Санкт-Петербургский национальный исследовательский
университет информационных технологий, механики и оптики

Факультет прикладной информатики
Направление подготовки 11.04.02

Лабораторная работа №3
«Проектирование архитектуры инфокоммуникационной системы»

Выполнил:
Швалов Даниил Андреевич K4112c
Проверила:
Болгова Екатерина Владимировна

Санкт-Петербург
2025

СОДЕРЖАНИЕ

1 Введение

Цель работы: разработать и обосновать архитектуру прикладной инфокоммуникационной системы с учётом уровней, компонентов, взаимодействий и используемых стандартов, сформировать представление о подходах UML и C4 для визуального моделирования системной архитектуры.

Задачи:

1) разработать модель архитектуры согласно методологии Zachman Framework;

2) спроектировать архитектуру системы;

3) проанализировать архитектурные и проектные решения;

4) выполнить трассировку требований.

2 Ход работы

2.1 Разработка модели архитектуры согласно методологии Zachman Framework

Для разрабатываемой системы была матричную модель Zachman по методологии Zachman Framework, в которой система описана с разных точек зрения и уровней детализации. Данная матричная модель представлена в таблице 1.

Таблица 1 — Матрица Zachman

Уровень/ Вопрос	**Что?**	**Как?**	**Где?**	**Кто?**	**Когда?**	**Почему?**
Область	Система предоставляет доступ к различной информации о кафе и ресторанах	Система предоставляет функции просмотра и поиска кафе и ресторанов	Система размещена в ДЦ	Конечные пользователи, маперы, владельцы кафе и ресторанов	Система работает 24/7	Система должна уменьшить время поиска кафе или ресторанов, предоставлять аналитические

						данные для кафе или ресторана
Бизнес-модель	Система позволяет получить доступ к данным о блюдах, особенностях, режиме работы кафе и ресторанов	Система предоставляет возможности поиска и фильтрации кафе и ресторанов по различным критериям	Система размещена в ДЦ на нескольких виртуальных машинах	Конечные пользователи могут искать информацию о кафе и ресторанах, маперы могут вносить изменения, владельцы кафе и ресторанов могут получать аналитические данные	Система содержит актуальные данные о кафе и ресторанах	Система стремиться улучшить жизнь пользователей засчет оптимизации процесса поиска кафе и ресторанов
Системная модель	Система состоит из подсистемы хранения данных, подсистемы поиска, подсистемы аналитических данных	Подсистема хранения данных обеспечивает хранение и доступ к информации о кафе и ресторанах, подсистема поиска обеспечивает поиск кафе и ресторанов по	Виртуальные машины, на которых размещена система, согласуются с помощью системы	Конечные пользователи, маперы и владельцы получают доступ к системе с помощью интернет-приложения	Подсистемы работают 24/7	Подсистемы должны быть надежными, эффективными и отказоустойчивыми

		различным критериям, подсистема аналитических данных обеспечивает сбор и доступ аналитической информации	оркестрации			
Технологии	Система использует PostgreSQL для хранения данных	Система обеспечивает доступ к данным через HTTPS протокол	Для оркестрации виртуальных машин используется Kubernetes	Доступ к системе на чтение доступен без аутентификации, доступ для изменения данных доступен после аутентификации по OAuth	Используемые технологии не содержат критичных ошибок или уязвимостей	Используемые технологии должны быть надежными, эффективными и отказоустойчивыми
Детализация	Система использует транзакции при изменении данных	Для реализации серверной части используется фреймворк userver, для реализации клиентской части исполь-	Для разворачивания виртуальных машин в Kubernetes используются	Взаимодействие с системой производится с шифрованном виде	Функции системы работают 24/7 без критичных ошибок	Для системы должен обеспечивать надежный мониторинг и своевременное устранение проблем

		зуется фреймворк Next.js	Deployments			
Рабочая система	Ввод и вывод данных в нужном виде	Работающая система	Интернет	Конечные пользователи	Система работает 24/7	Уменьшение времени поиска кафе и ресторанов

2.2 Архитектурное проектирование

2.2.1 Постановка задачи

OpenRestaMap — это новая система, которая позволяет пользователям быстро и просто находить кафе и рестораны по различным критериям, таким как содержимое, состав, КБЖУ блюд и тому подобное. Контекстная диаграмма на рисунке 1 показывает внешние объекты и системные интерфейсы системы.

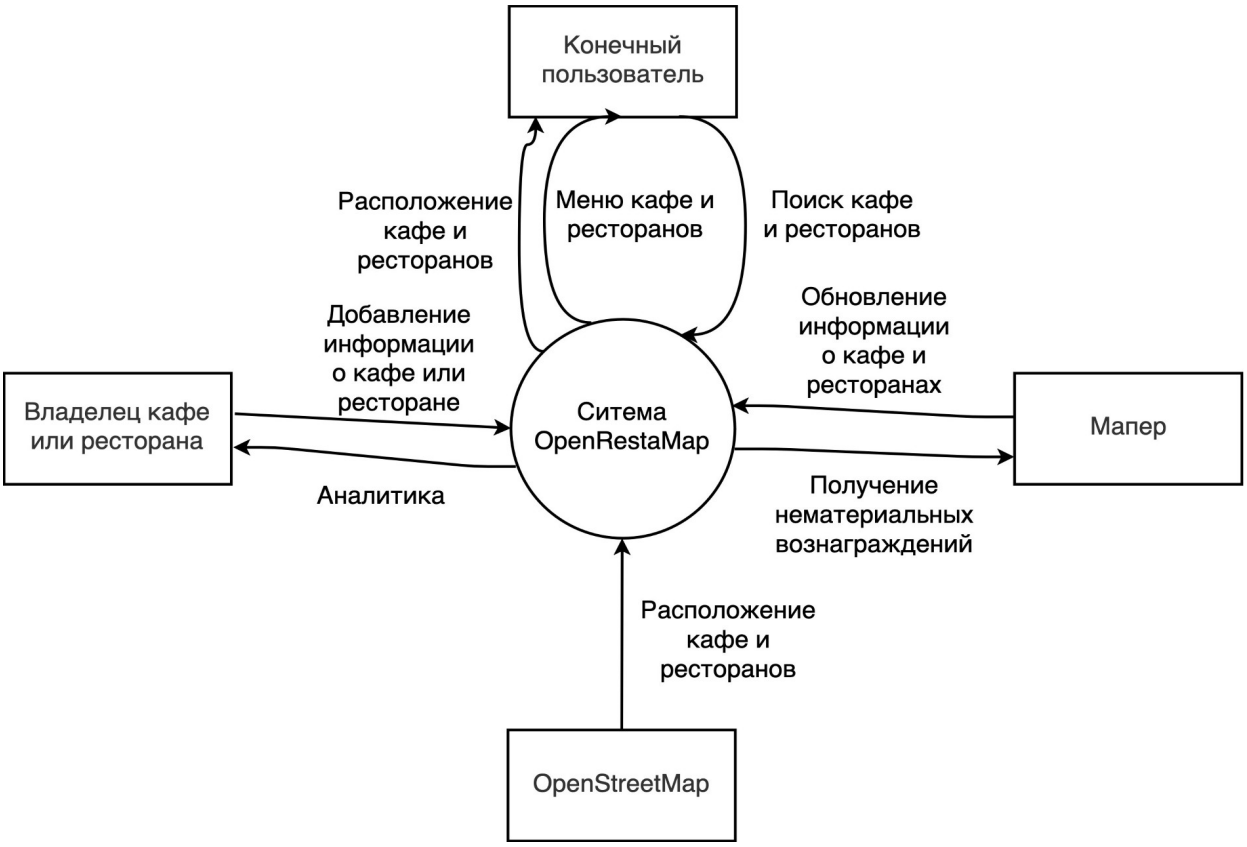

Рисунок 1 — Контекстная диаграмма системы OpenRestaMap

Система имеет следующие бизнес-цели:

1) сократить время поиска подходящего ресторана на 20% в течение 6 месяцев после первого выпуска системы;

2) увеличить полноту данных о кафе и ресторанах на 10% в течение 12 месяцев после первого выпуска системы.

