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 |  | Ю-Ф.А. Тюленев  гр. 114301 |
|  | (подпись) |  |

Минск, 2025

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

Ссылка на репозиторий GitHub: <https://github.com/FranGranDev/RSAS>

**Статические UML диаграммы:**

1. Диаграмма классов: <https://github.com/FranGranDev/RSAS/blob/main/Documentation/Code%20Design/Core/ClassDiagram.png>
2. Диаграмма развертывания представлена на рисунке 1.

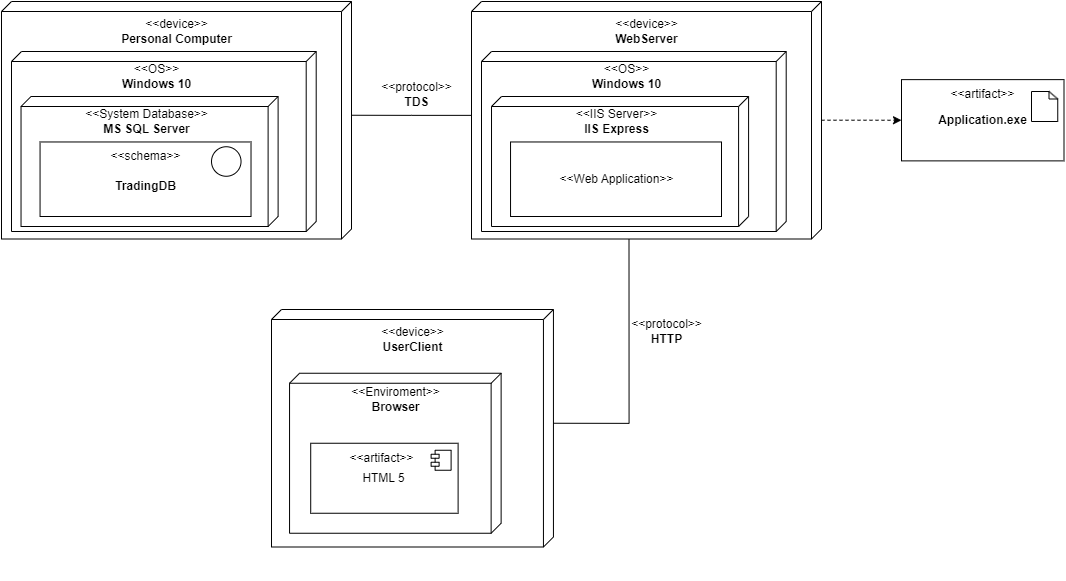


Рисунок 1 – Диаграмма развертывания

1. Диаграмма состояния заказа представлена на рисунке 2.

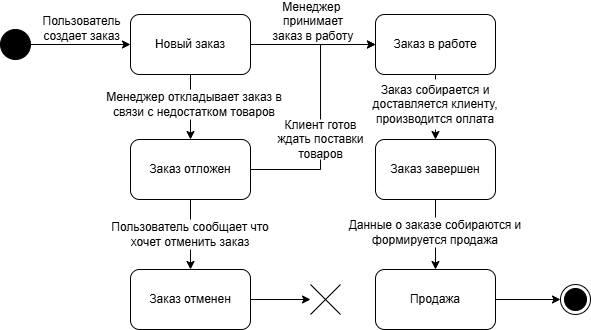


Рисунок 2 – Диаграмма состояния заказа

**Динамические UML диаграммы:**

1. Диаграмма взаимодействия представлена на рисунке 3.

