МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики

Кафедра информационных технологий

ОТЧЕТ

о выполнении лабораторной работы № 7

по дисциплине «Программная инженерия»

Выполнила: ст. гр. МО 32/2

Переузник В. С.

Проверил: доцент каф. ИТ

Полетайкин А. Н.

Краснодар

2025

Тема: разработка архитектуры программного продукта.

Индивидуальная тема: «Программа для аппроксимации функции».

Цель: приобретение навыков проектирования физической архитектуры ПП на языке UML, а также разработки базы данных для ПП.

Задание: 1) Опираясь на специальные требования к информационному обеспечению, представленные в техническом задании, а также на диаграмму классов UML, разработать решения по организации структур данных в ПП; 2) Выполнить описание структур данных на уровне атомарных атрибутов с указанием типов данных, ограничений на значения и значений по умолчанию. Каждую структуру данных представить в формате таблицы; 3) Если в качестве структуры данных разрабатывается база данных (БД), представить её нормализованную модель данных, список разработанных таблиц БД, а также описание связей между таблицами. Если нет – пропустить пункт; 4) Опираясь на специальные требования к математическому обеспечению, представленные в техническом задании, а также на диаграммы поведения UML, разработанные при выполнении лабораторной работы 6, перечислить и описать программные компоненты ПП в формате таблицы. Имена компонентов файлов привести с указанием расширений; 5) Построить диаграмму компонентов UML, отражающую взаимодействие его программных компонентов между собой и с компонентами информационного обеспечения ПП; 6) Выполнить описание физических элементов комплекса технических средств, необходимых для функционирования ПП, в формате таблицы; 7) Построить диаграмму развертывания UML, выражающую зависимости между узлами комплекса технических средств и развернутыми на них программными компонентами.

Ход выполнения работы:

1. Структуры данных ПП

На основании анализа технического задания и диаграммы классов UML разработаны структуры данных для основных сущностей программного продукта. При проектировании физической архитектуры было принято решение объединить некоторые концептуальные классы из UML-диаграммы в более эффективные структуры данных, что позволило упростить архитектуру без потери функциональности.

Для начала была определена базовая структура DataPoint, описанная в таблице 1.1, которая представляет одну экспериментальную точку. Используется для хранения исходных данных и промежуточных расчетов.

Атрибут	Тип данных	Размер, байт	Условие на значение	Значение по умолчанию	Примечание
x	number	8	Любое вещественное число	-	Координата X экспериментальной точки
y	number	8	Любое вещественное число	-	Координата Y экспериментальной точки

Таблица 1.1 - Структура данных DataPoint

Структура ExperimentalData, описанная в таблице 1.2, объединяет функционал классов "Экспериментальные данные" и "Математическая модель" из UML-диаграммы, так как они тесно связаны в логике моего веб-приложения.

Атрибут	Тип данных	Размер, байт	Условие на значение	Значение по умолчанию	Примечание
points	Array<DataPoint>	динамический	length ≥ 2	[]	Массив экспериментальных точек
isValid	boolean	1	true/false	false	Флаг корректности данных
functionType	string	20	"linear", "quadratic", "cubic", "exponential"	"linear"	Выбранный тип аппроксимирующей функции

Таблица 1.2 - Структура данных ExperimentalData

Структура ApproximationResult, описанная в таблице 1.3, представляет результаты расчетов и объединяет данные классов "Аппроксимация" и

"Математическая модель" из UML, поскольку результаты расчетов неразрывно связаны с использованной математической моделью.

Атрибут	Тип данных	Размер, байт	Условие на значение	Значение по умолчанию	Примечание
coefficients	Array<number>	динамический	length = 2-4	[]	Коэффициенты аппроксимирующей функции
formula	string	100	Непустая строка	""	Аналитическое представление функции
rSquared	number	8	$0 \leq rSquared \leq 1$	0	Коэффициент детерминации R^2
functionType	string	20	"linear", "quadratic", "cubic", "exponential"	"linear"	Тип использованной функции

Таблица 1.3 - Структура данных ApproximationResult

Структура Report, описанная в таблице 1.4, объединяет функционал классов "Отчет" и "График" из UML-диаграммы, так как визуализация является неотъемлемой частью итогового отчета.

Атрибут	Тип данных	Размер, байт	Условие на значение	Значение по умолчанию	Примечание
dataPoints	Array<DataPoint>	динамический	минимум 2 точки	[]	Исходные точки данных
curvePoints	Array<DataPoint>	динамический	минимум 2 точки	[]	Точки для рисования кривой функции
config	Object	динамический	-	{}	Настройки графика

Таблица 1.4 - Структура данных Report

Разработанные структуры образуют последовательную цепочку обработки информации, где каждая последующая структура использует данные предыдущей, обеспечивая сквозную обработку информации от ввода до финального отображения.

В данном ПП не используется реляционная база данных, так как все данные хранятся в оперативной памяти браузера во время сеанса работы. Поэтому пункт 3 задания пропущен.

2. Программные компоненты ПП

На основе анализа функциональных требований и спроектированных структур данных определен состав программных компонентов, представленный в таблице 2.1.

