|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА** – **Российский технологический университет»**  **РТУ МИРЭА** |

Институт информационных технологий (ИТ)  
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ** | |
| **по дисциплине** | |
| «Проектирование клиент-серверных систем»  **на тему**  «Информационная система Электронные визитки» | |
|  | |
| Выполнил студент группы ИКБО-20-21 | Мухаметшин А.Р. |
| Принял | Мельников Д.А. |

**Оглавление**

[Введение 3](#_Toc179188943)

[Практическая работа № 1. 4](#_Toc179188944)

[Практическая работа №2 8](#_Toc179188945)

[Практическая работа №3 11](#_Toc179188946)

[Практическая работа №4 13](#_Toc179188947)

[Практическая работа №5 15](#_Toc179188948)

[Практическая работа №6 20](#_Toc179188949)

[Практическая работа №7 24](#_Toc179188950)

[Практическая работа №8 28](#_Toc179188951)

[Практическая работа №9 34](#_Toc179188952)

[Практическая работа №10 35](#_Toc179188953)

[Практическая работа №11 36](#_Toc179188954)

[Практическая работа №12 37](#_Toc179188955)

[Список использованных источников 38](#_Toc179188956)

# Введение

Современный темп жизни с каждым годом ускоряется, что заставляет людей искать более удобные и быстрые способы взаимодействия с окружающим миром. Одним из ярких проявлений этого тренда является стремление к цифровизации большинства аспектов повседневной жизни, включая деловые контакты. Визитные карточки долгое время оставались важным элементом деловой коммуникации, однако с развитием технологий они постепенно переходят в электронный формат, предлагая больше удобства и функциональности.

На сегодняшний день существует множество цифровых платформ и сервисов, которые упрощают ведение бизнеса и управление контактами. Электронные визитные карточки — это современное и удобное решение, позволяющее быстро обмениваться контактной информацией, интегрировать её в цифровую среду и предоставлять дополнительные возможности, такие как ссылки на соцсети или портфолио.

Информационная система “Электронные визитки” призвана автоматизировать процесс создания, обмена и управления цифровыми визитками. Она позволяет пользователям создавать индивидуальные электронные визитные карточки с уникальным набором данных, а также делиться ими через различные цифровые каналы. Такой подход не только экономит время, но и способствует более эффективной и организованной работе с контактами, исключая необходимость ручного ввода данных и их обновления.

Целью данного проекта является разработка системы для создания и использования электронных визиток, которая позволит пользователям легко создавать, хранить и обмениваться визитными карточками, а также вести их актуализацию. Для этого необходимо провести анализ предметной области, спроектировать программный модуль, выбрать подходящие технологии и разработать программное решение, которое будет удовлетворять потребности пользователей в быстром и удобном обмене контактной информацией.

# Практическая работа № 1.

**Цель работы:**

1. Знакомство с графической нотацией формализации и описания бизнес-процессов IDEF0. Знакомство c понятием функциональной модели AS-IS («как есть»).

2. Описание и построение функциональной модели AS-IS выбранной предметной области с применением нотации IDEF0.

**Постановка задачи:**

Для заданной предметной области разработать модель AS-IS. Вы можете выбрать один из вариантов процессов, описанных в приложении, или предложить свой вариант.

**Ход работы:**

Была спроектирована контекстная диаграмма А0 в нотации IDEF0

В качестве входа по управлению были выбраны:

* Закон о персональных данных
* Внутренние правила компании
* Шаблоны визиток

В качестве входящих потоков были выбраны:

* Данные пользователей
* Контактная информация для визитки
* Запрос на поиск визитки

В качестве механизмов используются:

* Пользователь
* База данных
* Сервер

В качестве выходов после выполнения ИС получены:

* Электронная визитка
* Данные о визитках пользователей

Сама контекстная диаграмма процесса ИС Электронный журнал школьника представлена на рисунке 1.

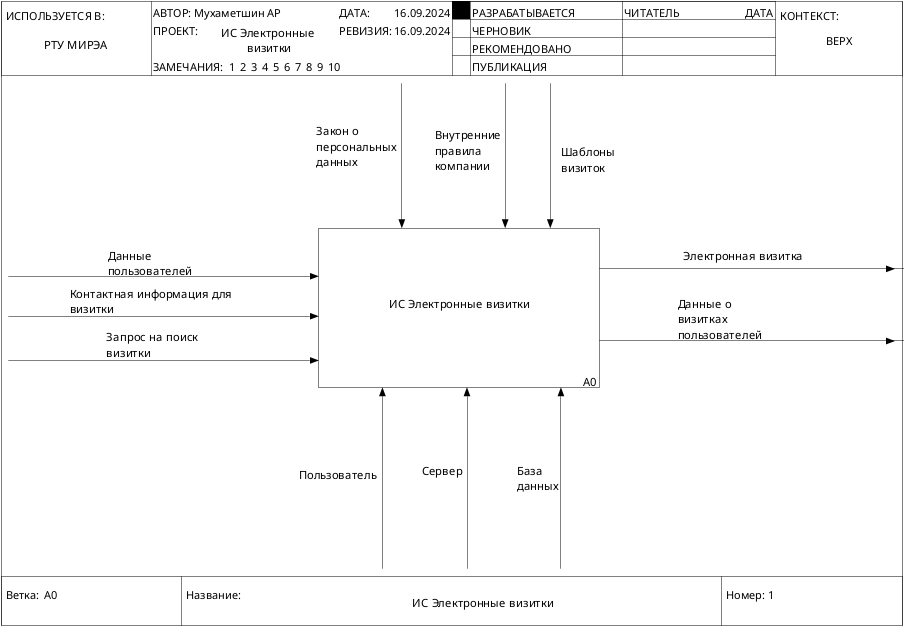


Рисунок 1 – Контекстная диаграмма А0

Далее была произведена декомпозиция основного функционального блока А0 (рисунок 2). Были получены следующие функциональные блоки:

* Регистрация пользователя – А1
* Создание визитки – А2
* Поиск и просмотр визиток других пользователей – А3

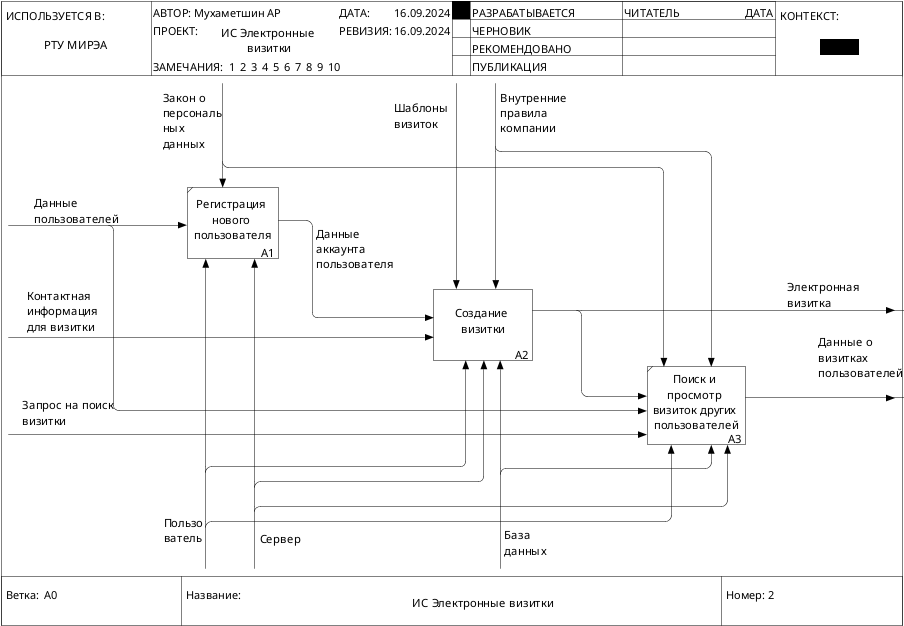


Рисунок 2 – Декомпозиция функционального блока

Для третьего уровня декомпозиции был выбрал блок А2. Были получены следующие блоки:

* Проверка введенных данных – А21
* Генерация визитки– А22
* Сохранение визитки – А23

Сама диаграмма представлена на рисунке 3.

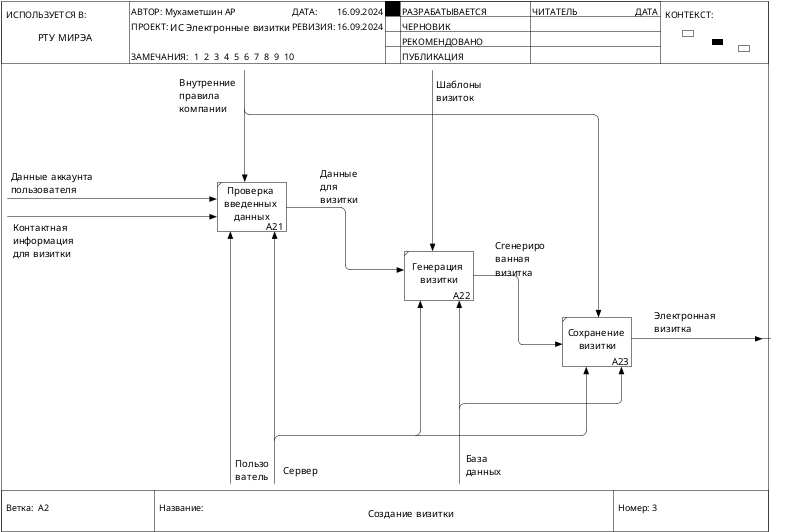


Рисунок 3 – Декомпозиция блока А3

**Вывод:**

В результате выполнения практической работы были получены теоретические знания в области диаграммы AS-IS, а также была разработана эта диаграмма в нотации IDEF-0.

# Практическая работа №2

**Цель работы:**

1. Знакомство с понятием функциональной модели TO-BE («как будет»).
2. Доработка созданной модели AS-IS с учетом выявленных недостатков в организации бизнес-процессов.

**Постановка задачи:**

Для заданной предметной области преобразовать созданную модель AS-IS в модель TO-BE. Внедрив информационную систему или клиент-серверную архитектуру.

**Ход работы:**

На основе анализа модели AS-IS были выявлены следующие улучшения для создания модели TO-BE:

1. Добавление функции автоматического обмена визитками через QR-код.
2. Синхронизации визиток с внешними сервисами (например, LinkedIn).
3. Введение аналитической панели для статистики использования визиток.

Контекстная диаграмма (A0) включает в себя следующие новшества:

Входы:

* Данные о синхронизации с внешними сервисами

Механизмы:

* Дашборд аналитики

Выходы:

* Отчет по использованию визиток
* Рекомендации по улучшению визитки

Функциональный блок А2 был преобразован в «Создание и синхронизация визитки», и был добавлен функциональный блок «Аналитика использования визиток». Так же была обновлена декомпозиция блока А2. На рисунках 4-6 показан результат произведенной работы.

