Белорусский государственный университет информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Отчет по лабораторной работе № 3

на тему

«Исследование архитектурного решения»

Выполнили: студенты группы 050541:
Соколов Д.В.
Островский А.А.
Лейкина А.Э.

Проверил: Жалейко Д.А.

Минск
2023

1 ЦЕЛЬ РАБОТЫ

Целью работы является проектирование архитектуры приложения, проведение анализа архитектуры, сравнение и рефакторинг.

2 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ

2.1 Определение типа приложения

Приложение «CalculatorGCD» разрабатывается для широкой клиентской аудитории на различных платформах, ему не требуется поддержание насыщенного UI и мультимедии, требуется простая модель развертывания в Веб, пользовательский интерфейс должен быть независимым от платформы, максимальное сокращение потребления ресурсов (дисковое пространство и вычислительная мощность процессора).

Учитывая вышеописанные требования приложение должно быть реализовано как Веб-приложение.

2.2 Выбор стратегии развертывания

При разработке приложения нет необходимости в распределении компонентов по разным физическим уровням. Для простоты и улучшения производительности в приложении должно использоваться нераспределенное развертывание (рисунок 2.1).

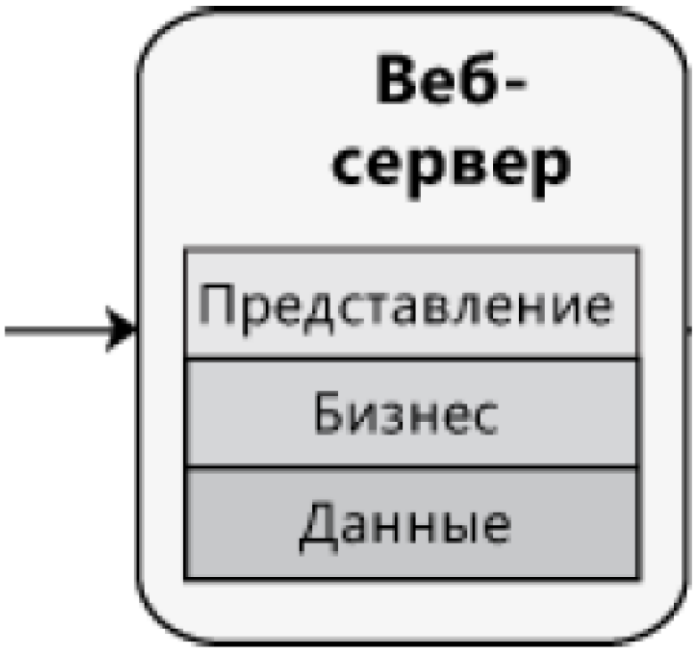

Рисунок 2.1 – Схема нераспределенного развертывания

2.3 Обоснование выбора технологий

В процессе разработки приложения «GCD Calculator» необходимо разделение уровней бизнес-логики и представления. Бизнес-логика должна быть реализована в библиотеке классов, а уровень представления – как приложение MVC.

Приложение «GCD Calculator» разрабатывается в среде ASP.NET на платформе .NET 6. Для клиентской части применяется язык гипертекстовой разметки HTML\CSS, библиотека Bootstrap, языки программирования JavaScript; для серверной части – язык программирования C#, стандартные библиотеки C#.

Обоснованием выбранных технологий являются необходимость работоспособности приложения на совершенно разных клиентских компьютерах, на клиентские компьютеры ничего не требуется устанавливать, реализация простых функций без насыщенных UI, анимации и мультимедии.

2.4 Показатели качества

Приложение должно удовлетворять следующим параметрам качества:
– концептуальная целостность;
– удобство и простота обслуживания;
– возможность повторного использования;
– доступность;
– производительность;
– надежность;
– масштабируемость;
– безопасность;
– обеспеченность технической поддержкой;
– тестируемость;
– удобство и простота использования.

2.5 Пути реализации сквозной функциональности

2.5.1 Управление исключениями

В приложении должна быть реализована обработка исключений ошибочно введенных данных, при которой приложение должно выдать предупреждение и ожидать ввода корректных данных.

Для этой цели подходит стратегия перехвата, обертывания и формирования исключений. Эта стратегия обеспечит перехват универсальных исключений с последующей очисткой ресурсов или любой другой соответствующей обработкой. Если не удается обработать ошибку, исключение заключается в другое исключение, более уместное для вызывающей стороны, и после этого формируется новое исключение для обработки кодом, расположенным выше в стеке кода.

2.5.2 Управление состоянием

В приложении должно присутствовать управление состоянием, так как требуется хранение введенных Пользователем данных. При этом приложение должно сохранять минимальный объем данных, не сохраняя данные на локальный диск или сервер, т. к. нет необходимости в долгосрочном хранении данных, и они могут быть удалены после перезапуска приложения.

2.6 Структурная схема приложения

Диаграмма компонентов приложения (архитектура «To be») приведена на рисунке 2.2.

