Министерство образования Республики Беларусь

Учреждение образования  
 БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
 ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Рефакторинг и оптимизация программного кода

Отчет

по лабораторной работе №3

на тему:

**Паттерны проектирования в рефакторинге**

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

Минск, 2024

**Цель:** *Представить детали реализации ПС через UML-диаграммы*

Ссылка на репозиторий GitHub: <https://github.com/D1le1/BUSIK/tree/main>

1. **СТАТИЧЕСКИЕ И ДИНАМИЧЕСКИЕ ДИАГРАММЫ**
   1. **Статические UML диаграммы**
2. Диаграмма классов:

<https://github.com/D1le1/BUSIK/blob/main/Documentation/Code/Class%20Diagram/Class.pdf>

1. Диаграмма размещения представлена на рисунке 1.

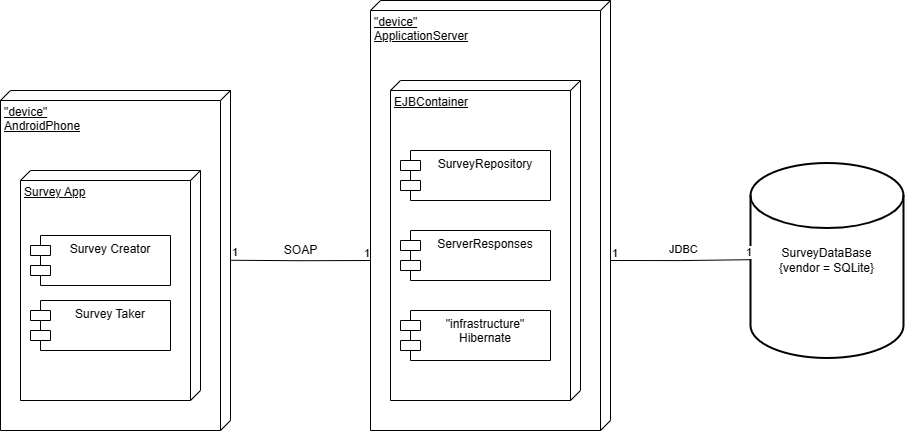


Рисунок 1 – Диаграмма размещения

1. Диаграмма состояния брони представлена на рисунке 2.



Рисунок 2 – Диаграмма состояния брони

**1.2 Динамические UML диаграммы**

1. Диаграмма взаимодействия представлена на рисунке 3.