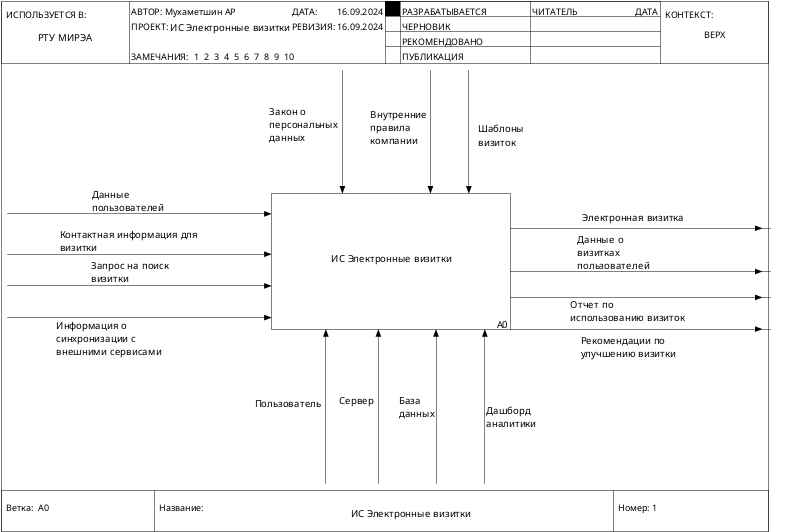


Рисунок 4 – Обновленный блок А0

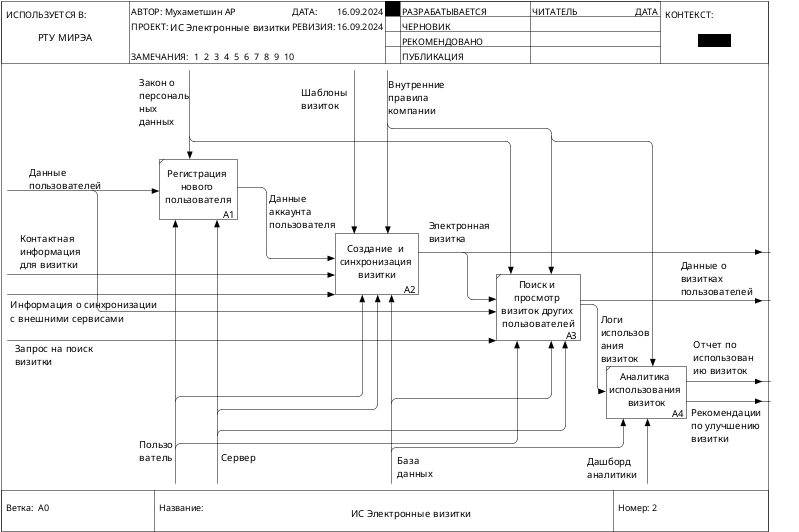


Рисунок 5 – Обновленная декомпозиция блока А0

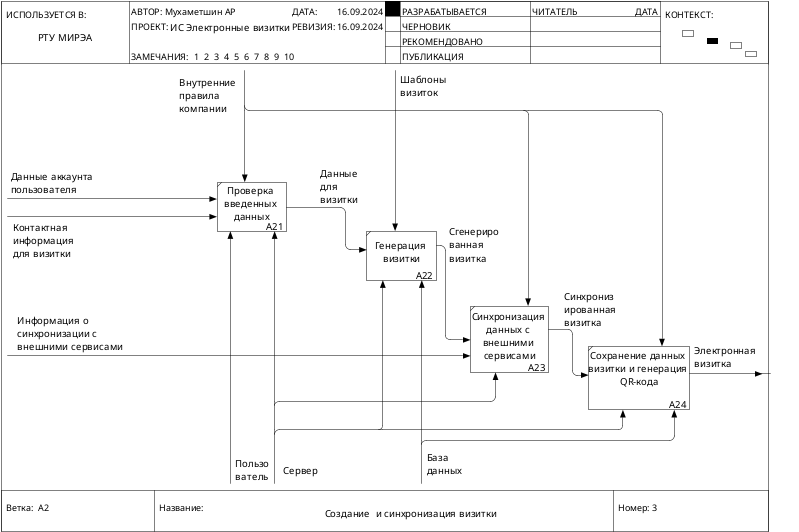


Рисунок 6 – Обновленная декомпозиция блока А2

**Вывод:**

В результате выполнения практической работы были получены теоретические знания в области диаграммы TO-BE, а также была разработана эта диаграмма в нотации IDEF-0.

# Практическая работа №3

**Цель работы:**

Получить практические навыки в построении IDEF3-модели бизнес-процесса.

**Постановка задачи:**

С помощью методологии IDEF3 декомпозировать 1 из функциональных блоков модели окружения (A0), используя все типы перекрестков.

**Ход работы:**

Изначально была установлена программа Erwin Process Modeller для построения IDEF-3 модели бизнес-процесса ИС Электронные визитки. Первый уровень декомпозиции контекстной диаграммы А0 включает в себя 5 функциональных блоков:

* Регистрация пользователя
* Создание визитки
* Просмотр и обновление визитки
* Аналитика использования визитки
* Сохранение изменений

Сами диаграммы представлены на рисунках 7-8.

Из рисунка видно, что в декомпозиции использованы все три перекрестка, что соответствует заданию. После входа может запуститься только один следующий процесс – Создание визитки, либо повторная регистрация после ошибки. Поэтому используется перекресток XOR.

После создания визитки происходит сохранение данных и переход на просмотр и обновление данной визитки при помощи перекрестка AND.

При просмотре и обновлении визитки есть возможность либо обновить визитку и сохранить изменения, либо просмотреть аналитику по данной визитке. Это реализуется при помощи перекрестка OR.

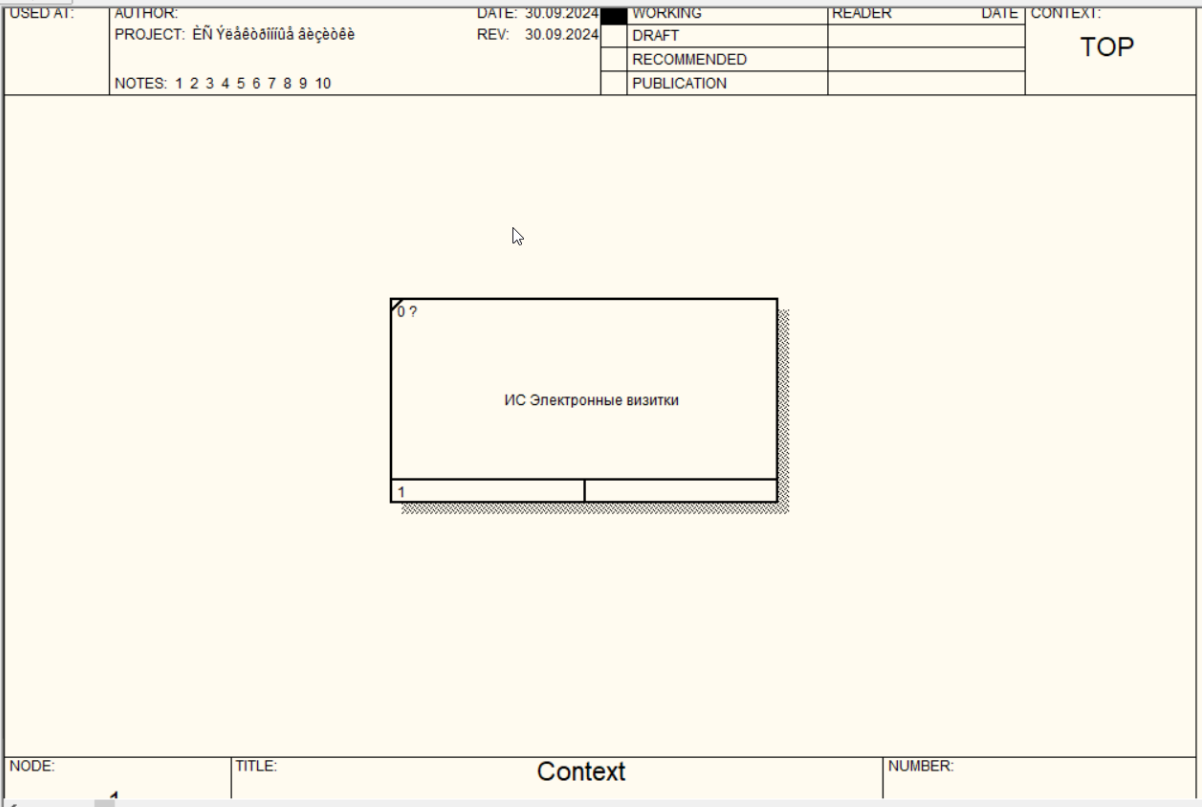


Рисунок 7 – Контекстная диаграмма А0

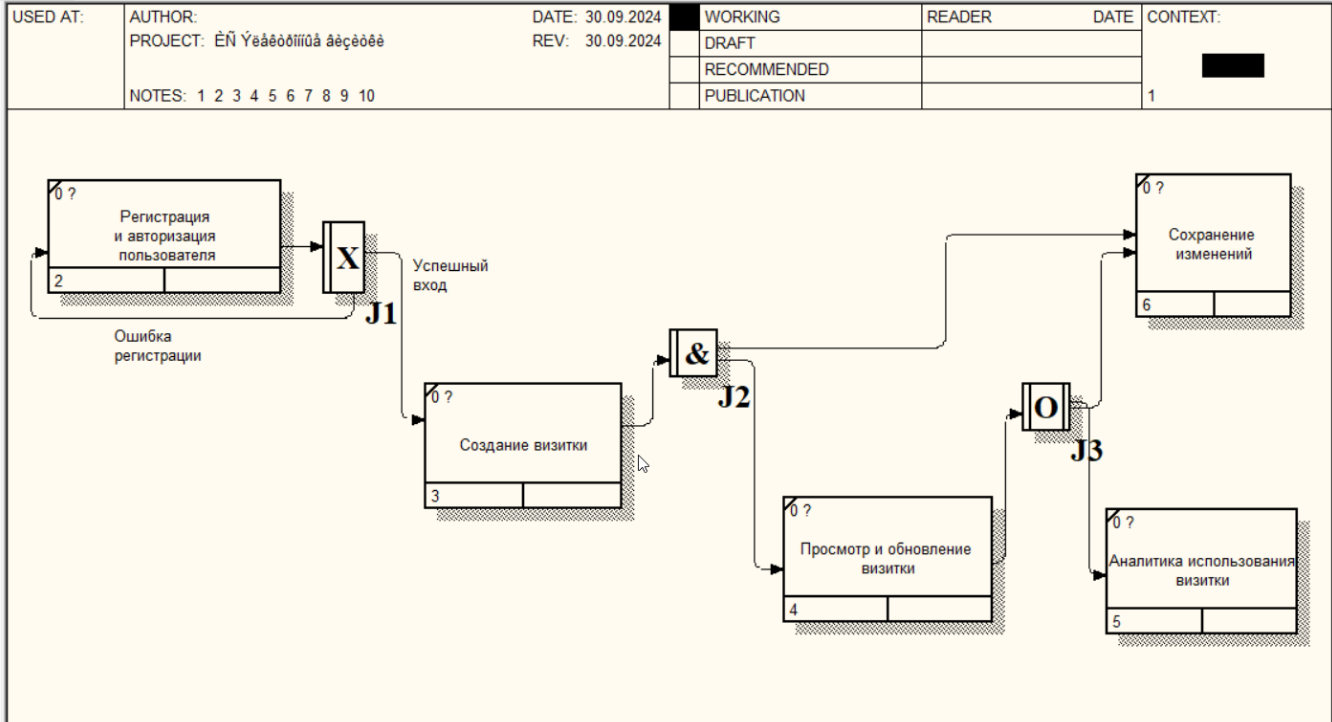


Рисунок 8 – Декомпозиция А0

**Вывод:**

В результате выполнения практической работы были получены теоретические и практические знания в области нотации IDEF-3. Были изучены и использованы все три типа перекрестков и построена контекстная диаграмма с одним уровнем декомпозиции.