№	Имя	Стереотип	Описание
1	index.html	«form»	Главная веб-страница с пользовательским интерфейсом
2	app.js	«source»	Основной класс приложения, управляет работой всех модулей
3	data-manager.js	«source»	Модуль для работы с данными: ввод, проверка, хранение
4	math-processor.js	«source»	Модуль математических расчетов методом наименьших квадратов
5	math.js	«library»	Внешняя библиотека для матричных вычислений и решения системы уравнений
6	chart.js	«library»	Внешняя библиотека для построения и визуализации графиков
7	styles.css	«document»	Файл стилей для оформления интерфейса

Таблица 2.1 - Перечень программных компонентов ПП

Диаграмма компонентов, представленная на рисунке 2.2, показывает, из каких частей состоит программа и как они связаны между собой. Она

помогает понять, какой файл за что отвечает и как они обмениваются данными.

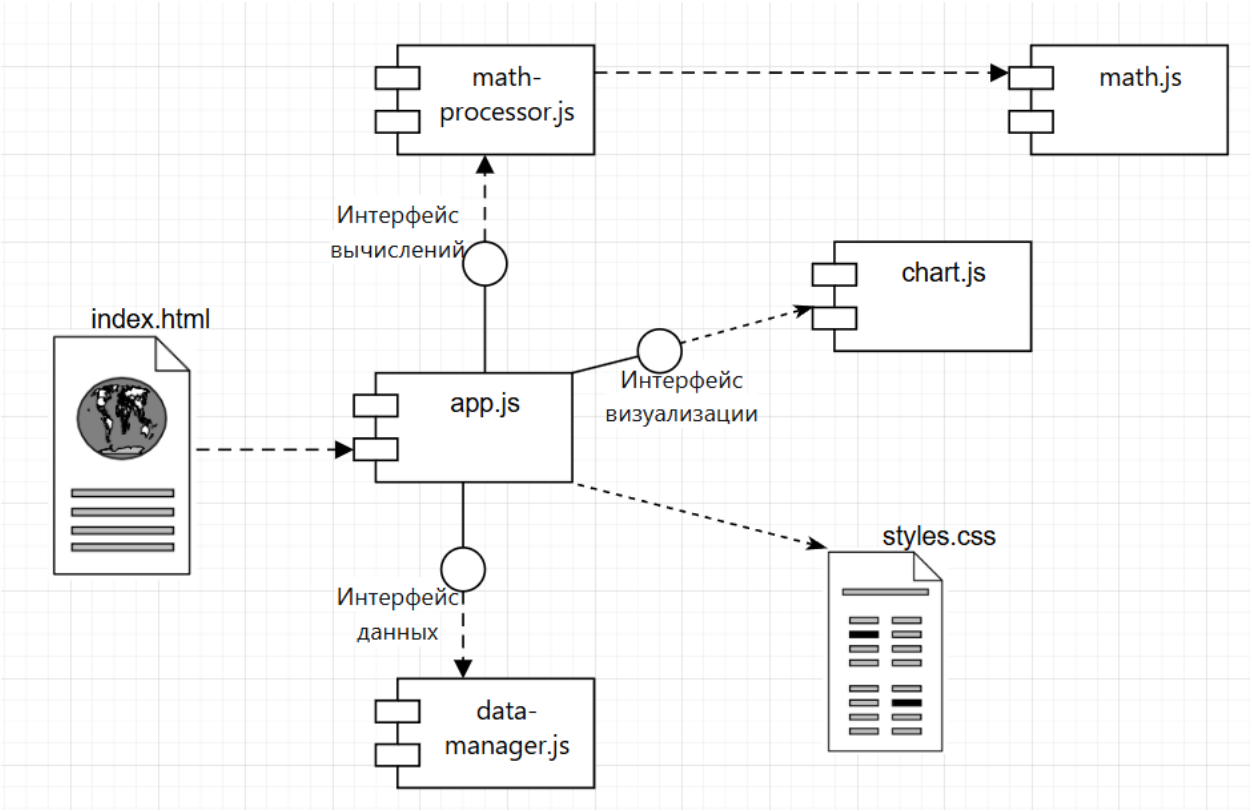

Рисунок 2.2 - Диаграмма компонентов UML

Главным компонентом является app.js - он управляет работой всей программы. Когда пользователь работает с интерфейсом (index.html), app.js получает данные и распределяет задачи между другими модулями: data-manager.js обрабатывает введенные числа, math-processor.js выполняет расчеты, а chart.js рисует графики. Связи между компонентами показывают, кто с кем общается. Сплошные линии означают, что один компонент предоставляет возможности другому, а пунктирные - что компонент использует чьи-то готовые функции. Например, math-processor.js использует готовые математические функции из библиотеки math.js.

3. Диаграмма развертывания UML

Разработана диаграмма развертывания, изображенная на рисунке 3.1, которая показывает физическую архитектуру программного продукта и расположение его компонентов в среде выполнения.