Система обладает следующими функциями:

1) просмотр точек общественного питания на карте в определенной области;

2) поиск и фильтрация точек общественного питания по различным критериям, в том числе по содержимому блюд;

3) просмотр меню точек общественного питания, содержащего подробное описание блюд: их состав, КБЖУ, стоимость, фотографии;

4) изменение точек общественного питания;

5) просмотр истории изменения точек общественного питания;

6) поощрение пользователей за внесение изменений в меню точек общественного питания с помощью системы рейтинга и нематериальных вознаграждений;

7) аналитика популярности точек общественного питания.

Для разрабатываемой системы были выделены следующие классы пользователей:

1) конечный пользователь — пользователь, который использует систему для поиска кафе или ресторанов;

2) мапер — пользователь, который является активным участником сообщества, часто вносит изменения в систему на добровольной основе, дополняет и актуализирует данные;

3) владелец кафе или ресторана — пользователь, который заинтересован в продвижении своей точки общественного питания.

2.2.2 Определение компонентов

В системе присутствуют следующие логические компоненты:

1) каталог точек общественного питания — это база данных, в которой хранится информация о расположении и особенностях точек общественного питания;

2) каталог меню точек общественного питания — это база данных, в которой хранится подробная информация о блюдах, которые есть в точках общественного питания;

3) каталог пользователей — это база данных, в которой хранится информация о пользователях и их действиях;

4) картографическая подсистема — это подсистема, которая используется для отображения информации о точках общественного питания на географической карте;

5) подсистема поиска — это подсистема, которая используется при поиске точек общественного питания по различным параметрам;

6) аналитическая подсистема — это подсистема, которая собирает, агрегирует и визуализирует аналитические данные;

7) подсистема аутентификации — это подсистема, которая управляет процессом идентификации и аутентификации пользователей;

8) подсистема администрирования — это подсистема, которая используется для управления системой администраторами системы;

9) клиентское интернет-приложение — это приложение, с помощью которого конечные пользователи получают доступ к системе.

В системе присутствуют следующие физические компоненты:

1) PostgreSQL — СУБД, которая используется для хранения текстовых данных из каталогов;

2) S3 — объектное хранилище, которое используется для хранения бинарных данных (в т. ч. изображений);

3) ClickHouse — СУБД, которая используется для хранения аналитических данных;

4) MapLibre GL JS — библиотека, используемая для отображения точек

общественного питания на географических картах;

5) сервер поиска — это сервер, которые обслуживает пользовательские запросы поиска точек общественного питания;

6) сервер аналитики — это сервер, который собирает, агрегирует, обрабатывает и предоставляет аналитические данные;

7) сервер аутентификации — это сервер, который идентифицирует и аутентифицирует пользователей;

8) сервер администрирования — это сервер, с помощью которого администраторы системы могут менять различные внутренние настройки системы;

9) сервер клиентского интернет-приложения — это сервер, который предоставляет пользователям функционал интернет-приложения для взаимодействия с системой.

2.2.3 Уровневая архитектура

2.2.3.1 Функциональная архитектура

Для разрабатываемой системы были выделены следующие классы пользователей:

1) конечный пользователь — пользователь, который использует систему для поиска кафе или ресторанов;

2) мапер — пользователь, который является активным участником сообщества, часто вносит изменения в систему на добровольной основе, дополняет и актуализирует данные;

3) владелец кафе или ресторана — пользователь, который заинтересован в продвижении своей точки общественного питания.

Для данных классов пользователей были выделены следующие пользовательские истории:

1) как конечный пользователь, я хочу иметь возможность посмотреть кафе и рестораны на карте в определенной области, чтобы можно было найти ближайшее подходящее заведение;

2) как конечный пользователь, я хочу иметь возможность искать кафе и рестораны по различным критериям, в т. ч. по содержимому меню, чтобы найти для себя наиболее подходящее место;

3) как конечный пользователь, я хочу иметь возможность посмотреть меню кафе или ресторана, в котором будет содержаться информация о составе блюд, КБЖУ, фотографии, чтобы понимать, какие блюда есть в кафе или ресторане;

4) как мапер, я хочу иметь возможность вносить изменения в информацию о кафе или ресторане, чтобы иметь возможность внести свой вклад;

5) как мапер, я хочу иметь возможность просмотреть историю изменений кафе или ресторана, чтобы иметь возможность отменить изменения;

6) как мапер, я хочу получать нематериальные вознаграждения за внесение изменений в систему, чтобы повысить мотивацию вносить изменения;

7) как владелец кафе или ресторана, я хочу видеть аналитические данные о популярности моего кафе или ресторана, чтобы собирать различную статистику о моем заведении.

Исходя из вышеописанных пользовательских историй была сформирована диаграмма вариантов использования. Она представлена на рисунке 2.

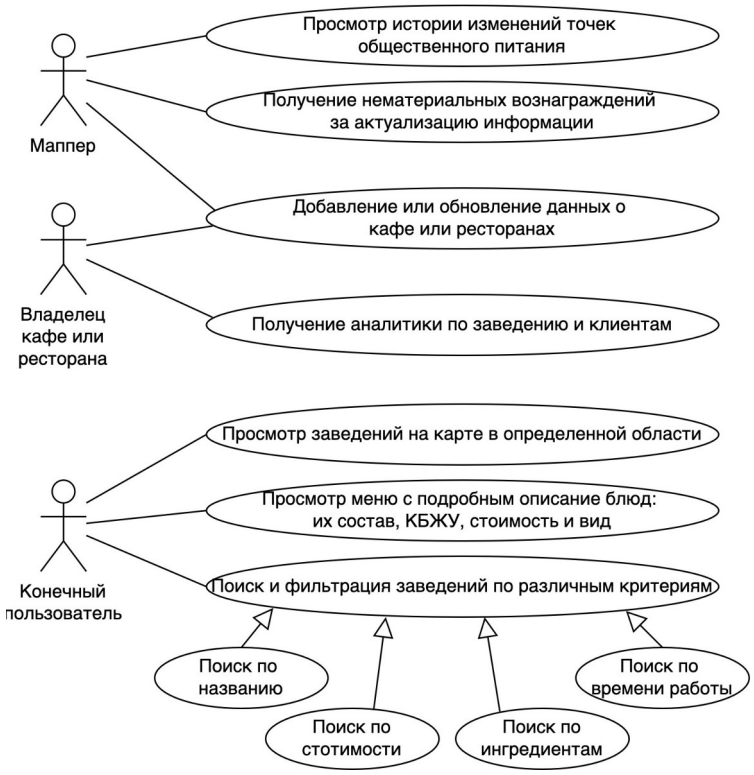

Рисунок 2 — Диаграмма вариантов использования

2.2.3.2 Логическая архитектура

Для системы была построена контекстная IDEF0, представленная на рисунке 3.