# Практическая работа №4

**Цель работы:**

Получить практические навыки в построении DFD-модели бизнес-процесса.

**Постановка задачи:**

С помощью методологии DFD декомпозировать 1 из функциональных блоков. Можно выбрать часть процесса, который моделировался на предыдущих лабораторных работах. При выборе учтите, что процесс обязательно должен предусматривать обработку информации, лучше, чтобы это была автоматизированная обработка с использованием одной или нескольких информационных систем.

**Ход работы:**

Для декомпозиции процесса в нотации DFD было принято решение переделать уже существующую декомпозицию блока А2 «Создание и синхронизация визитки». Все подпроцессы остались такими же, как и были. Были добавлены :

**Хранилища данных**:

* База данных пользователей: хранит информацию о зарегистрированных пользователях.
* База данных визиток: хранит все созданные электронные визитки и данные о синхронизации с внешними сервисами.

**Внешние сущности**:

* Пользователь: взаимодействует с системой для регистрации, создания визитки, поиска и просмотра визиток.
* Внешние сервисы: получают данные о визитке и возвращают подтверждение синхронизации.

Измененная диаграмма представлена на рисунке 9.

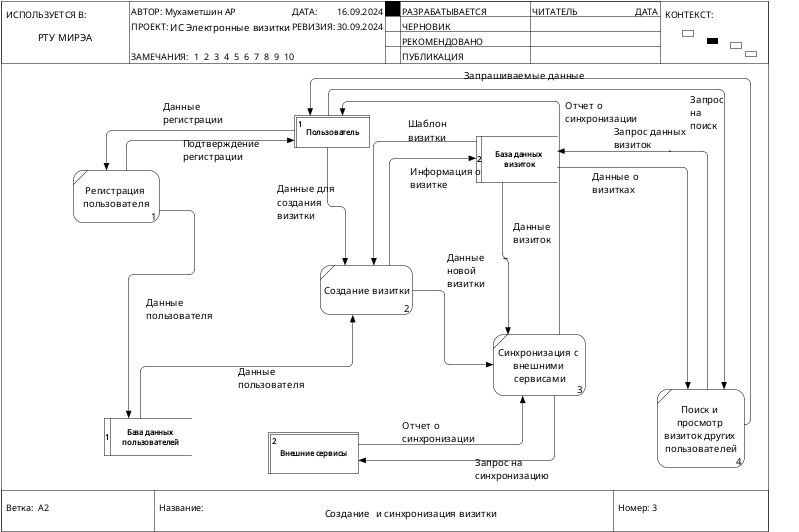


Рисунок 9 – Измененная декомпозиция блока А2 в нотации DFD

**Вывод:**

В результате выполнения практической работы, были получены теоретические и практические навыки в нотации DFD, были изучены основные термины – внешняя сущность и хранилище данных.

# Практическая работа №5

**Цель работы:**

Получить практические навыки в построении UML-моделей бизнес-процесса.

**Постановка задачи:**

Выберите бизнес-процесс для которого будете формировать модель. Можно выбрать один из процессов, для которых на предыдущих лабораторных работах строилась модель по одной из структурных методологий. Желательно, чтобы процесс имел различные версии, т.е. альтернативные потоки событий.

**Ход работы:**

Для информационной системы "Электронные визитки" были разработаны следующие UML-диаграммы:

**Диаграмма классов**:

Диаграмма классов включает ключевые сущности системы: Пользователь, Электронная визитка, Синхронизация и Аналитика. Пользователь создает визитки, каждая визитка может быть синхронизирована с внешними сервисами (например, LinkedIn). Аналитика предоставляет данные о просмотре визиток.

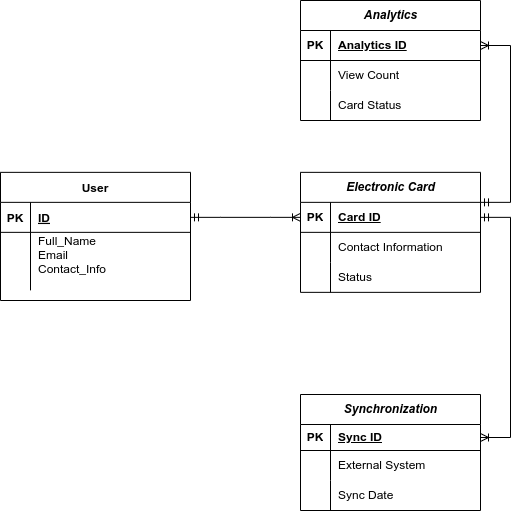


Рисунок 10 – диаграмма классов

**Диаграмма последовательности**:

Была разработана диаграмма последовательности поиска визитки по запросу.

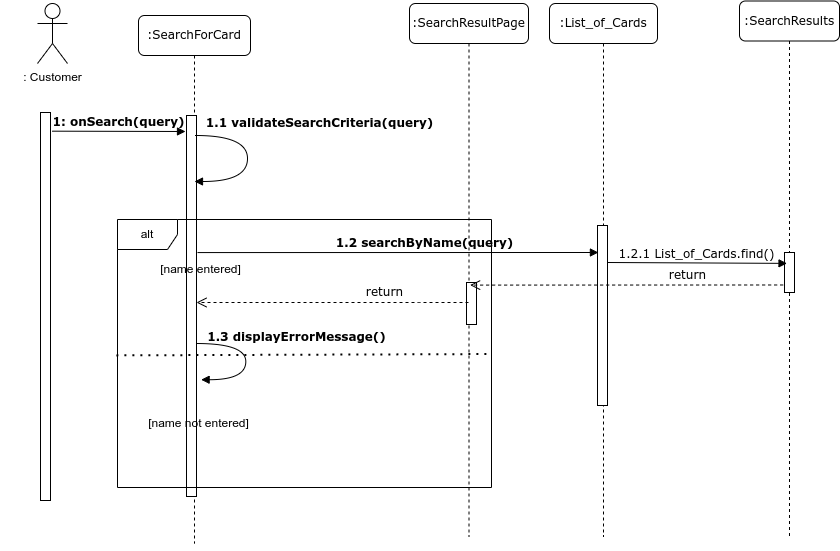


Рисунок 11 – диаграмма последовательности

**Диаграмма обзора взаимодействий**:

Диаграмма описывает процессы запроса синхронизации и аналитики визитки.

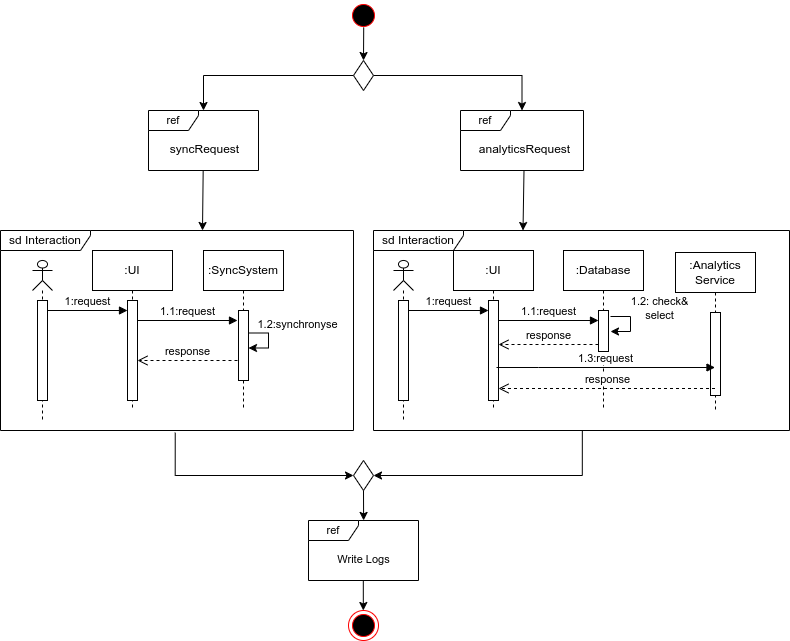


Рисунок 12 – диаграмма обзора взаимодействий

**Диаграмма пакетов**:

Построена диаграмма пакетов, где система разделена на модули.

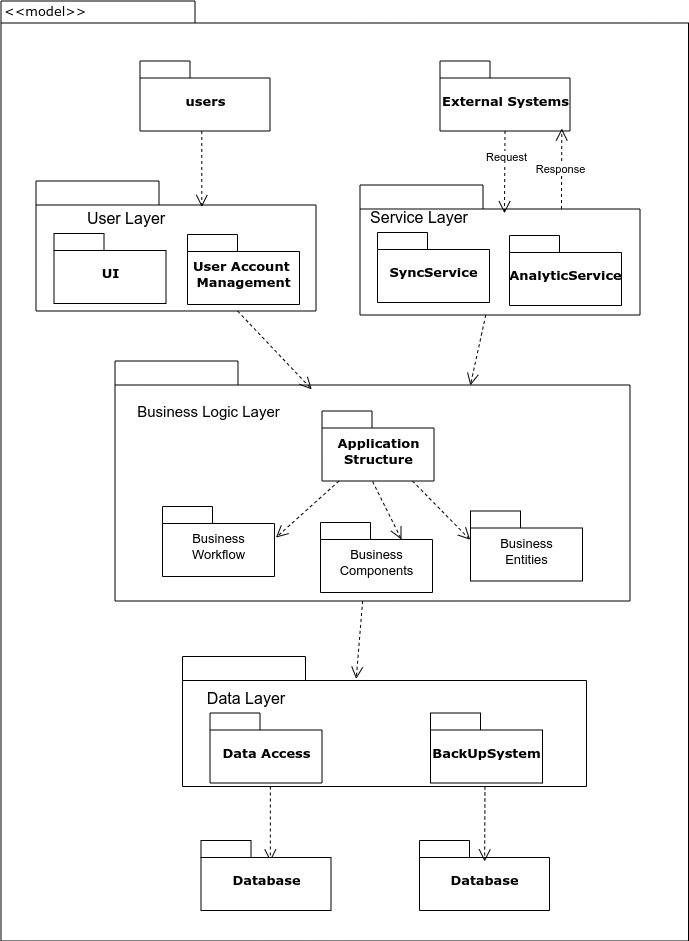


Рисунок 13 – диаграмма пакетов

**Вывод:**

В результате работы были разработаны UML-диаграммы классов, последовательности, взаимодействий и пакетов для информационной системы "Электронные визитки". Эти диаграммы помогают визуализировать структуру и процессы системы, обеспечивая наглядное представление бизнес-процесса.