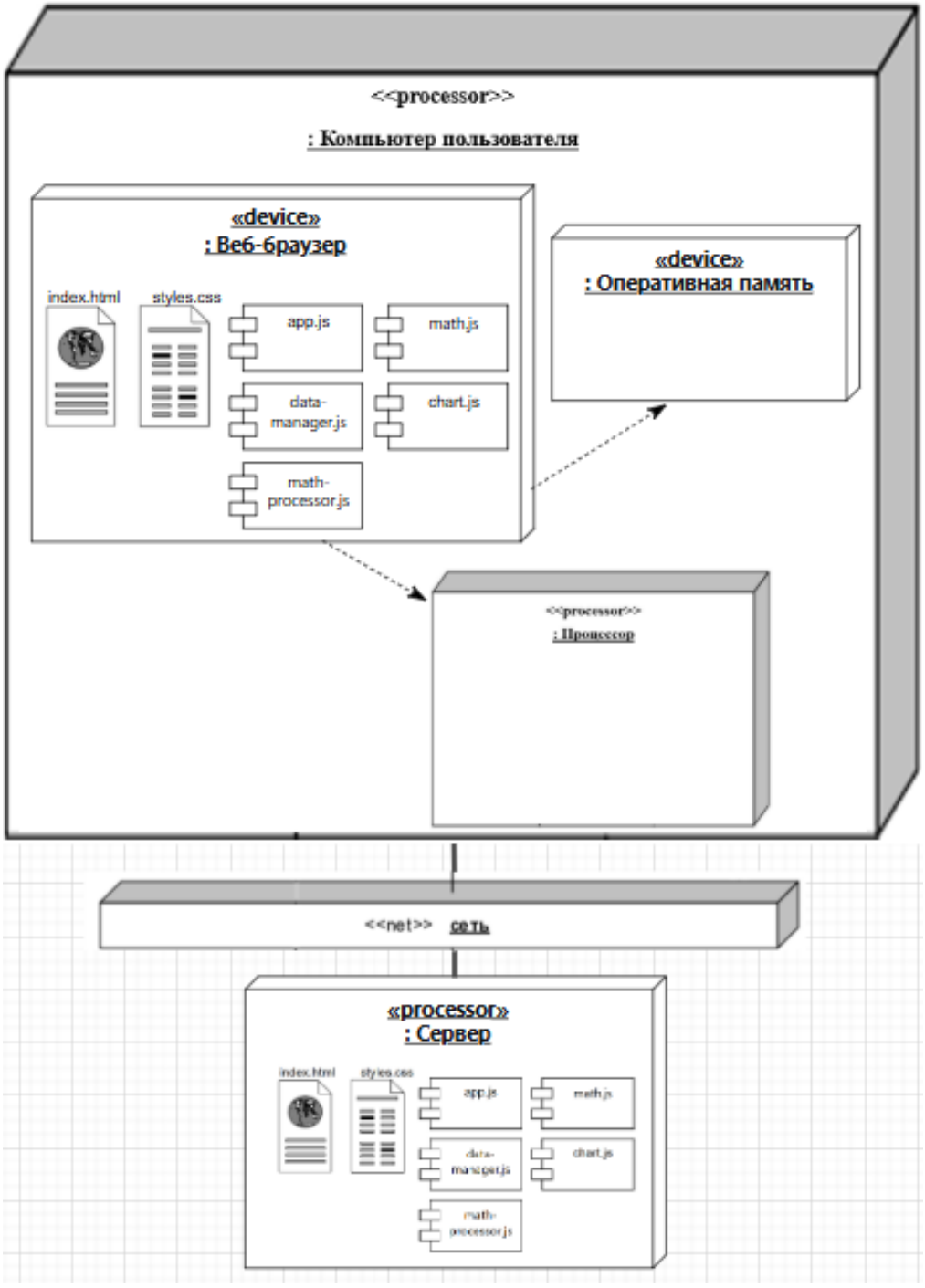

Рис 3.1 - Диаграмма развертывания UML

Система реализована по клиент-серверной модели, где сервер выполняет исключительно роль хранилища файлов, а все вычисления производятся на стороне клиента. Основным узлом является компьютер пользователя, содержащий веб-браузер для выполнения приложения,

процессор для математических расчетов и оперативную память для временного хранения данных.

При обращении к приложению веб-браузер загружает компоненты с сервера через сеть интернет, после чего программа работает полностью автономно. Все вычисления выполняются непосредственно в браузере с использованием JavaScript, что исключает необходимость в специализированном серверном оборудовании и обеспечивает безопасность данных пользователя.

Вывод: в ходе выполнения лабораторной работы №7 была успешно разработана архитектура программного продукта «Программа для аппроксимации функции». Разработаны основные структуры данных, оптимизированные для веб-среды и отражающие бизнес-логику приложения. Определен состав из программных компонентов с четким разделением ответственности. Построена диаграмма компонентов UML, демонстрирующая взаимодействие модулей через специализированные интерфейсы. Разработана диаграмма развертывания, показывающая клиент-серверную архитектуру с выполнением всех вычислений на стороне пользователя. Все проектные решения соответствуют техническому заданию и обеспечивают эффективное функционирование программного продукта. Архитектура обеспечивает минимальные требования к инфраструктуре, безопасность данных и высокую производительность за счет локального выполнения вычислений.