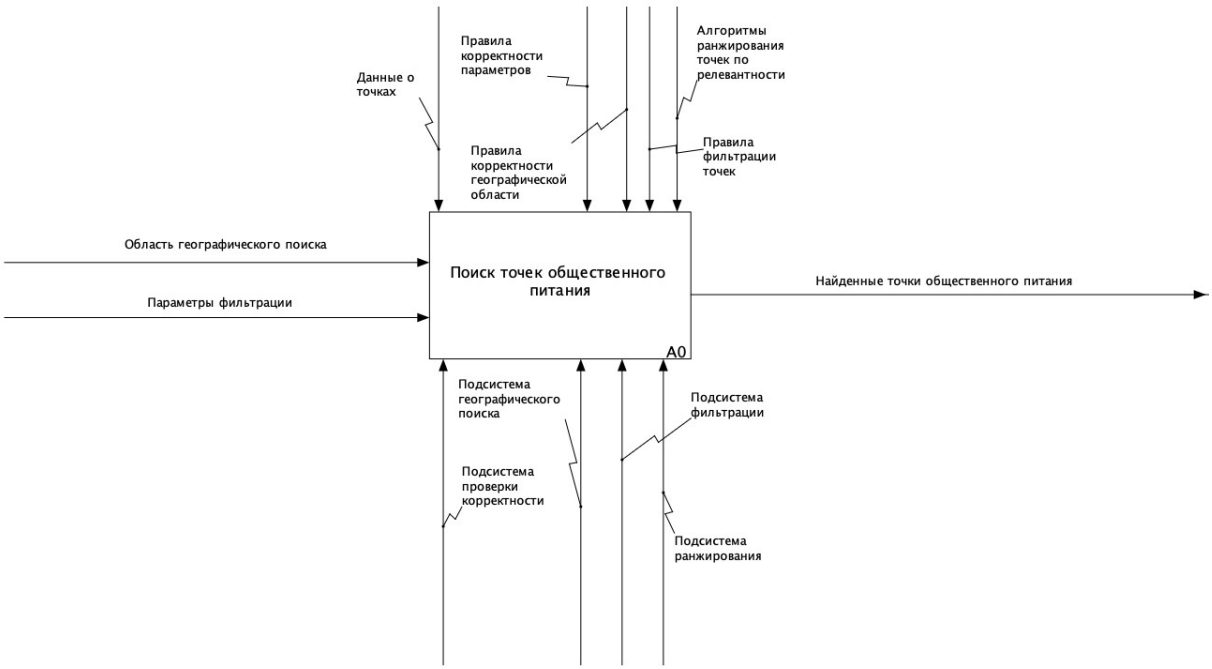

Рисунок 3 — Контекстная IDEF0 диаграмма

Данная диаграмма отражает процесс поиска точек общественного питания конечным пользователем. На ней в качестве входных данных представлены:

1) область географического поиска — это прямоугольная область, которую конечный пользователь выбирает в интерфейсе карты;

2) параметры фильтрации — это набор настроек и фильтров, которые конечный пользователь может дополнительно указать с помощью соответствующего меню.

Выходными данными является набор точек общественного питания, которые находятся в заданной области и соответствуют параметрам фильтрации.

В качестве потоков управления выступают:

1) данные о точках — это различная информация о точках общественного питания, которая используется при фильтрации и которая возвращается пользователю для отображения;

2) правила корректности параметров и географической области — это набор правил, который используется при проверке входных данных, переданных пользователем;

3) правила фильтрации точек — это набор правил, который используется при выборе точек по заданным конечным пользователем параметрам.

4) алгоритмы ранжирования точек по релевантности — это набор алгоритмов, которые используются для упорядочивания точек по различным критериям релевантности.

Для контекстной IDEF0 диаграммы была построена IDEF0 диаграмма декомпозиции 1-го уровня, представленная на рисунке 4.

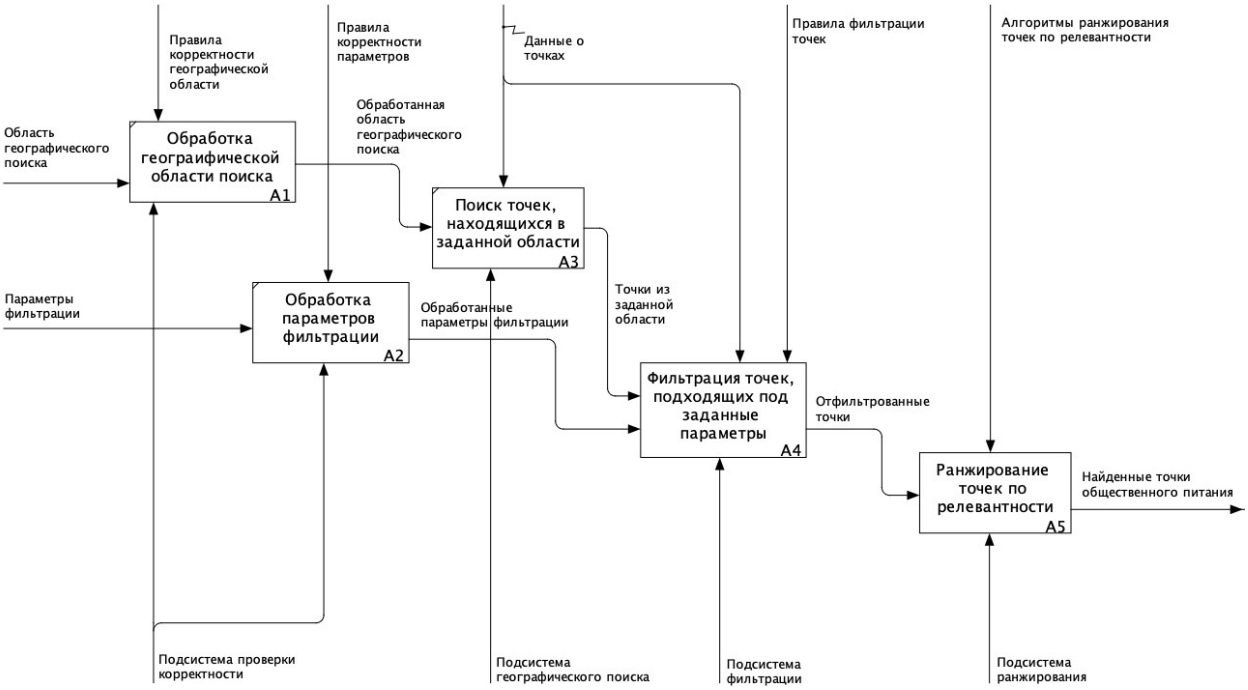

Рисунок 4 — IDEF0 диаграмма декомпозиции 1-го уровня

На IDEF0 диаграмме декомпозиции 1-го уровня представлены следующие функции:

1) обработка географической области поиска — это функция, которая проверяет информацию об области поиска на корректность и преобразует ее в понятный для дальнейших функций формат;

2) обработка параметров фильтрации — это функция, которая проверяет информацию о параметрах фильтрации на корректность и преобразует их в понятный для дальнейших функций формат;

3) поиск точек, находящихся в заданной области — это функция, которая использует ранее полученную область поиска и соотносит ее с имеющейся информацией о расположении точек, после чего возвращает пересечение этих множеств;

4) фильтрация точек, подходящих под заданные параметры — это функция, которая фильтрует точки, полученные на предыдущем шаге, в соответствии с переданными конечным пользователем параметрами фильтрации;

5) ранжирование точек по релевантности — это функция, которая сортирует точки по соответствию заданным конечным пользователем требованиям.