# Практическая работа №6

**Цель работы:**

Рассмотреть факторы, которые могут повлиять на масштабируемость системы, такие как архитектура базы данных, алгоритмы и другие. Проанализировать возможности масштабирования и связанные с этим риски.

**Постановка задачи:**

По результатам анализа сделать мотивированный выбор архитектурной модели для проекта по выбранной тематике, описать возможности масштабирования проектируемой системы и связанные с этим риски.

**Ход работы:**

Для системы электронных визиток были проанализированы ключевые факторы, влияющие на масштабируемость:

**Факторы, влияющие на масштабируемость:**

1. **Архитектура**: В зависимости от выбранной архитектурной модели (например, клиент-сервер или микросервисы), система может масштабироваться по-разному. Клиент-серверная архитектура требует вертикального масштабирования серверов, тогда как микросервисы позволяют горизонтальное масштабирование отдельных компонентов системы.
2. **База данных**: Вертикальная масштабируемость может быть достигнута с помощью реляционных баз данных (PostgreSQL, MySQL), тогда как NoSQL базы данных (MongoDB, Cassandra) предлагают горизонтальное масштабирование. Важно предусмотреть шардирование и репликацию для обработки больших объёмов данных.
3. **Алгоритмы**: Для увеличения производительности необходимо оптимизировать алгоритмы поиска, фильтрации и сортировки данных. Алгоритмы, работающие с большими объемами данных, должны минимизировать использование ресурсов и поддерживать эффективное распределение нагрузки.
4. **Сетевое соединение**: Быстрое и стабильное сетевое соединение особенно важно при взаимодействии большого количества пользователей с системой. Необходимо предусмотреть кэширование данных и оптимизацию API-запросов для минимизации нагрузки на серверы.

**Изменения для обеспечения масштабирования**

Для обеспечения масштабируемости могут потребоваться следующие изменения:

1. **Облачные решения**: Использование облачной инфраструктуры позволит гибко распределять ресурсы в зависимости от текущей нагрузки. Облачные сервисы (например, AWS, GCP) позволяют автоматическое масштабирование серверов в пиковые моменты.
2. **Микросервисная архитектура**: Переход на микросервисы обеспечит независимость отдельных компонентов системы. Например, модуль управления визитками может масштабироваться независимо от модуля аутентификации пользователей.
3. **Балансировка нагрузки**: Балансировщики нагрузки позволят эффективно распределять запросы между различными серверами, предотвращая их перегрузку.
4. **Кэширование**: Использование технологий кэширования (Redis, Memcached) для часто запрашиваемых данных позволит снизить нагрузку на сервер и базу данных.

**Мотивированный выбор архитектурной модели**

Для системы электронных визиток наиболее подходящим вариантом является **микросервисная архитектура**, так как она позволяет:

* Разделить систему на независимые модули (например, модуль управления визитками, модуль поиска, модули работы с внешними сервисами).
* Масштабировать каждый модуль независимо, что особенно важно при увеличении числа пользователей и запросов.
* Использовать разные базы данных для разных модулей, что улучшает производительность и гибкость системы.

**Стратегии масштабирования**

1. **Контейнеризация**: Использование Docker для запуска микросервисов позволит быстро развертывать и управлять компонентами системы.
2. **Облачные ресурсы**: Переход на облачные платформы (AWS, Azure) позволит гибко настраивать ресурсы для обработки трафика и временно увеличивать мощность в периоды пиковых нагрузок (например, во время распродаж).
3. **Кэширование**: Кэширование товаров и пользовательских данных значительно сократит количество обращений к базе данных.
4. **Балансировка нагрузки**: Использование балансировщиков для распределения запросов между несколькими экземплярами серверов позволит избежать перегрузок.

**Оценка количества пользователей и запросов**

Допустим, система должна обслуживать 1000 активных пользователей одновременно, и каждый пользователь отправляет в среднем 3 запросов в минуту (просмотр визиток, добавление новых контактов и т.д.). Это даёт нам:

* 1000 пользователей × 3 запросов = 3000 запросов в минуту.
* Система должна быть рассчитана на возможность обработки 3000 запросов в минуту при стандартной нагрузке и больше при пиковых нагрузках.

**Риски**

* **Нагрузка на базу данных**: Высокое число пользователей может привести к перегрузке базы данных. Решение — использование шардирования и репликации данных, что позволит равномерно распределять нагрузку между серверами базы данных.
* **Сетевые задержки**: Высокая нагрузка на сеть может замедлить отклик системы, особенно в условиях пикового трафика. Использование CDN и оптимизация запросов поможет минимизировать сетевые задержки.
* **Управление микросервисами**: С увеличением количества микросервисов возрастает сложность их администрирования и обновлений. Использование CI/CD и мониторинга сервисов снизит эти риски.

**Вывод:**

Для системы электронных визиток наиболее оптимальным вариантом является микросервисная архитектура с использованием облачных решений и контейнеризации. Данная архитектура обеспечит гибкость, масштабируемость и устойчивость к нагрузкам, что позволит системе эффективно работать с увеличением числа пользователей и запросов.

# Практическая работа №7

**Цель работы:**

Выполнить анализ архитектуры клиент-серверной системы.

**Постановка задачи:**

Использовать выбранную ранее студентом тему практик и рассматривать в рамках своего проекта, с учетом первый практик, где был IDEF0 AS-IS и TO-BE.

**Ход работы:**

Для выполнения анализа архитектуры клиент-серверной системы на примере системы электронных визиток, давайте рассмотрим предложенные этапы последовательно:

**1. Монолитная архитектура**

На текущем этапе система электронных визиток может функционировать на основе монолитной архитектуры, где все основные функциональные элементы, такие как создание визиток, управление контактами, и обработка данных пользователей, объединены в едином приложении. Все модули системы тесно связаны и взаимодействуют напрямую друг с другом.

Ограничения и риски:

* Изменение одного компонента системы может потребовать изменений и в других компонентах, что замедляет разработку и усложняет тестирование.
* Ограниченная масштабируемость: вертикальное масштабирование возможно, но зависит от производительности одного сервера.
* Развертывание и обновление требует остановки всей системы, что может привести к временному прекращению её работы.

**2. Модульная архитектура**

При переходе к модульной архитектуре можно разделить систему на независимые компоненты, каждый из которых отвечает за конкретную функциональность. Например, модули управления пользователями, обработки данных визиток и визуализации контента могут существовать независимо друг от друга.

**Гибкость**:

* Модульная система позволяет вносить изменения в отдельные части системы, не затрагивая другие. Это упрощает тестирование и поддержку.
* Каждая часть системы может масштабироваться независимо, что улучшает производительность в условиях увеличения нагрузки.

**Преимущества**:

* Независимость компонентов друг от друга уменьшает взаимные зависимости и облегчает разработку.
* Возможность развертывания модулей на разных серверах или виртуальных машинах для повышения эффективности работы системы.

**3. Микросервисная архитектура**

Микросервисная архитектура разделяет систему на автономные сервисы, каждый из которых отвечает за конкретную бизнес-логику (например, управление пользователями или обработка данных визиток). Микросервисы взаимодействуют через API и могут развертываться независимо друг от друга.

**Готовность к масштабированию**:

* Микросервисная архитектура идеально подходит для масштабирования, так как каждый микросервис можно масштабировать отдельно, исходя из его нагрузки.
* Микросервисы можно разворачивать на различных платформах, используя контейнеры (например, Docker), что обеспечивает гибкость в управлении.

**Риски**:

* Более высокая сложность управления сервисами и их координацией.
* Необходимость внедрения механизмов оркестрации (Kubernetes) для контроля над жизненным циклом микросервисов.

**4. Изменения в архитектуре**

Монолитная архитектура:

Для обеспечения масштабируемости потребуется разделение системы на отдельные модули или микросервисы, чтобы снизить зависимости между компонентами.

Модульная архитектура:

Модульную архитектуру можно улучшить за счёт внедрения микросервисов для наиболее нагруженных модулей. Это повысит гибкость системы и упростит горизонтальное масштабирование.

Микросервисная архитектура:

Для успешного масштабирования микросервисов могут потребоваться решения для автоматического управления ресурсами и мониторинга (например, Kubernetes).

**5. Стратегии масштабирования**

Монолит:

* Вертикальное масштабирование: добавление ресурсов серверу.
* Использование кэшей для уменьшения нагрузки на базу данных.

Модульная архитектура:

* Разделение нагрузки между серверами с использованием балансировки.
* Горизонтальное масштабирование модулей.

Микросервисы:

* Автоматическое масштабирование через контейнеры и оркестрацию с использованием Kubernetes.
* Применение облачных технологий для динамического расширения ресурсов.

**6. Риски**

Монолит: Невозможность быстрого масштабирования, сложности с обслуживанием при высоких нагрузках.

Модульная архитектура: Возможные трудности с интеграцией между модулями при изменениях.

Микросервисы: Управление сложными зависимостями между микросервисами, сложности с безопасностью и версионностью API.

**7. Оценка нагрузки**

Монолит: Для поддержки 5000 пользователей система должна быть оптимизирована. Возможно использование мощных серверов для обработки запросов.

Модульная архитектура: Модульная система может поддерживать до 15 тысяч пользователей за счет независимого масштабирования модулей.

Микросервисы: Микросервисная архитектура позволит обслуживать более 50 тысяч пользователей за счет независимого масштабирования каждого микросервиса.

**8. Выбор архитектурной модели**

Для информационной системы электронных визиток на первом этапе целесообразно использовать модульную архитектуру. Она обеспечивает достаточную гибкость и масштабируемость, что позволяет вносить изменения и добавлять новые функции без сложностей, связанных с монолитом. В дальнейшем, по мере роста нагрузки и расширения функционала, возможно переход на микросервисную архитектуру для улучшения масштабируемости и управляемости системы.