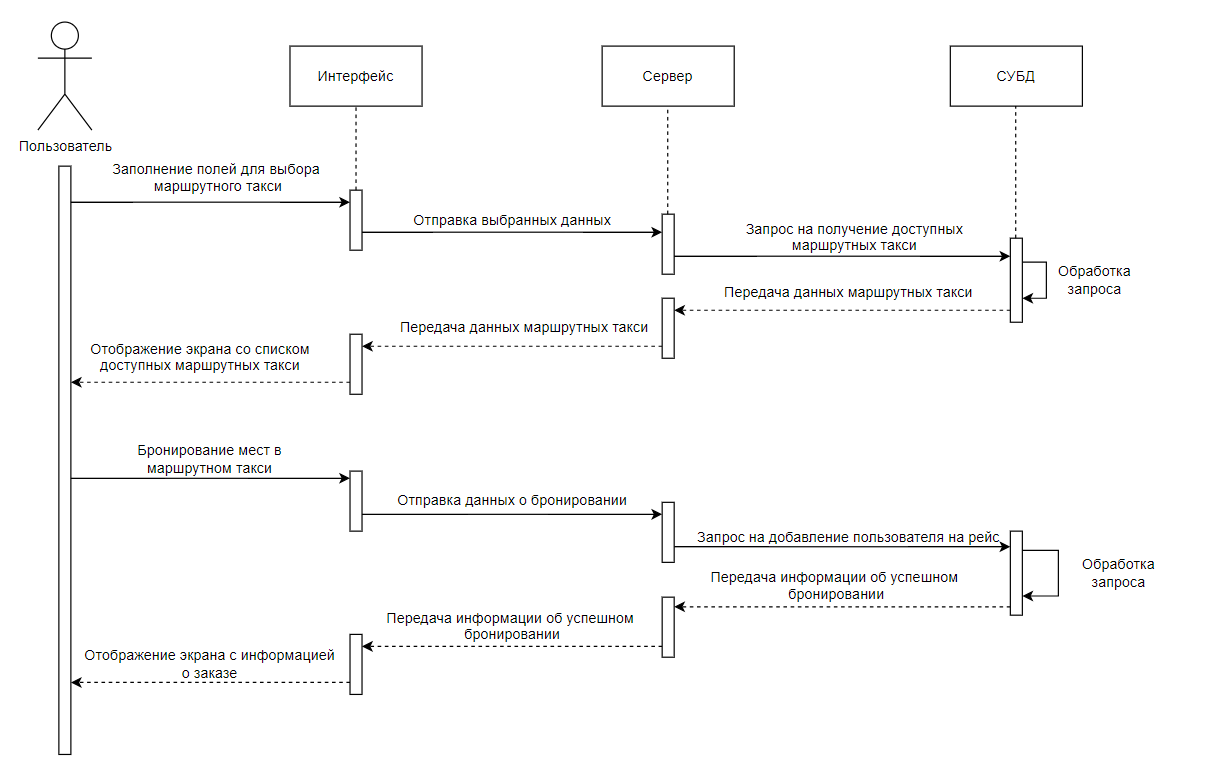


Рисунок 3 – Диаграмма взаимодействия

1. Диаграмма деятельности представлена на рисунке 4.

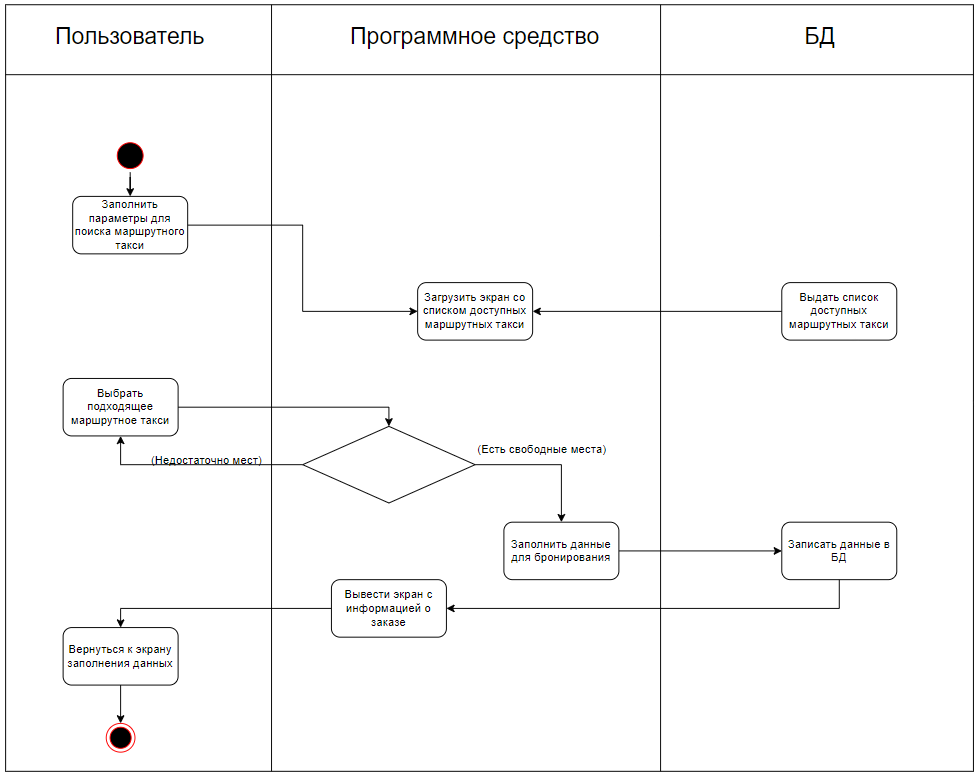


Рисунок 4 – Диаграмма деятельности

**2 ПРИНЦИПЫ ПРОГРАММИРОВАНИЯ И ПАТТЕРНЫ ПРОЕКТИВРОАНИЯ**

**2.1 Принципы программирования**

В процессе разработки программного средства использовалась объектно-ориентированная парадигма программирования (ООП). ООП основана на концепции объектов, которые могут содержать как данные (в виде полей, известных как свойства или атрибуты), так и код (в виде методов или процедур).

Ключевые принципы ООП включают:

1. Инкапсуляция: сокрытие внутренней реализации объекта и предоставление доступа к нему только через определенные методы.

2. Наследование: создание новых классов на основе существующих, что позволяет повторно использовать код.

3. Полиморфизм: способность методов обрабатывать объекты разных типов.

4. Абстракция: упрощение сложных систем путем моделирования классов на основе их основных характеристик.

Применение ООП способствует улучшению структуры и организации кода, облегчая его поддержку и расширение.