Для системы была построена диаграмма потоков данных. Она представлена на рисунке 5.

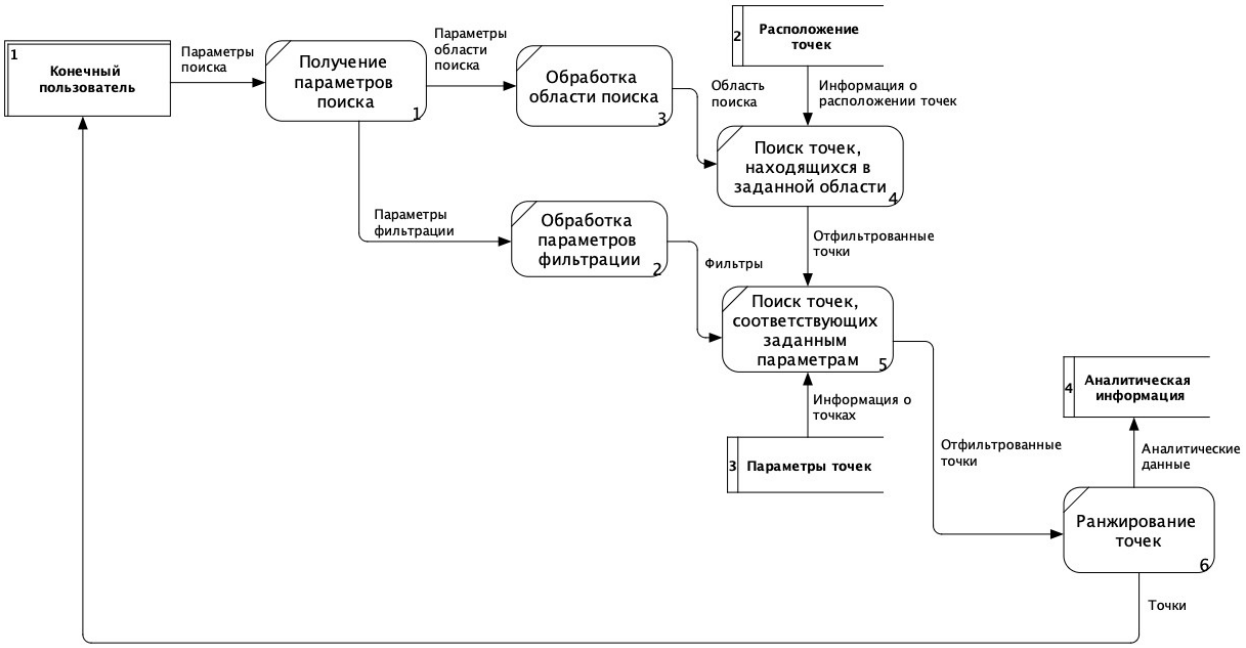

Рисунок 5 — Диаграмма потока данных поиска точек

На диаграмме потоков данных поиска точек в качестве активного объекта выступает конечный пользователь. В качестве процессов на диаграмме представлены:

1) получение параметров поиска — это процесс, который принимает информацию от конечного пользователя и передает его другим процессам, связанным с поиском;

2) обработка области поиска — это процесс, который проверяет информацию об области поиска на корректность и преобразует ее в понятный для дальнейших процессов формат;

3) обработка параметров фильтрации — это процесс, который проверяет параметры фильтрации на корректность и преобразует их в понятный для дальнейших процессов формат;

4) поиск точек, находящихся в заданной области — это процесс, который использует ранее полученную область поиска и соотносит ее с имеющейся информацией о расположении точек из соответствующего хранилища, после чего возвращает пересечение этих множеств;

5) фильтрация точек, подходящих под заданные параметры — это процесс, который фильтрует точки, полученные на предыдущем шаге, в соответствии с переданными конечным пользователем параметрами фильтрации;

6) ранжирование точек по релевантности — это процесс, который сортирует точки по соответствию заданным конечным пользователем требованиям, а также сохраняет аналитическую информацию в соответствующее хранилище.

На диаграмме также представлены следующие хранилища:

1) расположение точек — это хранилище, которое содержит информацию о географическом расположении точек и используется для поиска и фильтрации;

2) аналитическая информация — это хранилище, которое содержит различные аналитические данные и используется для составления аналитических отчетов.

2.2.3.3 Физическая архитектура

Для системы была построена физическая архитектура системы. Она представлена на рисунке 6.