**Вывод:**

Модульная архитектура является оптимальным решением на начальном этапе, так как она обеспечивает достаточную гибкость и масштабируемость для модификации системы. В будущем можно рассмотреть переход на микросервисы для более детализированного управления нагрузками и улучшения производительности.

# Практическая работа №8

**Цель работы:**

Провести анализ и изучение применения шаблонов и паттернов проектирования в контексте клиент-серверных систем.

**Постановка задачи:**

1. Рассмотрите следующие шаблоны и паттерны:

- Шаблоны клиент-серверных систем (например, шаблоны для взаимодействия клиента и сервера).

- Структурные паттерны (например, "Адаптер" (Adapter) для интеграции разных интерфейсов, "Мост" (Bridge) для разделения абстракции и реализации).

- Паттерны поведения (например, "Состояние" (State) для управления состоянием системы, "Стратегия" (Strategy) для замены алгоритмов).

- Порождающие паттерны (например, "Фабричный метод" (Factory Method) для создания объектов, "Одиночка" (Singleton) для гарантированной единственной инстанциации).

2. Выбор конкретных шаблонов и паттернов: Выберите несколько конкретных шаблонов и паттернов, которые, наиболее подходят для применения в вашей клиент-серверной системе. Технически грамотно обоснуйте выбор этих паттернов.

3. Анализ применения: рассмотрите, как выбранные вами шаблоны и паттерны могут быть использованы для улучшения архитектуры и функциональности вашей системы. Опишите, как эти шаблоны и паттерны могут решать конкретные проблемы или улучшать производительность проектируемой клиент-серверной системы.

4. Проектирование и реализация: спроектируйте часть системы, используя выбранные шаблоны и паттерны. Результаты проектирования представьте в виде диаграмм и текстового описания, если потребуется добавьте глоссарий.

5. Отчет: подготовьте отчет, включающий ваш анализ применения шаблонов и паттернов, описания применения в выбранной системе и, результаты проектирования.

**Ход работы:**

**1. Рассмотрение шаблонов и паттернов**

Для анализа в клиент-серверной архитектуре магазина компьютеров выберем несколько ключевых шаблонов и паттернов, которые могут существенно улучшить взаимодействие между клиентом и сервером, а также структуру приложения.

**Шаблоны клиент-серверных систем:**

**Запрос-ответ** (Request-Response): Это базовый шаблон для клиент-серверного взаимодействия, где клиент отправляет запрос серверу, а сервер отвечает с результатом. В контексте интернет-магазина, такой шаблон может использоваться для обработки запросов на поиск товаров, оформление заказов и получение информации о наличии товаров.

**Push Model** (модель с обратной связью): Сервер активно уведомляет клиента об изменениях, например, при обновлении информации о наличии товаров или при изменении статусов заказа.

**Структурные паттерны:**

**Адаптер** (Adapter): Этот паттерн может быть применён для интеграции с внешними системами, такими как платёжные шлюзы или системы логистики, предоставляющие разные интерфейсы.

**Мост** (Bridge): Паттерн "Мост" полезен для разделения бизнес-логики (например, обработка товаров) и представления (например, мобильное приложение или веб-интерфейс). Это позволит легко адаптировать приложение к разным платформам.

**Паттерны поведения:**

**Стратегия** (Strategy): Для обработки разных стратегий расчетов стоимости доставки в зависимости от региона или метода доставки может быть использован паттерн "Стратегия". Это позволяет гибко изменять алгоритмы без изменения основного кода.

**Состояние** (State): Для управления состояниями заказа (создан, оплачен, отправлен, доставлен) можно использовать паттерн "Состояние". Это позволяет легко обрабатывать переходы между состояниями и управлять изменениями статусов.

**Порождающие паттерны:**

**Фабричный метод** (Factory Method): Паттерн может применяться для создания различных объектов заказа, которые могут отличаться в зависимости от типа доставки, оплаты или скидочных предложений.

**Одиночка** (Singleton): Паттерн "Одиночка" может использоваться для создания единственного экземпляра подключения к базе данных или конфигурационного менеджера, что предотвращает создание множества экземпляров и оптимизирует использование ресурсов.

**2. Выбор конкретных шаблонов и паттернов**

Для информационной системы «Электронные визитки» наилучшим образом подходят следующие шаблоны и паттерны:

* Запрос-ответ (Request-Response): Основной шаблон для взаимодействия между клиентом и сервером, который будет использоваться для операций по запросам к базе данных визиток.
* Адаптер (Adapter): Этот паттерн будет использован для интеграции с внешними системами (например, обмен визитками через другие приложения).
* Стратегия (Strategy): Для обработки алгоритмов фильтрации и сортировки визиток по различным параметрам (например, дате создания, имени владельца).
* Состояние (State): Для управления состояниями визиток: активная, архивная и т.д.
* Одиночка (Singleton): Для оптимизации подключения к базе данных.

Эти паттерны обеспечат гибкость, масштабируемость и улучшенную производительность системы.

**3. Анализ применения**

1. **Запрос-ответ (Request-Response):** Основной механизм для выполнения запросов на добавление, изменение и удаление визиток в системе.
2. **Адаптер (Adapter):** Позволяет легко интегрировать систему с API других сервисов или приложений для обмена визитками.
3. **Состояние (State):** Удобен для управления состояниями визиток, позволяя легко изменять их статус в процессе работы.
4. **Одиночка (Singleton):** Обеспечивает единое подключение к базе данных, минимизируя количество обращений и увеличивая производительность.

Представление паттернов на диаграмме пакетов (рисунок 13).

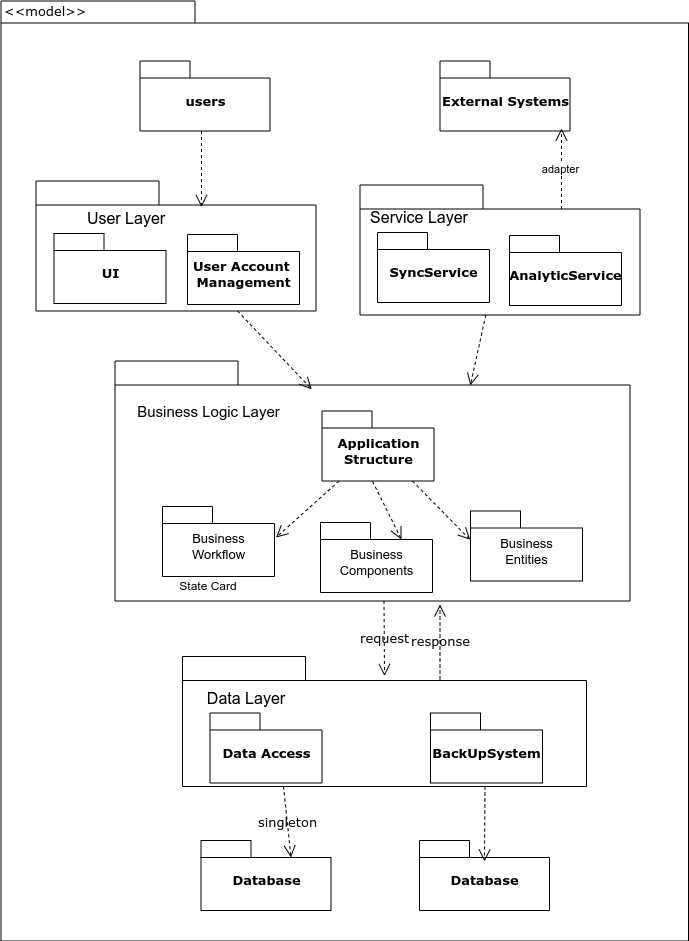


Рисунок 13 – Диаграмма пакетов

**Вывод:**

В ходе анализа клиент-серверной архитектуры для системы электронной визитки были выбраны и применены несколько ключевых паттернов: "Запрос-ответ" для обработки запросов клиентов, "Адаптер" для интеграции с внешними системами, "Мост" для разделения интерфейсов и бизнес-логики, "Стратегия" для гибкости в расчетах доставки, "Состояние" для управления статусами заказов, и "Одиночка" для оптимизации подключения к базе данных. Эти паттерны обеспечивают гибкость, масштабируемость и упрощают дальнейшую модификацию системы.диаграмму последовательности

# Практическая работа №9

## Цель

Создать диаграммы состояний, компонентов, развертывания, пакетов с использованием PluntUML.

## Ход работы

На рисунке 14 представлен исходный код диаграммы состояний.

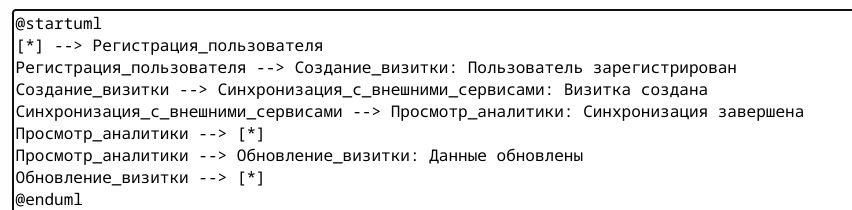


Рисунок 14 – Код диаграммы состояний

Диаграмма состояний показывает возможные состояния объекта визитка. На рисунке 15 представлена диаграмма состояний.

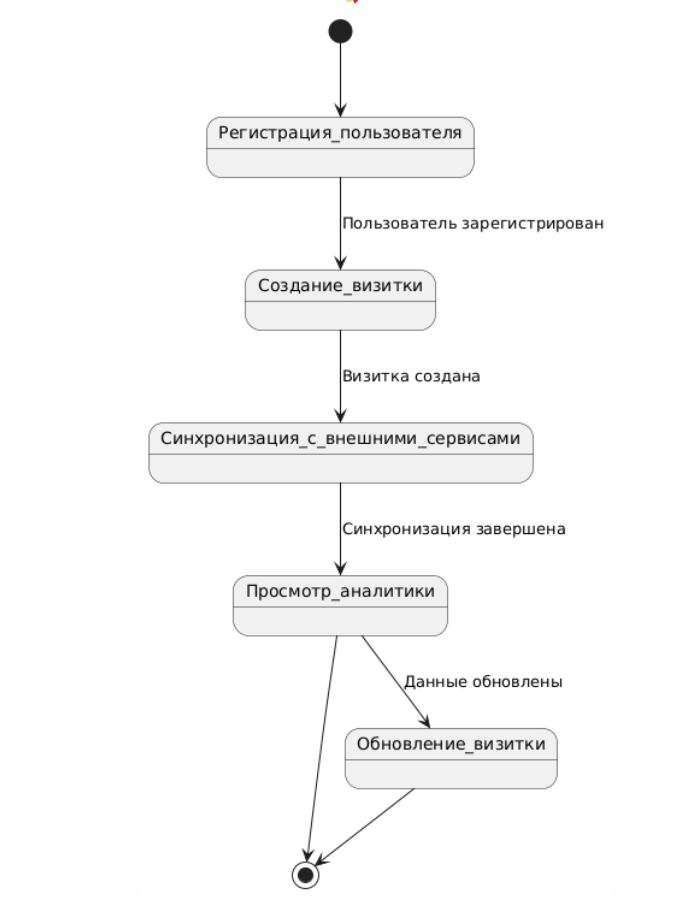


Рисунок 15 – Диаграмма состояний

На рисунке 16 представлен исходный код диаграммы компонентов.