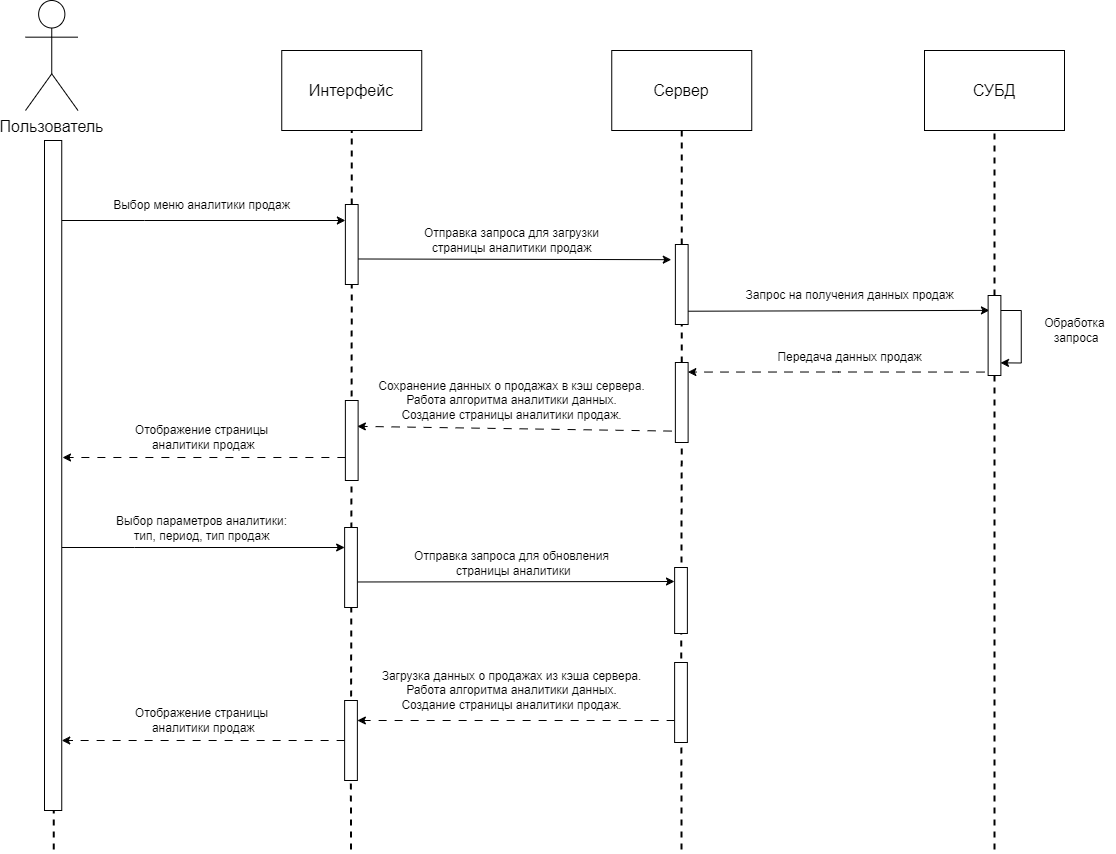


Рисунок 3 – Диаграмма взаимодействия

1. Диаграмма деятельности представлена на рисунке 4.