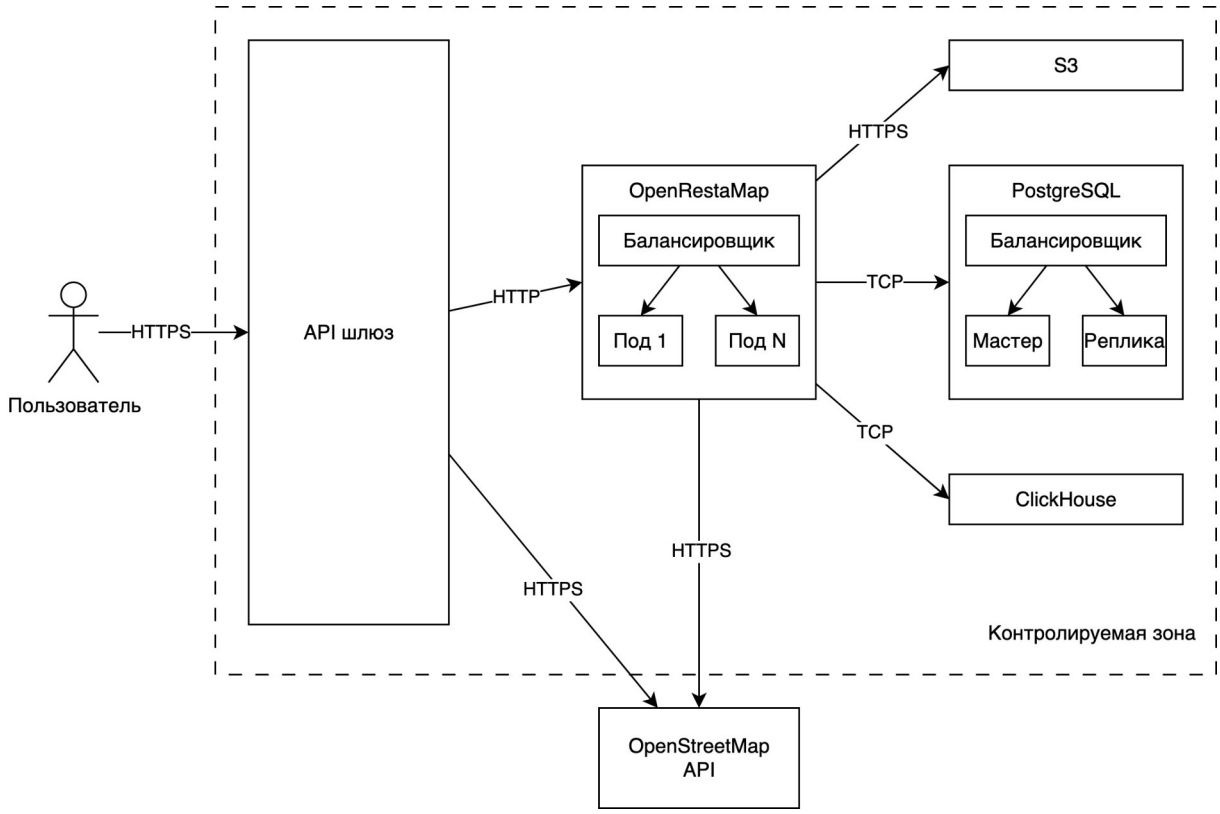

Рисунок 6 — Физическая архитектура системы

На диаграмме физической архитектуры системы приведены следующие компоненты системы:

1) API шлюз — это компонент, который предоставляет единую точку входа в систему для пользователя. Этот компонент маршрутизирует все запросы от пользователя в другие системы согласно заранее определенным правилам;

2) OpenRestaMap — это основной компонент системы, который содержит всю бизнес-логику;

3) S3 — это объектное хранилище, которое используется для хранения бинарных данных (например, изображений);

4) PostgreSQL — это СУБД, которая используется для хранения основной текстовой и числовой информации о точках общественного питания;

5) ClickHouse — это СУБД, которая используется для хранения

аналитических данных;

6) OpenStreetMap API — это API, которое используется для получения информации о расположении точек общественного питания из OpenStreetMap.

2.2.4 Взаимодействие компонентов

На рисунке 7 приведена системная диаграмма последовательности.

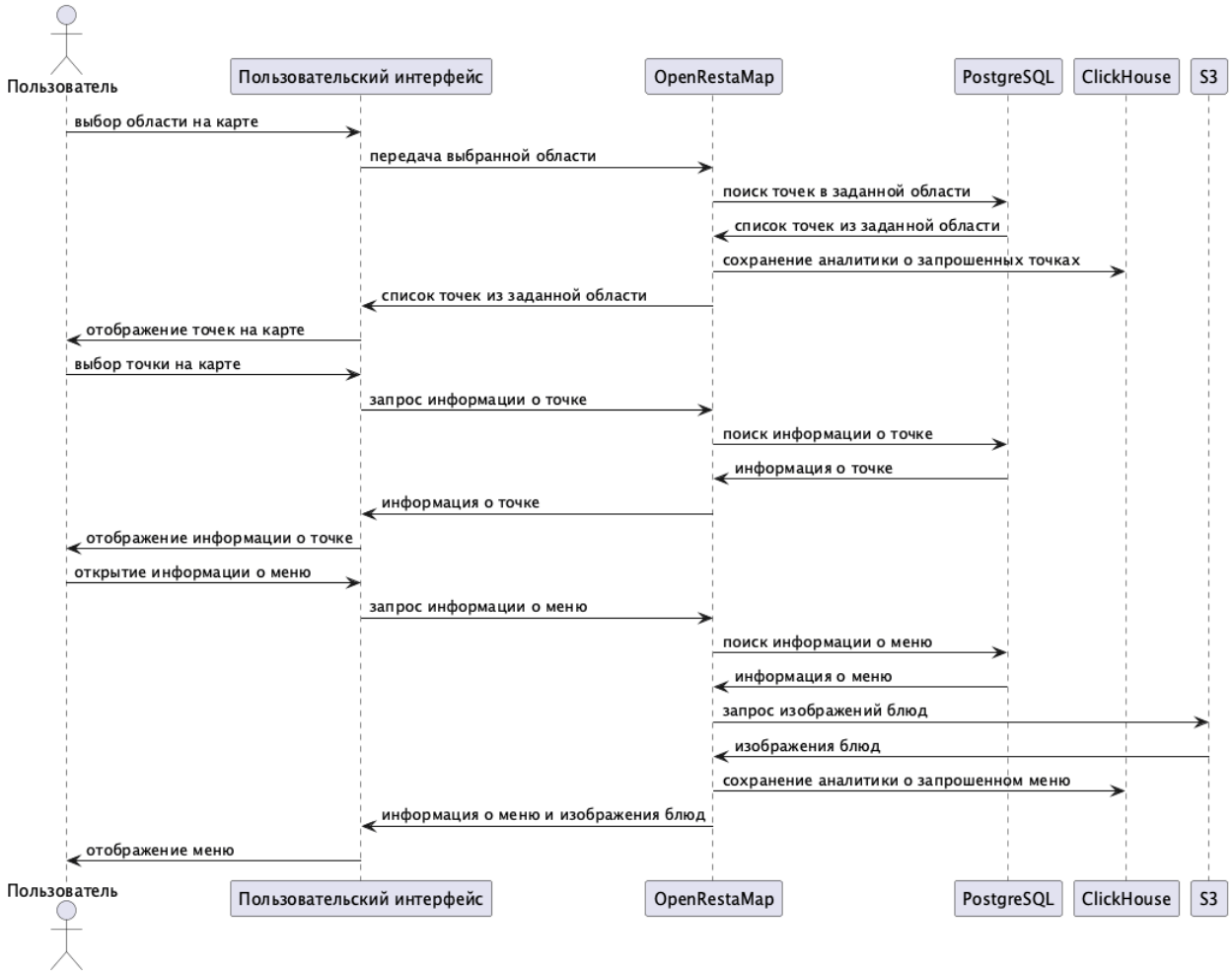

Рисунок 7 — Системная диаграмма последовательности

На системной диаграмме последовательности изображены следующие компоненты:

1) пользовательский интерфейс — это компонент, который предоставляет функции по отображению интерфейса пользователю, позволяет пользователю взаимодействовать с системой;

2) OpenRestaMap — это компонент, который содержит основную

бизнес-логику системы;

3) PostgreSQL — это компонент, который хранит всю основную текстовую и числовую информацию о точках, меню, блюдах;

4) ClickHouse — это компонент, в который сохраняется аналитическая информация о точках общественного питания;

5) S3 — это компонент, который хранит бинарные данные (например, изображения блюд).

На рисунке 8 представлена диаграмма компонентов системы.