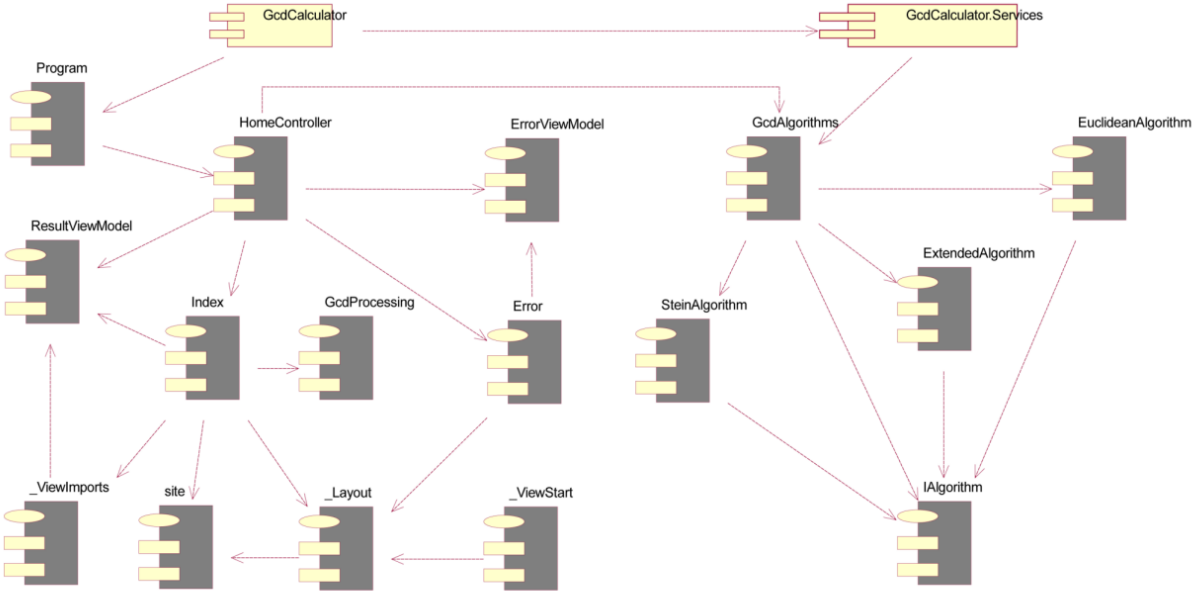

Рисунок 2.2 – Архитектура приложения «To be»

3 АНАЛИЗ АРХИТЕКТУРЫ

На рисунке 3.1 представлена диаграмма классов уровня бизнес-логики написанного кода, продемонстрированного на первом Sprint Review, т.е. архитектура приложения «As is».

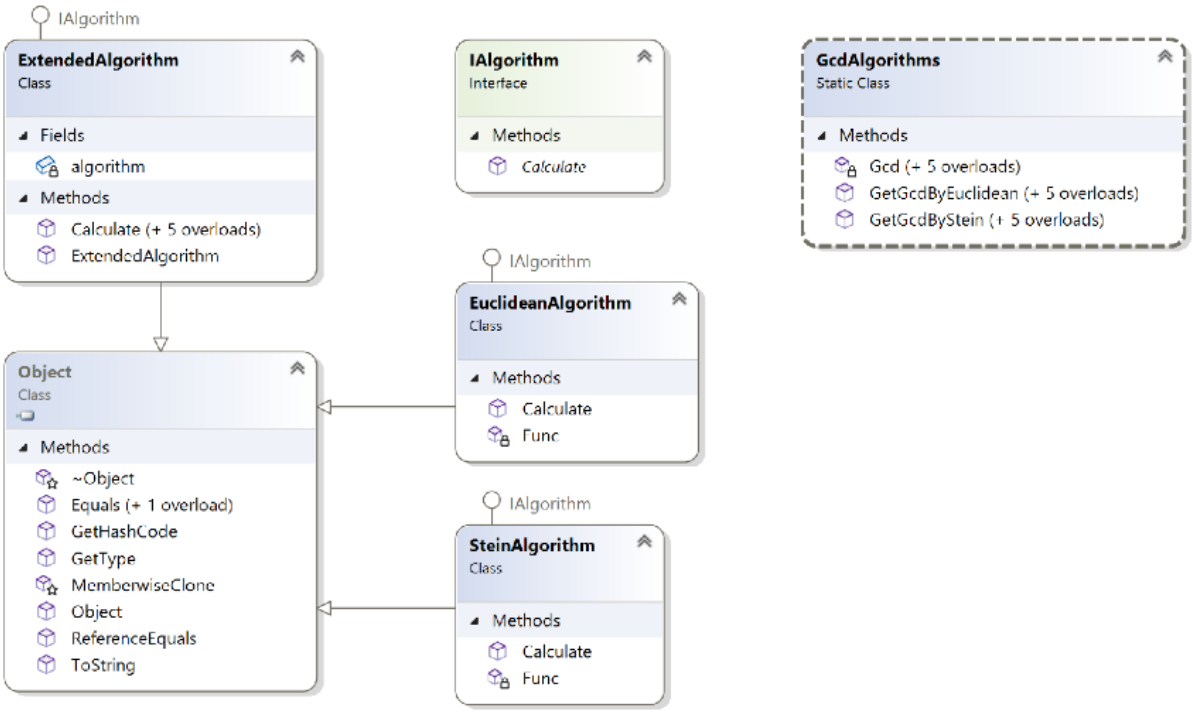

Рисунок 3.1 – Архитектура приложения «As is».

4 СРАВНЕНИЕ И РЕФАКТОРИНГ

Анализируя диаграммы, изображенные на рисунках 2.2 и 3.1 можно сделать вывод, что все требования соблюдены и полученная архитектура (архитектура «As is») соответствует ранее спроектированной архитектуре приложения (архитектура «To be»).