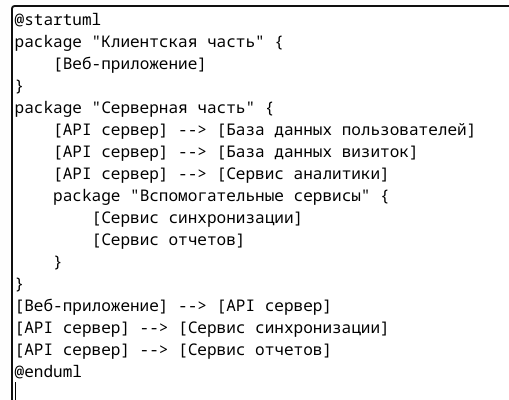


Рисунок 16 – Код диаграммы компонентов

Диаграмма представлена на рисунке 17.

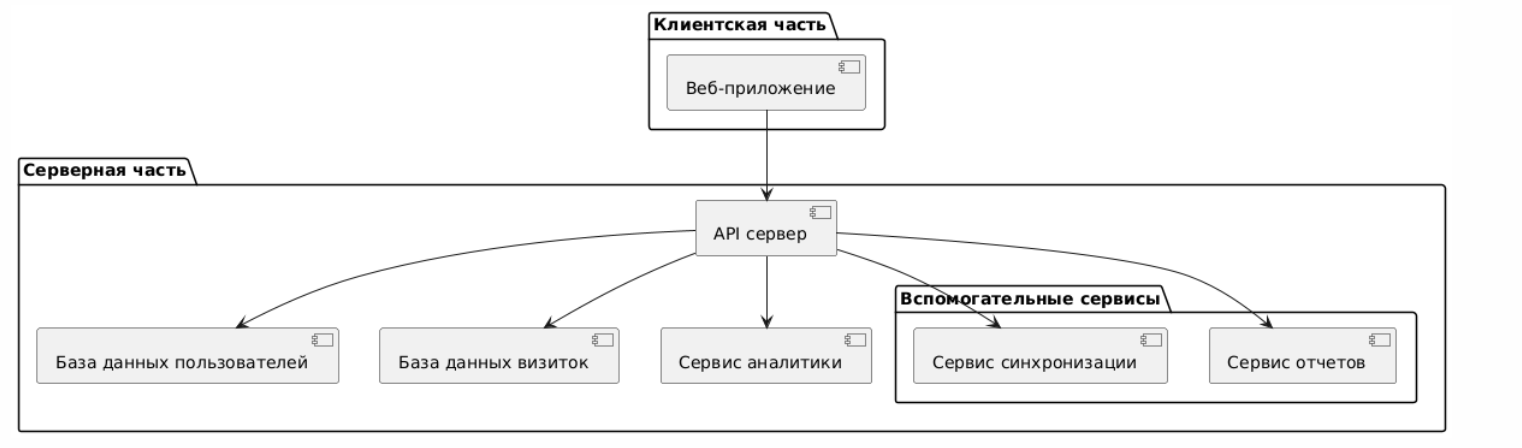


Рисунок 17 – Диаграмма компонентов

Далее рассмотрим диаграмму развёртывания. На рисунке 18 представлен исходный код.

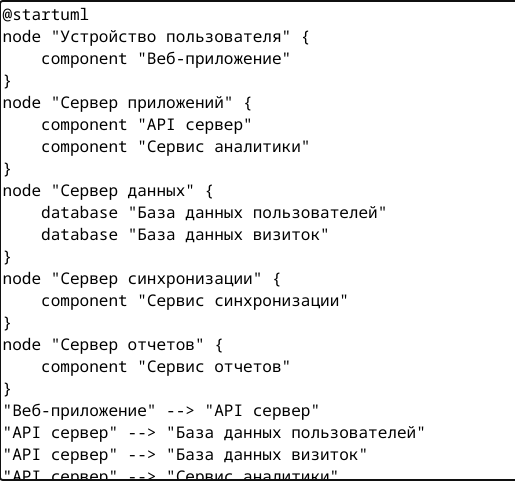


Рисунок 18 – Код диаграммы развёртывания

На рисунке 19 представлена диаграмма компонентов.

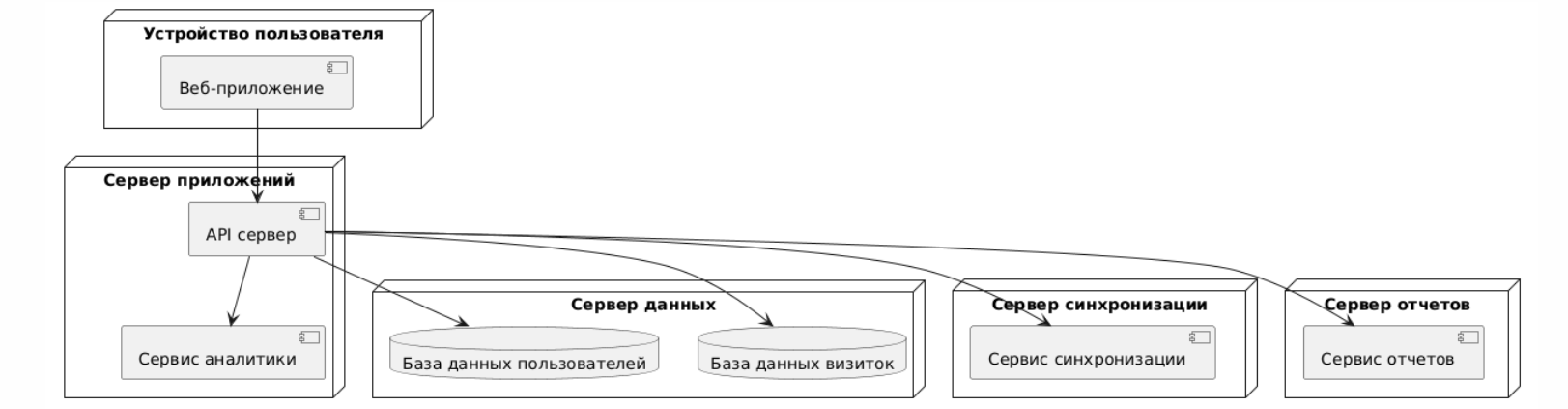


Рисунок 19 – Диаграмма развёртывания

На рисунке 20 представлен исходный код диаграммы пакетов.

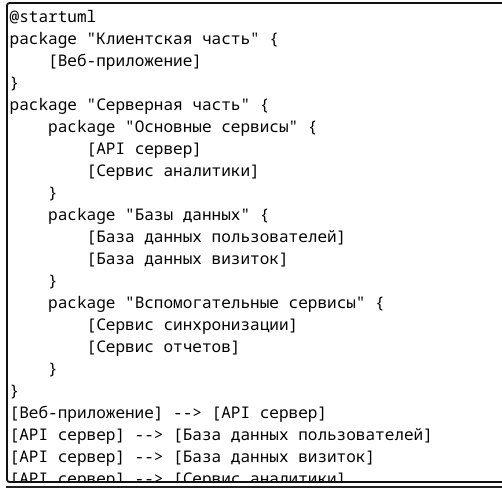


Рисунок 20 – Исходный код диаграммы пакетов

На рисунке 21 представлена диаграмма пакетов.

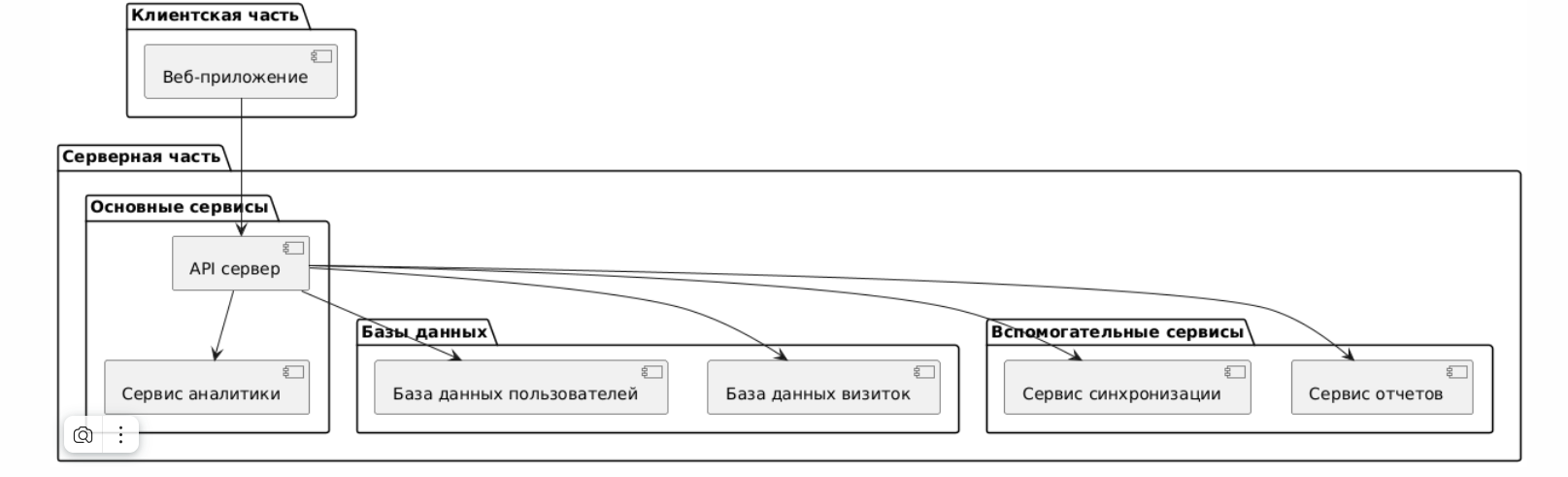


Рисунок 21 – Диаграмма пакетов

# Вывод

В результате выполнения практической работы были диаграммы состояний, компонентов, развертывания, пакетов с помощью PluntUML. Также были получены навыки работы с PluntUML и настройки стилей.

# Практическая работа №10

## Цель

Освоить использование различных инструментов и нотаций для моделирования информационных систем, научиться создавать сетевые диаграммы с помощью nwdiag, познакомиться с архитектурным моделированием с использованием языка ArchiMate, составить структуру разбиения работ (WBS) для планирования проекта, применить эти инструменты при выполнении практических заданий и курсовой работы.

## Ход работы

На рисунке 23 представлен исходный код диаграммы nwdiag.