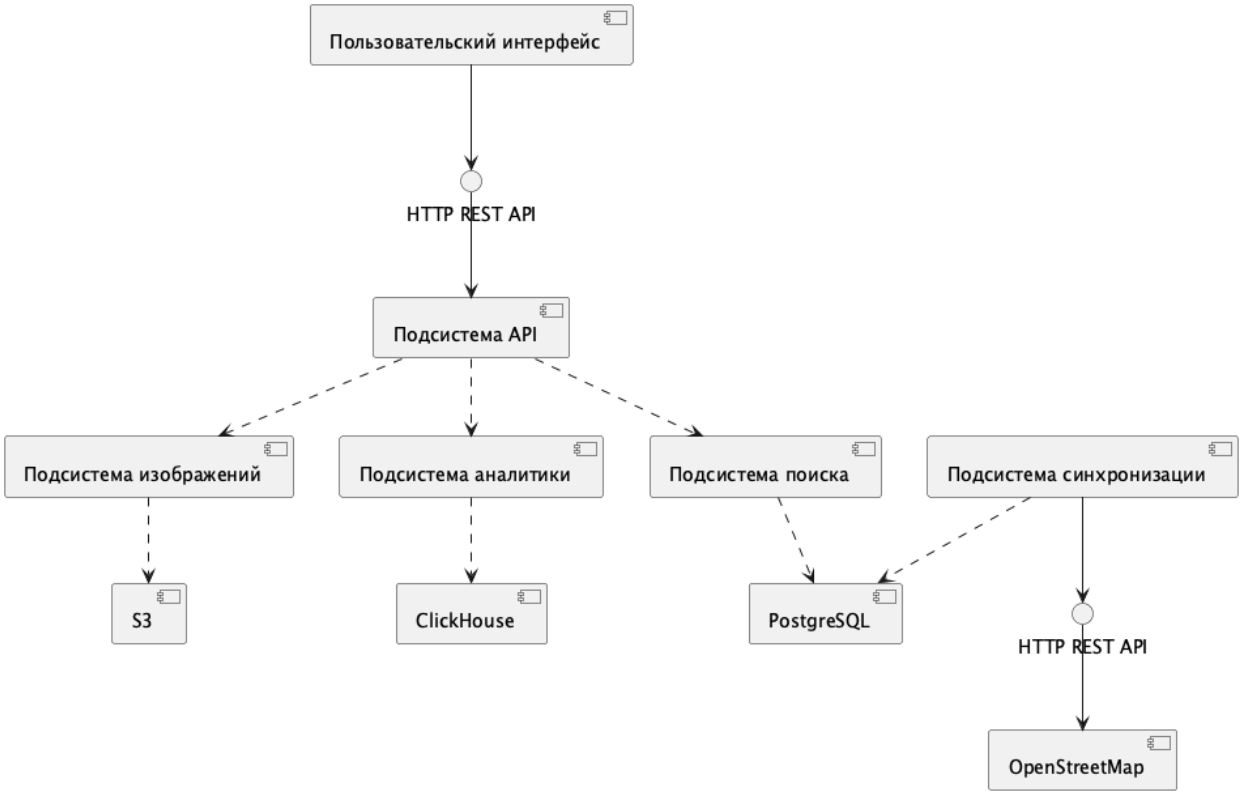

Рисунок 8 — Диаграмма компонентов

На диаграмме компонентов системы представлены следующие компоненты:

1) пользовательский интерфейс — это компонент, который предоставляет функции по отображению интерфейса пользователю, позволяет пользователю взаимодействовать с системой;

2) подсистема изображений — это компонент, который обрабатывает

запросы получения и загрузки изображений;

3) подсистема аналитики — это компонент, который обрабатывает запросы получения и обновления аналитических данных;

4) подсистема поиска — это компонент, который обрабатывает запросы поиска точек в заданной области по различным критериям;

5) подсистема синхронизации — это компонент, который синхронизирует расположение и данные о точках из OpenStreetMap;

6) S3 — это компонент, который хранит бинарные данные (например, изображения блюд);

7) ClickHouse — это компонент, в который сохраняется аналитическая информация о точках общественного питания;

8) PostgreSQL — это компонент, который хранит всю основную текстовую и числовую информацию о точках, меню, блюдах;

9) OpenStreetMap — это система, которая предоставляет информацию о расположении точек.

На рисунке 9 представлена диаграмма развертывания системы.

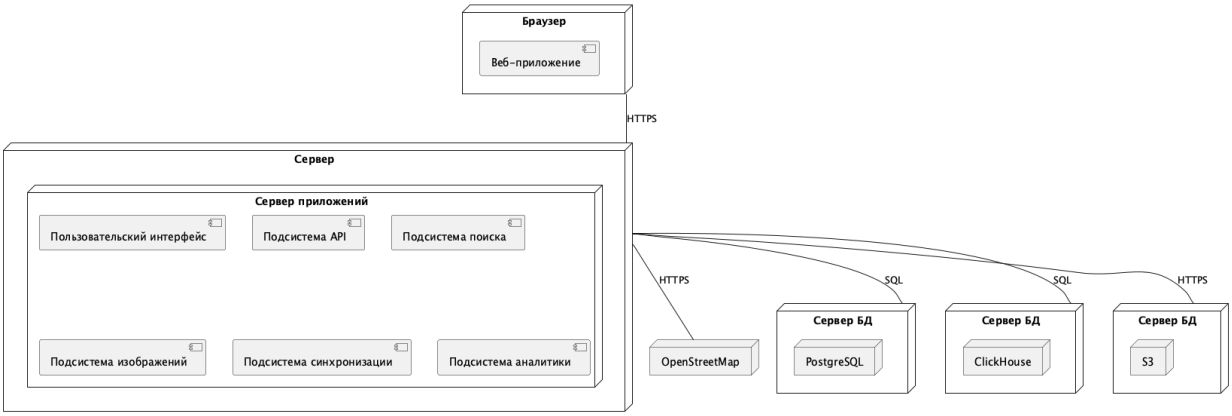

Рисунок 9 — Диаграмма развертывания

На рисунках 10-12 представлены диаграммы C4 с первого по третий уровни.