В ходе разработки программного средства также использовались следующие принципы программирования:

1. DRY (Don't Repeat Yourself): принцип, направленный на избегание дублирования кода, что повышает эффективность и упрощает сопровождение, обеспечивая уникальное представление каждого фрагмента знаний или логики.

2. KISS (Keep It Simple, Stupid): принцип, призывающий к простоте в проектировании и реализации программного обеспечения, избегая ненужной сложности.

3. SOLID: набор принципов ООП, направленных на улучшение дизайна и архитектуры программного обеспечения, включая принципы единой ответственности, открытости/закрытости, подстановки Лисков, разделения интерфейсов и инверсии зависимостей. Эти принципы способствуют созданию более устойчивого, гибкого и легко поддерживаемого кода.

**2.2 Паттерны проектирования**

В разрабатываемом программном средстве применяются следующие паттерны проектирования:

1. Адаптер – это структурный паттерн проектирования, который позволяет объектам с несовместимыми интерфейсами работать вместе. При этом адаптер оборачивает один из объектов, так что другой объект даже не знает о наличии первого. Например, вы можете обернуть объект, работающий в метрах, адаптером, который бы конвертировал данные в футы.

2. Фабричный метод – это порождающий паттерн проектирования, который определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов.

3. Одиночка — это порождающий паттерн проектирования, который гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа. Все реализации одиночки сводятся к тому, чтобы скрыть конструктор по умолчанию и создать публичный статический метод, который и будет контролировать жизненный цикл объекта-одиночки.

4. Строитель — это порождающий паттерн проектирования, который позволяет создавать сложные объекты пошагово. Строитель даёт возможность использовать один и тот же код строительства для получения разных представлений объектов. Паттерн Строитель предлагает вынести конструирование объекта за пределы его собственного класса, поручив это дело отдельным объектам, которые следует называть строителями.

**2.3 Алгоритм**

Схема алгоритма изменения статуса пассажира представлена на рисунке 5:

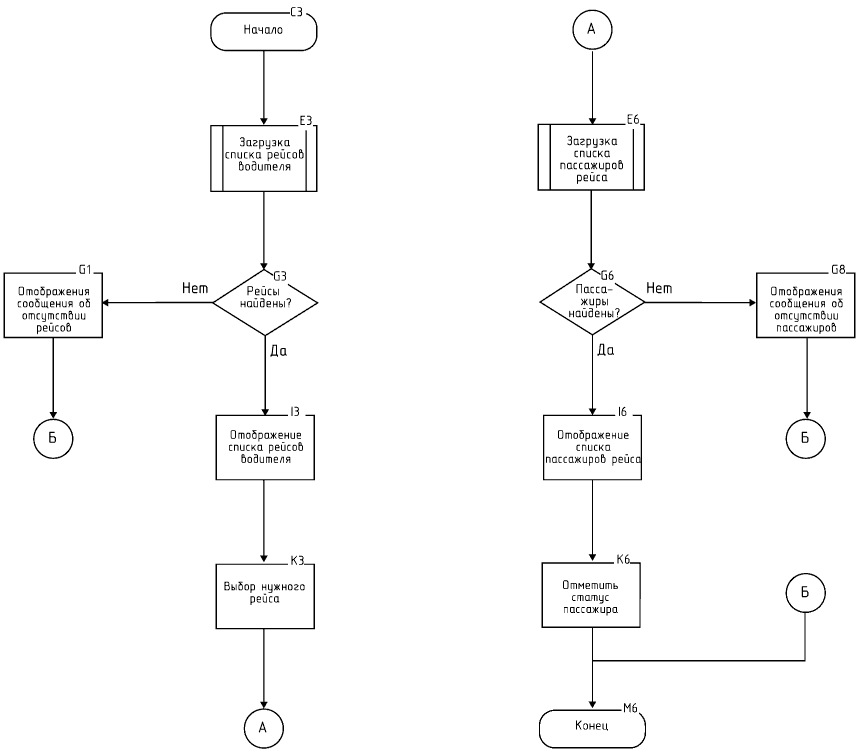


Рисунок 5 – Алгоритм изменения статуса пассажира

**Вывод**: в процессе проектирования архитектуры программного средства была использована парадигма программирования ООП, применялись принципы *KISS*, *DRY* и *SOLID*. Ключевым паттерном проектирования выступил паттерн *MVC*. В ходе выполнения работы были разработаны *UML*-диаграммы отражающие статические и динамические аспекты программного средства, а так же разработана диаграмма архитектуры ключевого бизнес-процесса.