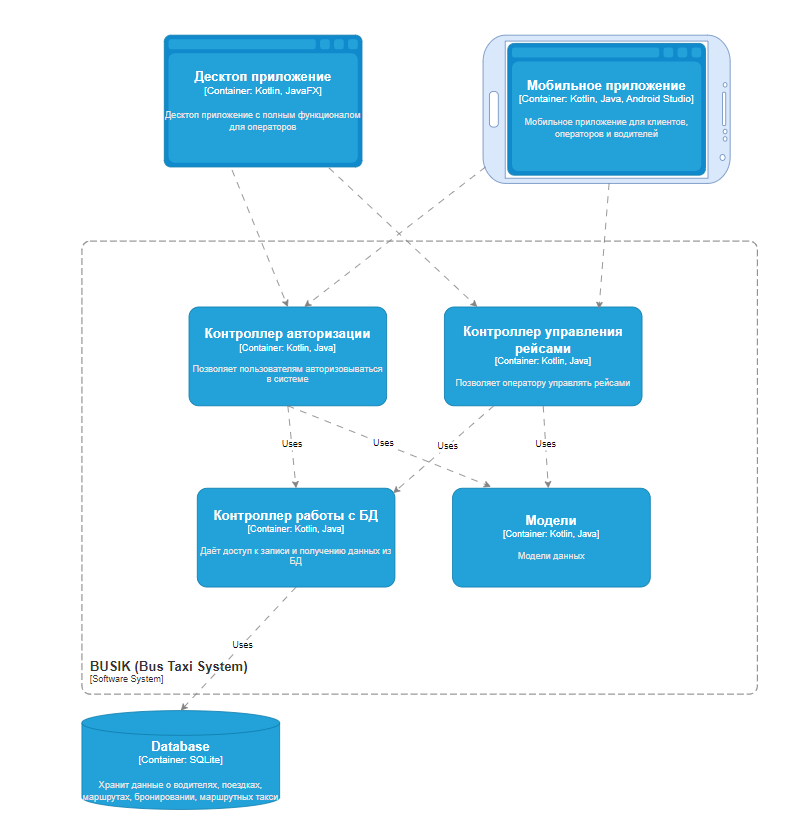


Рисунок 2 - Диаграмма С4 на компонентном уровне

Исходя из диаграммы компонентного уровня, RSAS Backend состоит из следующих компонентов:

* Контроллер авторизации: позволяет пользователям авторизовываться в системе и получать доступ к определенным функциям в зависимости от их роли.
* Контроллер управления рейсами: предназначен для пользователя «Оператор». Он позволяет ему полностью контролировать жизненный цикл рейса.
* Контроллер работы с БД: позволяет пользователям получать данные и записывать их в БД.

Приложения, разработанные с использованием чистой архитектуры, обладают высокой адаптивностью. Они легко подстраиваются под новые требования, сохраняя при этом качество кода и производительность. Такая структура упрощает понимание и сопровождение системы, делая её доступной для других разработчиков.

Чистая архитектура также помогает избежать проблем с зависимостями, разделяя приложение на логические модули. Это облегчает добавление новых функций и изменение существующих, сокращая время разработки и делая приложение более гибким и масштабируемым.

Clean Architecture состоит из нескольких принципов, которые помогают разработчикам создавать качественное и устойчивое программное обеспечение.

Ключевой принцип Clean Architecture – разделение приложения на уровни, каждый из которых выполняет свои задачи и управляет своей ответственностью. Обычно такое разделение выглядит следующим образом:

1. Интерфейс пользователя (UI): отвечает за взаимодействие с пользователем.
2. Представление (Presentation): отображает данные и обрабатывает пользовательский ввод.
3. Приложение (Application): содержит бизнес-логику и оркестрирует взаимодействие между различными компонентами системы.
4. Домен (Domain): содержит ядро системы и включает в себя основную бизнес-логику и модели данных.
5. Инфраструктура (Infrastructure): предоставляет реализации абстракций, таких как базы данных, внешние сервисы и т.д.

Уровень представления отвечает за взаимодействие с пользователем и обработку запросов. Уровень приложения выполняет бизнес-логику и координирует работу между уровнями представления и домена. Уровень домена содержит бизнес-логику и компоненты, отвечающие за работу с данными. Уровень инфраструктуры занимается поддержкой структур приложения и связью с внешними системами (например, базами данных, API и т.д.).

**Диаграмму классов в GitHub:** <https://github.com/D1le1/BUSIK/blob/main/Documentation/Code/Class%20Diagram/Class.pdf>

Таким образом, данная диаграмма классов иллюстрирует структуру системы, которая обеспечивает эффективное взаимодействие пользователей и бизнес-логики, в рамках концепции чистой архитектуры, способствуя созданию масштабируемого и поддерживаемого приложения розничной торговли.

**Макет дизайна программного средства:** <https://github.com/D1le1/BUSIK/blob/main/Documentation/UI/UI%20Pages/UIpages.pdf>

**Вывод:** В системе RSAS реализована интеграция мобильного прилоежния, десктоп приложения, серверного приложения и базы данных согласно принципам Clean Architecture. Система предназначена для автоматизации процесса бронирования и управления поездками посредством маршрутных такси. Она включает в себя три модуля: Мобильное приложение, десктоп приложения и Backend-приложение.

Мобильное приложение предназанчено для всех типов пользователей. Обеспечивает доступ ко всем функциям системы с любой точки в пару кликов.

Backend-приложение предназначено для обработки бизнес-логики и управления данными в системе управления рейсами. Оно обеспечивает взаимодействие между Мобильным и десктоп приложениями и базами данных, обрабатывает запросы пользователей, управляет аутентификацией и авторизацией.