Рисунок 10 — Диаграмма C4 первого уровня

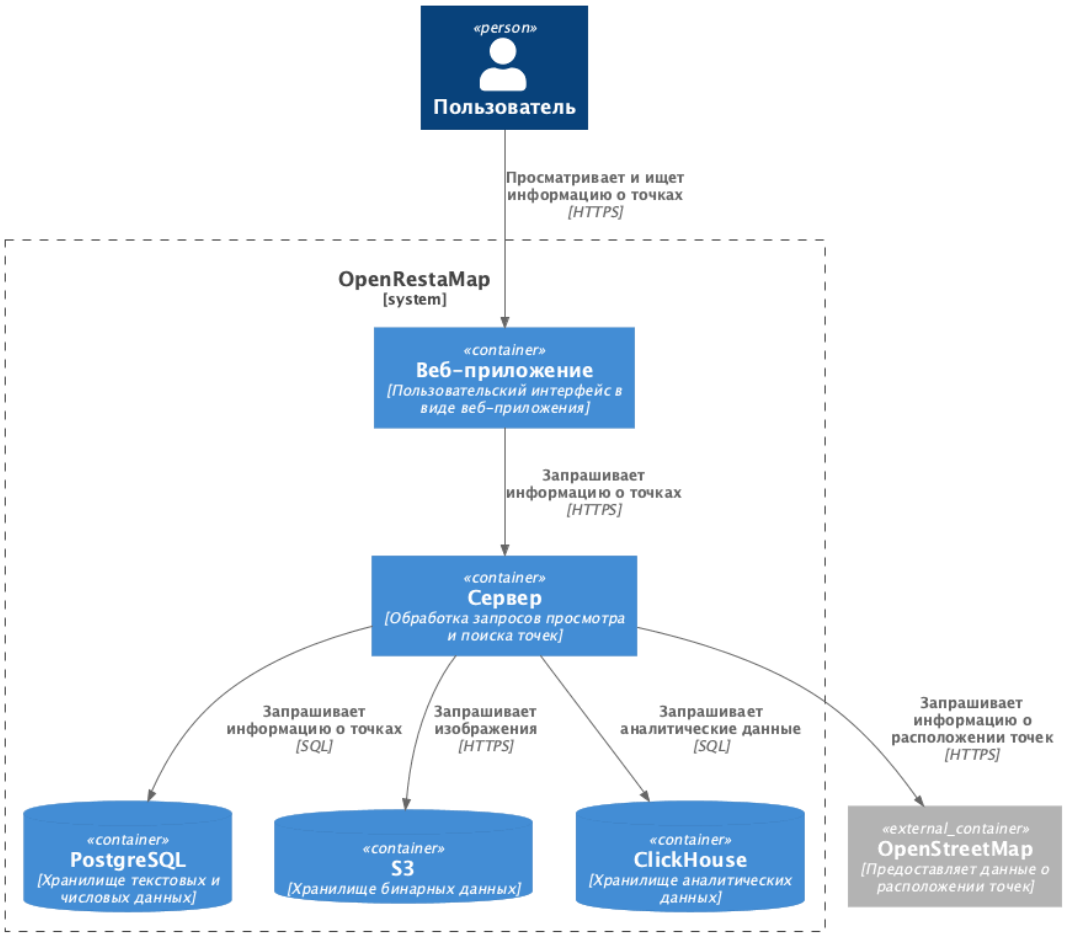

Рисунок 11 — Диаграмма C4 второго уровня

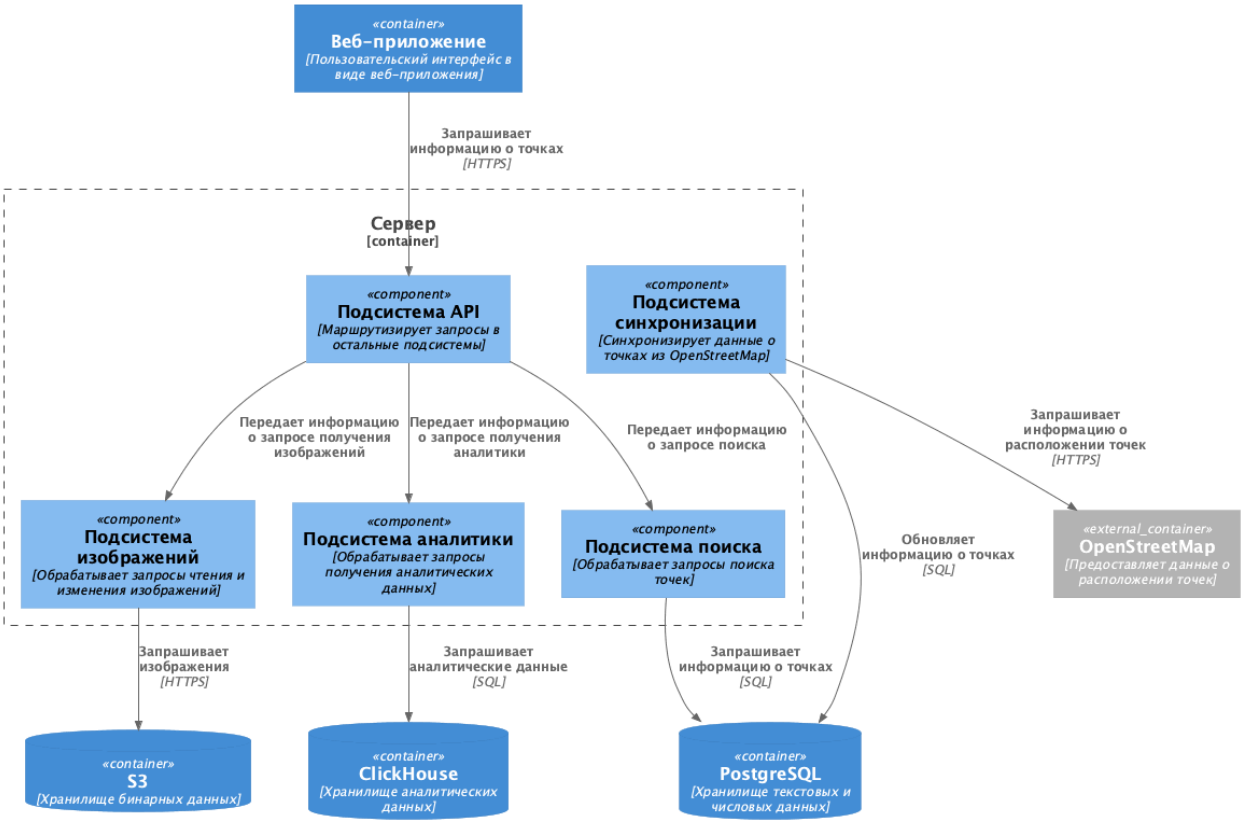

Рисунок 12 — Диаграмма C4 третьего уровня

UML и C4 сравнимы по следующим критериям:

1) уровень детализации: UML предоставляет больше инструментов для детализации системы, при этом C4 обладает достаточным количеством деталей для большинства задач проектирования;

2) понятность для разных ролей: UML подходит для построения диаграмм, нацеленных больше на технических специалистов, в то время как C4 предназначен для понимания как техническими специалистами, так и руководству;

3) простота создания и поддержки: C4 проще в освоении и поддержки, т. к. имеет меньше абстракций и визуальных элементов;

4) влияние на документацию и сопровождение проекта: C4 является модульным, из-за чего обновление разных уровней диаграмм может происходить независимо. UML более является более сложным языком, из-за чего поддерживать UML-диаграммы в актуальном виде значительно

сложнее.

C4 является достаточно детальной и при этом простой нотацией, поэтому в дальнейшем при возможности именно эта нотация будет использоваться при проектировании системы.

2.2.5 Технологический стек

Для развертывания системы будет использовать следующее программное обеспечение:

1) Ubuntu — открытый, бесплатный и самый популярный дистрибутив ОС GNU/Linux. Ubuntu является достаточно стабильным, используется многими компаниями, обладает богатым сообществом. Для Ubuntu существует большое количество руководств, а также документации, благодаря чему решение сопутствующих проблем происходит достаточно быстро;

2) Docker — программное обеспечение для развертывания приложений в контейнерах. Docker позволяет быстро и просто развернуть приложение практически в любой среде, поскольку благодаря Docker не приходится беспокоиться о том, что в системе не установлены нужные версии зависимостей. Это достигается засчет упаковывания приложения со всем своими зависимостями в контейнер. Docker на данный момент является стандартом индустрии разработки программного обеспечения, т. к. является надежным и производительным решением;

3) Kubernetes — программное обеспечение для оркестровки контейнеров, в т. ч. Docker-контейнеров, которое позволяет автоматизировать процессы развертывания, масштабирования и координации контейнеров. Kubernetes является стандартным средством для оркестровки контейнеров, т. к. является высокопроизводительным, гибким и надежным решением, которое используется во многих компаниях.