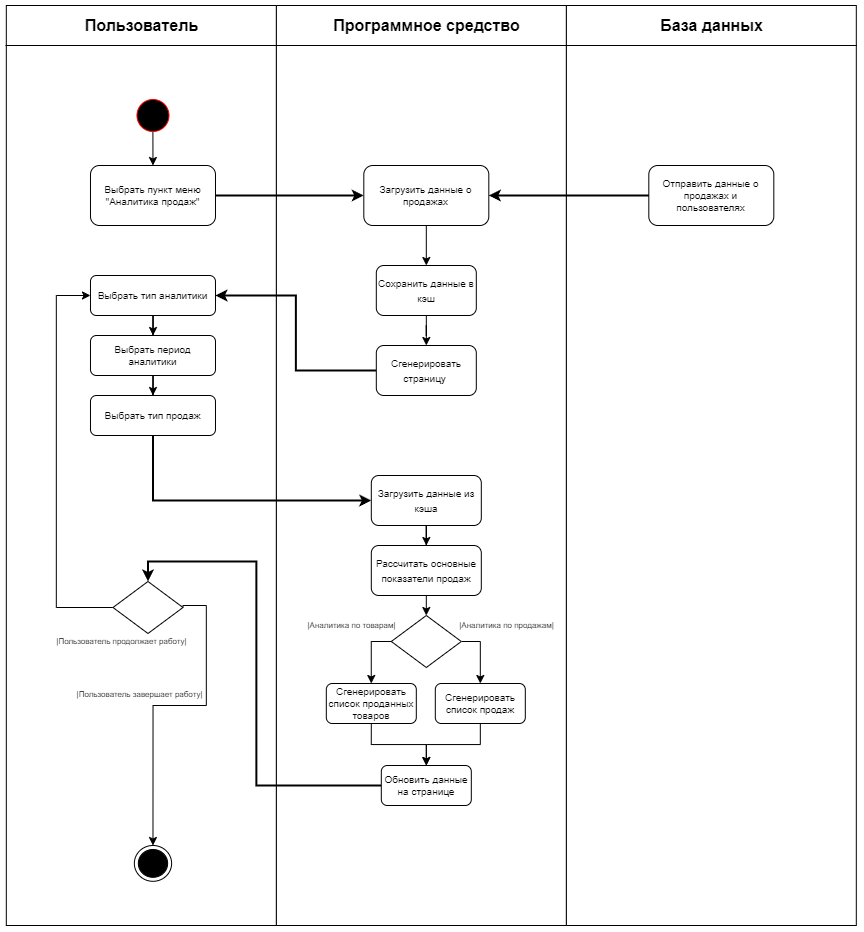


Рисунок 4 – Диаграмма деятельности

**Принципы программирования.**

При написании программного средства применялось объектно-ориентированное программирование (ООП). Объектно-ориентированное программирование (ООП) — это парадигма программирования, основанная на концепции объектов, которые могут содержать данные и код: данные в виде полей (часто известных как свойства или атрибуты), и код в виде процедур (часто известных как методы).

Основные принципы ООП включают:

* Инкапсуляция: скрытие внутренней реализации объекта и предоставление доступа только через определенные методы.
* Наследование: создание новых классов на основе существующих, что позволяет повторно использовать код.
* Полиморфизм: способность методов обрабатывать объекты разных типов.
* Абстракция: упрощение сложных систем путем моделирования классов на основе их основных характеристик.

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

В ходе разработки программного средства применялись следующие принципы программирования:

* *DRY (Don't Repeat Yourself)*: это принцип разработки программного обеспечения, который призывает избегать дублирования кода. Он направлен на повышение эффективности и упрощение сопровождения, обеспечивая, чтобы каждый фрагмент знаний или логики был уникально представлен в системе.
* *KISS (Keep It Simple, Stupid)* — это принцип разработки, который призывает к простоте в проектировании и реализации программного обеспечения. Он предполагает, что системы должны быть максимально простыми и понятными, избегая ненужной сложности.
* *SOLID*: это набор принципов объектно-ориентированного программирования, направленных на улучшение дизайна и архитектуры программного обеспечения. Он включает принципы единичной ответственности, открытости/закрытости, подстановки Лисков, разделения интерфейса и инверсии зависимостей. Эти принципы помогают создавать более устойчивый, гибкий и легко поддерживаемый код.

**Паттерны проектирования.**

В разрабатываемом программном средстве применяются следующие паттерны проектирования:

* *MVC*: Паттерн проектирования *MVC (Model-View-Controller)* разделяет приложение на три взаимосвязанных компонента: модель, представление и контроллер. Модель управляет данными и бизнес-логикой, представление отвечает за отображение данных пользователю, а контроллер обрабатывает пользовательский ввод и взаимодействует с моделью. Это разделение упрощает управление сложностью, улучшает модульность и облегчает сопровождение кода. Иллюстрация паттерна *MVC* представлена на рисунке 5:

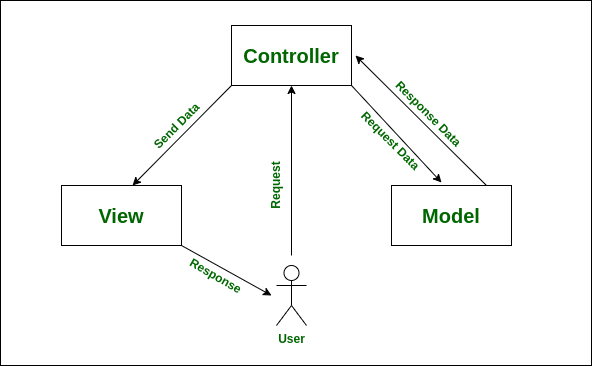


Рисунок 5 – Паттерн проектирования *MVC*

* *Dependency Injection*: Паттерн проектирования *Dependency Injection (DI)* это техника, в которой объект получает свои зависимости от внешнего источника, а не создаёт их самостоятельно. Это способствует снижению связанности между компонентами, улучшает модульность, облегчает тестирование и упрощает замену компонентов в приложении.
* *Builder*: Паттерн проектирования Builder — это порождающий паттерн, который используется для пошагового создания сложных объектов. Он позволяет отделить процесс конструирования объекта от его представления, обеспечивая возможность создавать различные представления объекта с помощью одного и того же процесса конструирования. Это повышает гибкость и упрощает управление конструкцией сложных объектов.