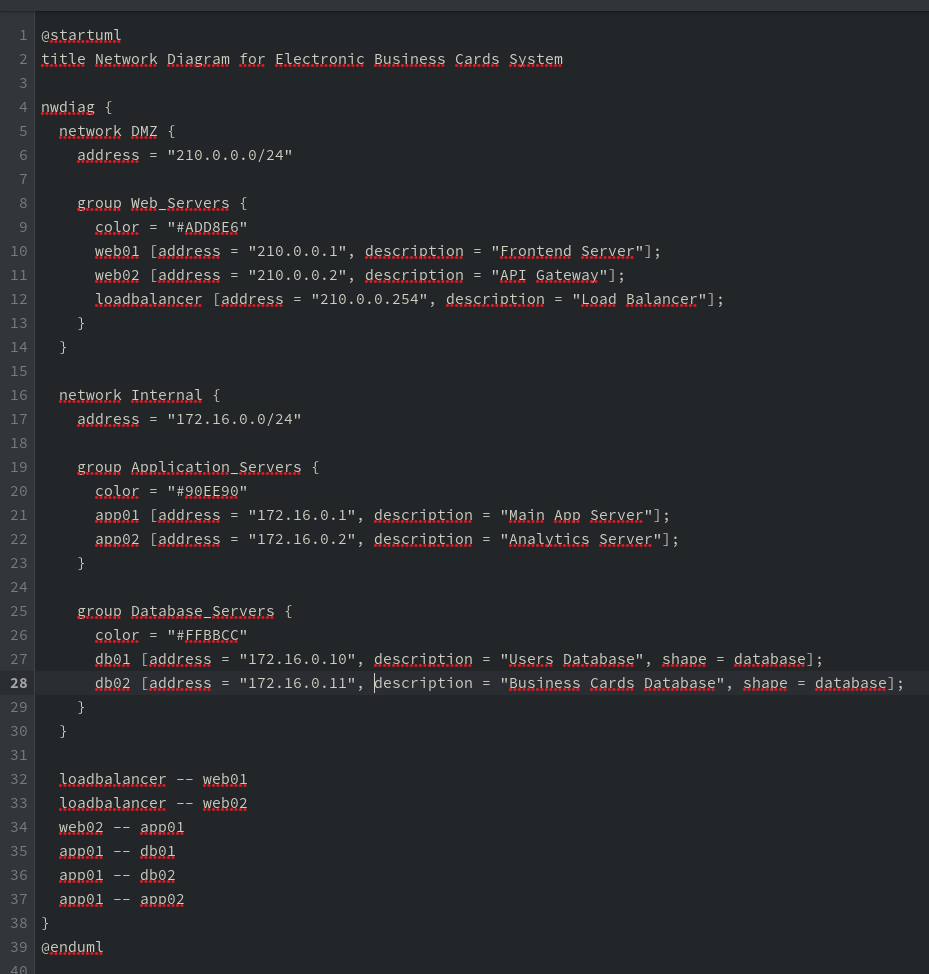


Рисунок 33 – Код диаграммы nwdiag

На рисунке 24 представлена диаграмма nwdiag.

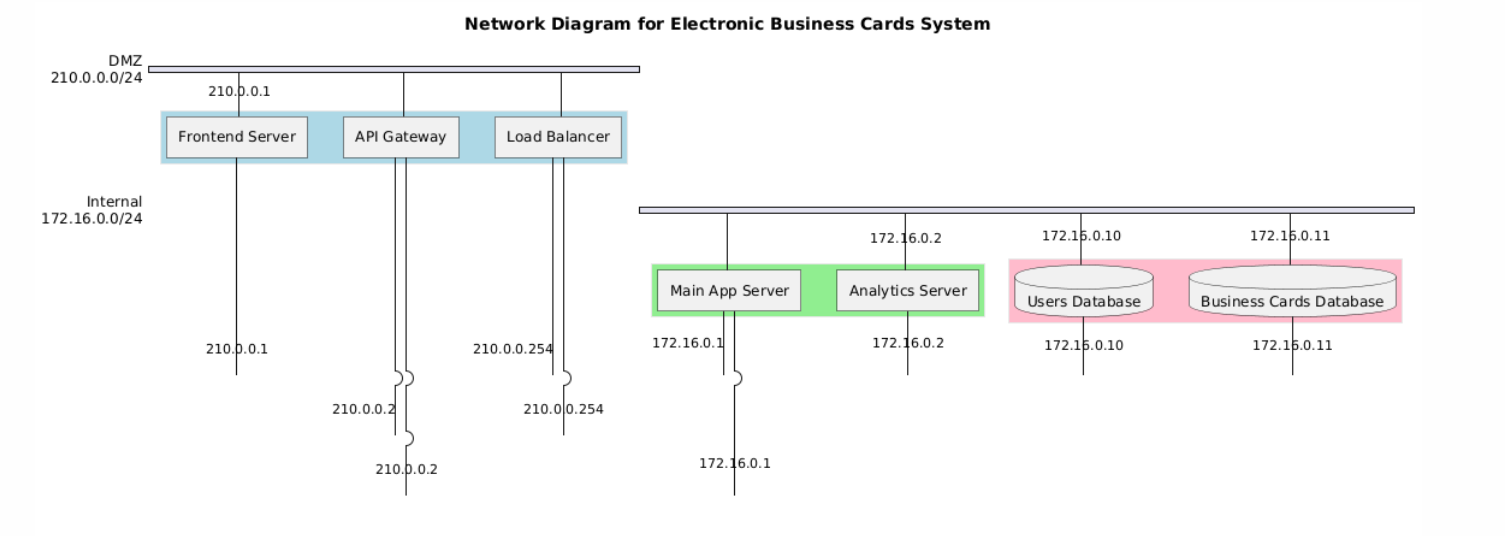


Рисунок 24 – Диаграмма nwdiag

На рисунке 25 представлен исходный код диаграммы ArchiMate.

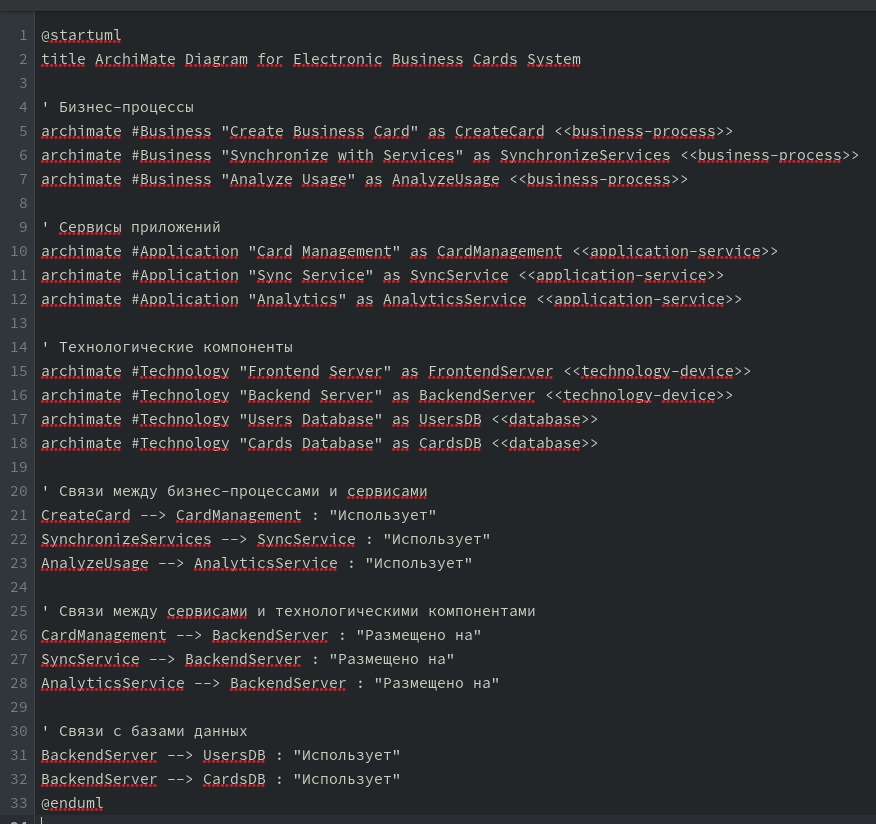


Рисунок 25 – Код диаграммы ArchiMate

Диаграмма ArchiMate представлена на рисунке 26.

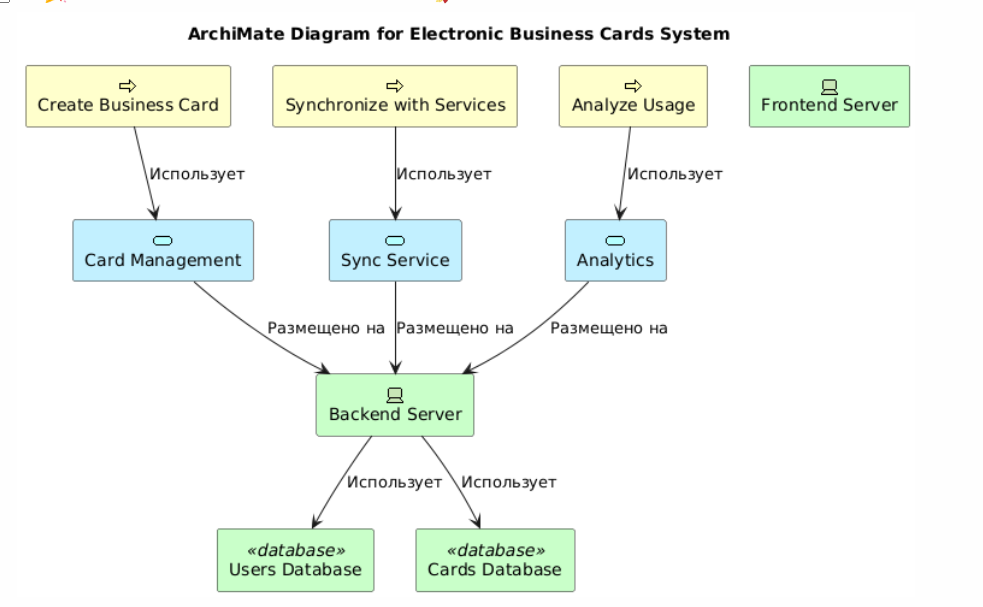


Рисунок 26 – Диаграмма ArchiMate

На рисунке 27 представлен исходный код диаграммы структура разбиения работ.