Для хранения данных будет использовать следующее программное обеспечение:

1) для числовых и текстовых данных — СУБД PostgreSQL, поскольку она является высокопроизводительной и надежной, обладает большим сообществом и имеет огромное количество руководств и документации. СУБД PostgreSQL используется во многих компаниях и зарекомендовала себя как надежное и производительное решение;

2) для аналитических данных — СУБД ClickHouse, поскольку она является зарекомендовавшим себя выбором для аналитических задач, где критичны скорость обработки запросов, масштабируемость и эффективность хранения, является открытой и бесплатной;

3) для бинарных данных — S3, поскольку оно обладает высокой масштабируемостью, гибкостью, производительностью и экономностью, не имеет архитектурных ограничений на объём данных, что позволяет хранить терабайты информации без необходимости менять инфраструктуру.

При разработке серверной части будут использоваться следующие технологии:

1) HTTPS — это протокол прикладного уровня, основанный на протоколе TCP, предназначенный для передачи различных данных, в т. ч. текстовых и бинарных файлов. HTTPS поддерживает шифрование и является безопасным решением для передачи информации между серверной и клиентской частей;

2) REST — это набор архитектурных шаблонов, которые позволяют организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и систему можно было просто масштабировать;

3) C++ — это язык программирования общего назначения, поддерживающий различные парадигмы программирования. Он позволяет создавать высокопроизводительные и масштабируемые программы, в т. ч. серверные приложения;

4) userver — это фреймворк для создания высокопроизводительных и

эффективных микросервисов на языке C++. Фреймворк обладает большим количеством готовых функций и шаблонов, которые позволяют быстро разрабатывает различную бизнес-логику.

При разработке клиентской части будут использоваться следующие технологии:

1) HTML — это язык гипертекстовой разметки, который используется для описания веб-страниц. HTML используется для создания структуры и содержания веб-страницы;

2) CSS — это язык для изменения внешнего вида веб-страницы, основанной на HTML. Он позволяет изменять различные параметры отображения блоков в HTML;

3) JavaScript — это основной язык программирования для создания интерактивных элементов в веб-браузере. Он позволяет создавать интерактивные элементы, которые могут изменять свой внешний вид или содержимое в зависимости от действий пользователя;

4) React — это библиотека для JavaScript, которая предлагает различные компоненты и функции для создания пользовательских веб-интерфейсов. Она позволяет ускорить процесс разработки, упростить веб-приложение, а также сделать его более масштабируемым;

5) Next.js — это фреймворк для JavaScript, основанный на библиотеке React, созданный для создания веб-приложений. Он содержит различные функции для реализации производительного и масштабируемого веб-приложения.

2.2.6 Безопасность

Для системы существуют следующие угрозы безопасности:

1) DDoS атаки;

2) атаки с массовым удалением или порчей данных;

3) инъекции кода.

Данные угрозы безопасности могут быть предотвращены следующими

способами:

1) использование межсетевого экрана для защиты от DDoS атак;

2) журналирование всех действий, хранение истории изменений;

3) использование систем обнаружения вторжений;

4) регулярное создание резервных копий;

5) использование концепции нулевого доверия;

6) валидация доступов и данных на всех уровнях системы.

2.2.7 Обоснование архитектуры

Архитектура системы приближена к монолитной архитектуре. Благодаря этому на первоначальном этапе разработка, тестирование и развертывание системы происходит просто, быстро и дешево, по сравнению с микросервисной архитектурой. На данном этапе развития системы монолитная архитектура полностью справляется с поставленными задачами. Микросервисная архитектура же может привнести сложности с развертыванием системы.

Однако с течением времени система может стать настолько большой, что ее будет сложно поддерживать и развивать, она может перестать отвечать требованиям надежности. В таком случае монолитную архитектуру необходимо будет сменить на микросервисную.

2.3 Анализ архитектурных и проектных решений

В таблице 2 приведен анализ архитектурных и проектных решений.

Таблица 2 — Анализ архитектурных и проектных решений

Решение	**Архитектурное или проектное**	**Обоснование**
Выбор монолитной архитектуры	Архитектурное	Стратегический выбор стиля
Выбор PostgreSQL	Архитектурное	Выбор технологии на уровне компонентов
Выбор S3	Архитектурное	Выбор технологии на уровне компонентов

Выбор ClickHouse	Архитектурное	Выбор технологии на уровне компонентов
Использование REST в API	Архитектурное	Способ взаимодействия
Шифрование данных	Архитектурное	Общее решение безопасности
Репликация данных	Архитектурное	Общее решение отказоустойчивости
Балансировка нагрузки	Архитектурное	Общее решение отказоустойчивости
Авторизация через JWT	Архитектурное (может быть на границе)	Общее решение безопасности
Передача данных через HTTPS	Архитектурное (может быть на границе)	Способ взаимодействия
Использование фреймворка userver для реализации серверной части	Проектное	Конкретная реализация серверной части
Использование шаблона Repository	Проектное	Конкретная реализация доступа к данным
Использование React.js для реализации пользовательского интерфейса	Проектное	Конкретная реализация пользовательского интерфейса
Использование OpenStreetMap для получения географических данным	Проектное	Конкретная реализация доступа к географическим данным
MapLibre GL JS в качестве картографического движка	Проектное	Конкретная реализация картографического движка
Rate Limiting для защиты от DDoS	Проектное	Конкретная реализация защиты от DDoS

Для данного анализа были даны ответы на следующие вопросы:

1) Почему выбор архитектуры нельзя менять в ходе проекта без значительных последствий, или все-таки можно?

Ответ: архитектуру системы в ходе проекта стоит изменять только в случае, если изменение архитектуры принесет больше пользы, чем проблем. На ранних стадиях проекта риски при изменении архитектуры ниже, тогда как на поздних этапах изменения могут быть крайне затратными и более рискованными. Перед изменением архитектуры нужно убедиться, что нет способа решить проблему с помощью доработки существующей архитектуры.

2) Какие проектные решения зависят от архитектурных?

Ответ: использование фреймворка userver зависит от передачи данных через HTTPS.

3) Какие проектные решения можно адаптировать в разных архитектурах?

Ответ: использование фреймворка userver, использование Rate Limiting для защиты от DDoS можно использовать как для монолитной, так и для микросервисной архитектуры.

4) Какие решения показались вам пограничными, почему?

Ответ: следующие решения показались пограничными: авторизация через JWT, передача данных через HTTPS. Они показались пограничными потому, что эти решения непосредственно влияют как на архитектуру системы в целом, так и на реализацию функциональных элементов.

2.4 Трассировка требований

В таблице 3 приведена матрица трассировки требований.

Таблица 3 — Матрица трассировки требований

Бизнес-требования	**Функциональные требования**						
	Просмотр информации о точке	Поиск точек	Просмотр меню	Изменение точек	Просмотр истории изменений	Нематериальные вознаграждения	Аналитика по точкам
Просмотр информации о точках	X						
Поиск точек	X	X					
Просмотр меню	X		X				
Изменение точек	X		X	X			
Просмотр истории изменений	X		X		X		
Нематериальные вознаграждения						X	
Аналитика по точкам	X						X

3 Вывод

В ходе выполнения данной лабораторной работы была разработана и обоснована архитектура прикладной инфокоммуникационной системы с учётом уровней, компонентов, взаимодействий и используемых стандартов,. сформировано представление о подходах UML и C4 для визуального моделирования системной архитектуры.