**Алгоритмы.**

Схема алгоритма аналитики заказов и продаж представлена на рисунке 6:

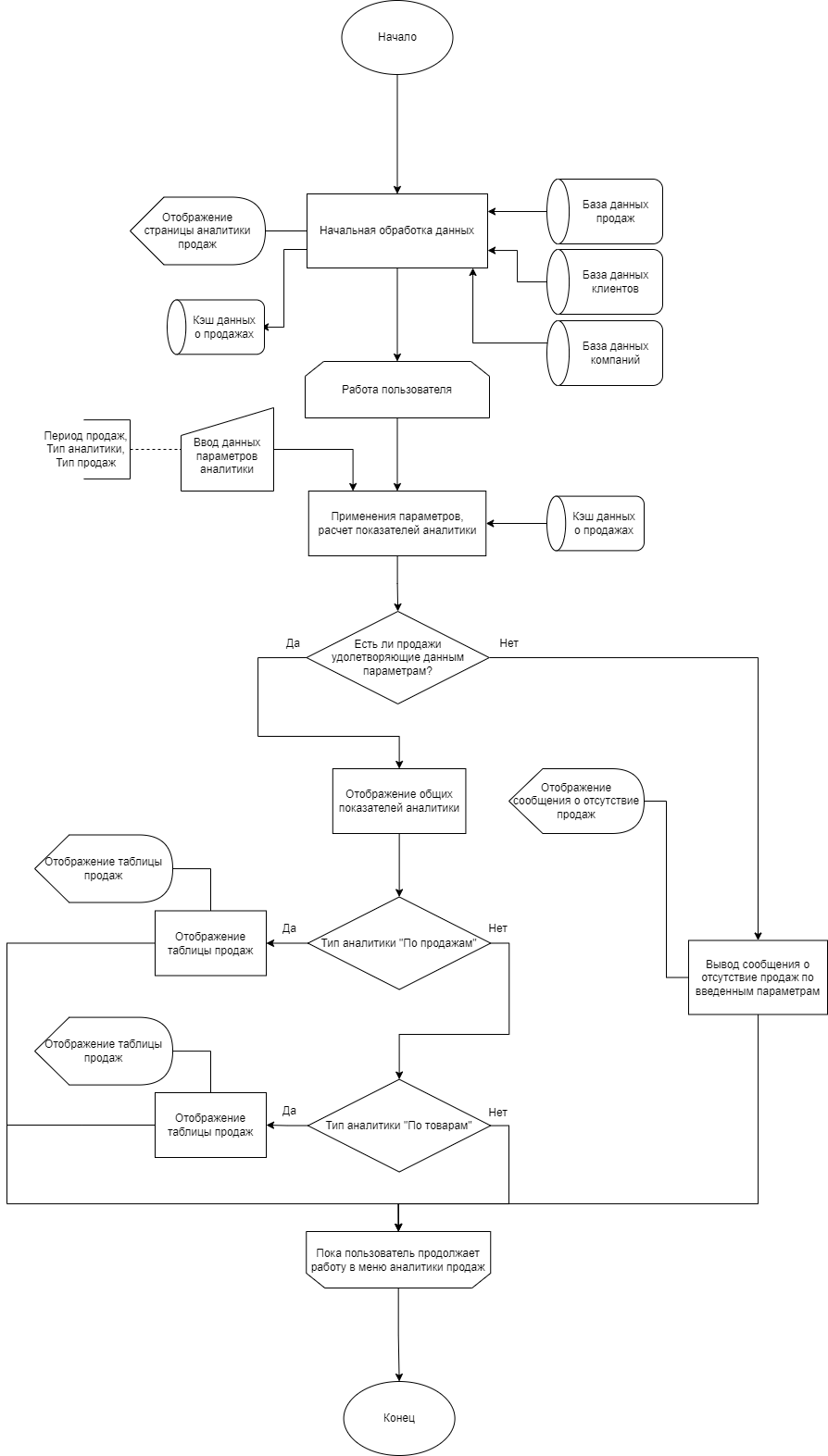


Рисунок 6 – Алгоритм аналитики заказов и продаж

**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.