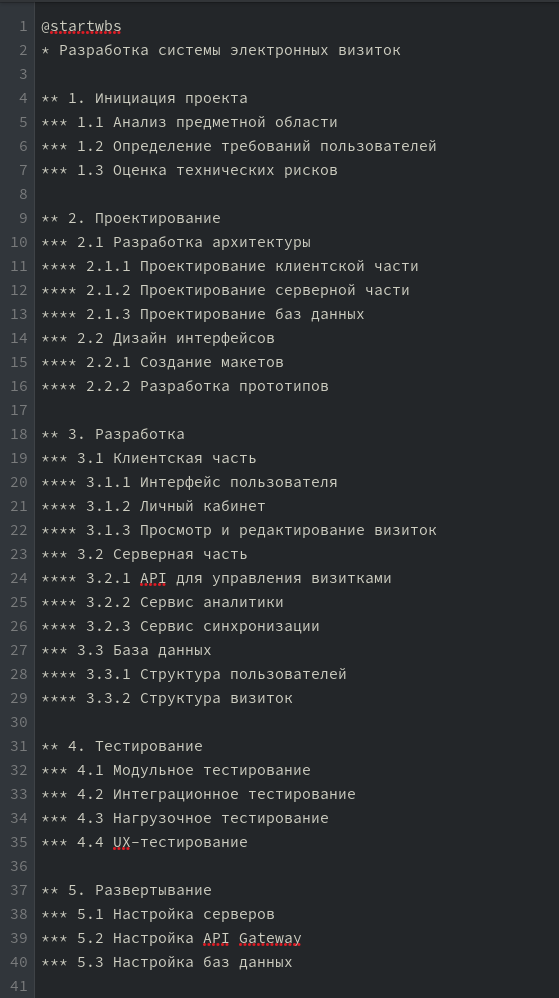


Рисунок 27 – Код WBS диаграммы

На рисунке 28 – Представлена WBS диаграмма.

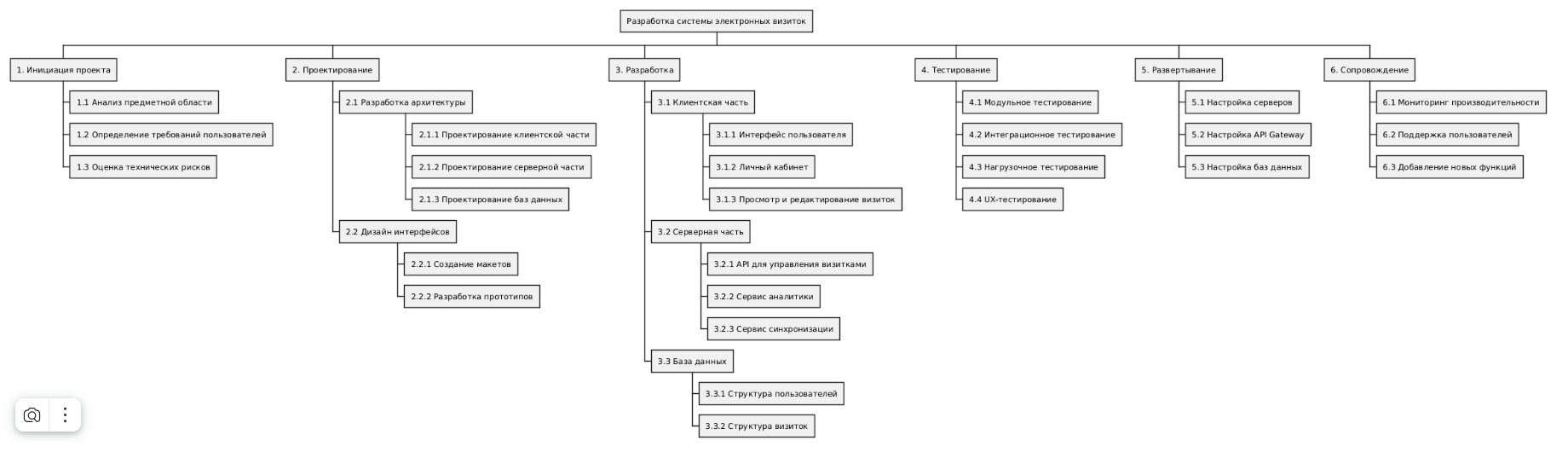


Рисунок 28 – WBS диаграмма

## Вывод

В результате выполнения практической работы были освоены использования различных инструментов и нотаций для моделирования информационных систем, научились создавать сетевые диаграммы с помощью nwdiag, познакомились с архитектурным моделированием с использованием языка ArchiMate, составили структуру разбиения работ (WBS) для планирования проекта.

# Практическая работа №11

## Цель

Освоить использование C4-модели для описания архитектуры программных систем.

Научиться создавать диаграммы различных уровней C4-модели.

Применить полученные знания к своей индивидуальной теме проекта.

Углубить понимание архитектурных решений и их документирование.

## Ход работы

На рисунке 29 представлен код модели C4 уровня Context.

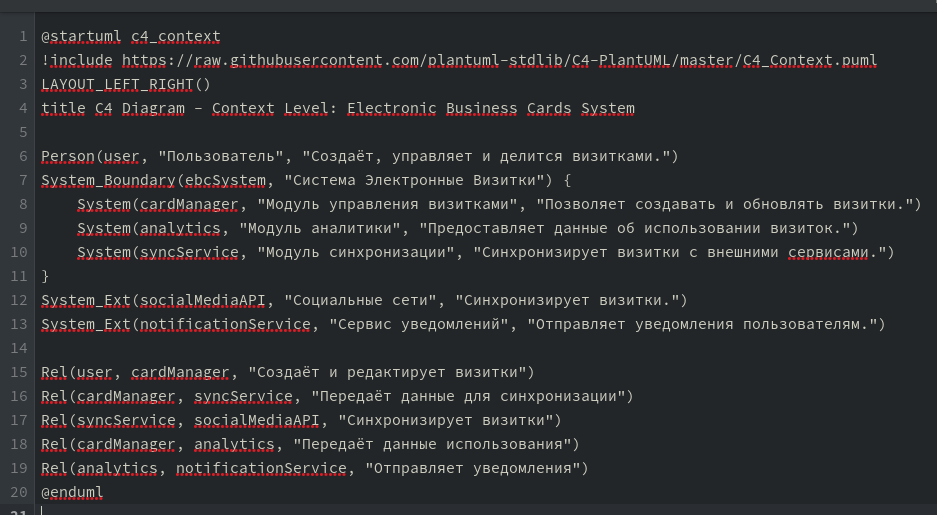


Рисунок 29 – Код уровня Context

На рисунке 30 представлена диаграмма Context.

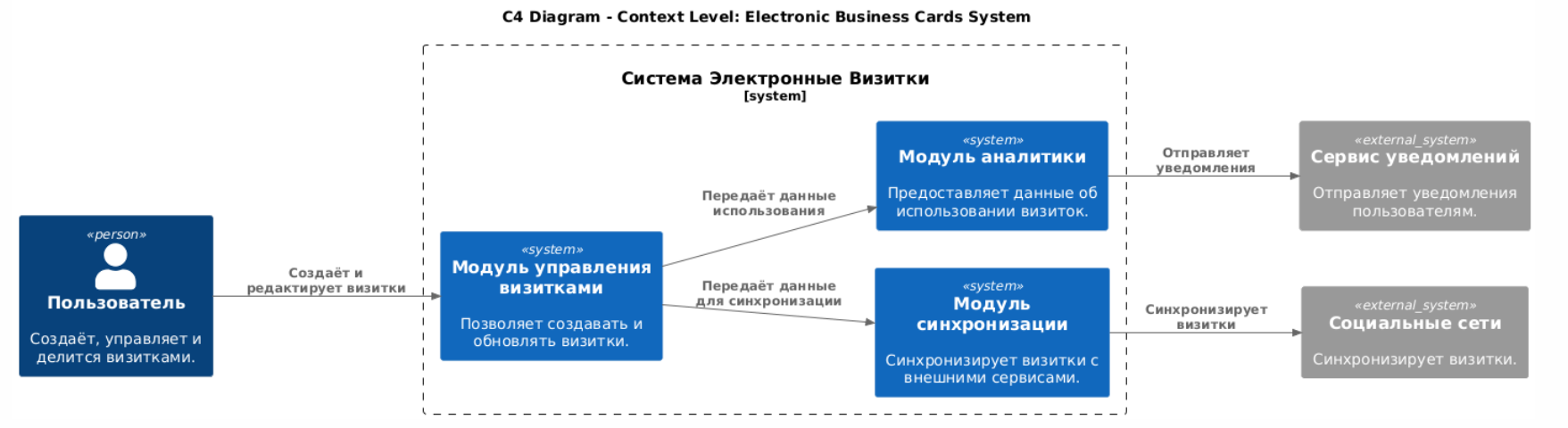


Рисунок 30 – Диаграмма уровня Context

На рисунке 31 представлен исходный код уровня Container.

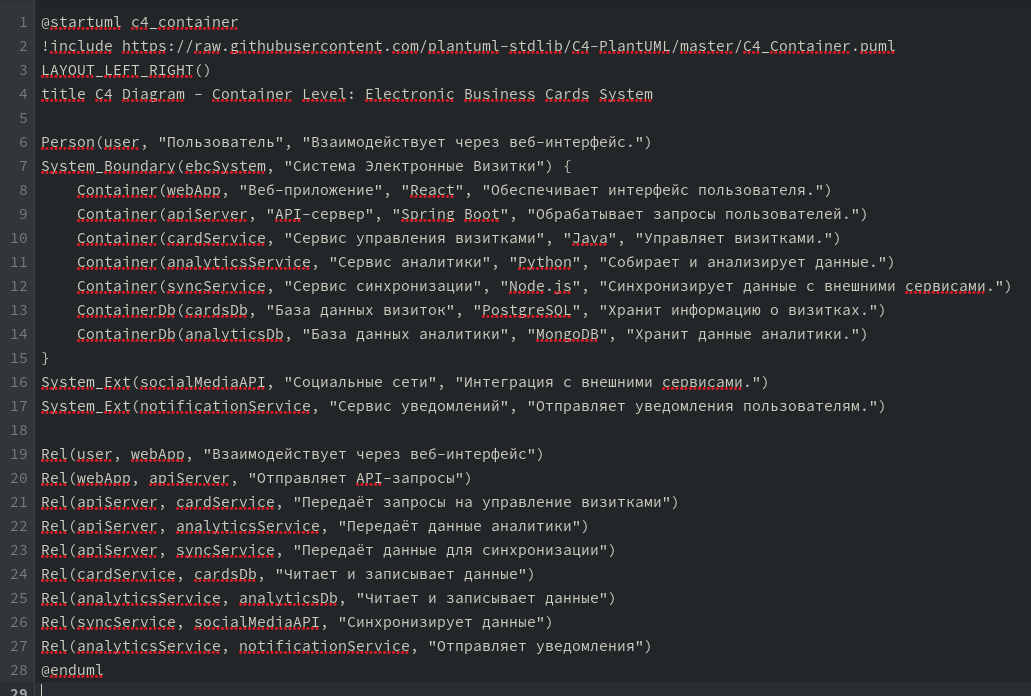


Рисунок 31 – Код уровня Container.

На рисунке 32 представлен код модели C4 уровня Container.

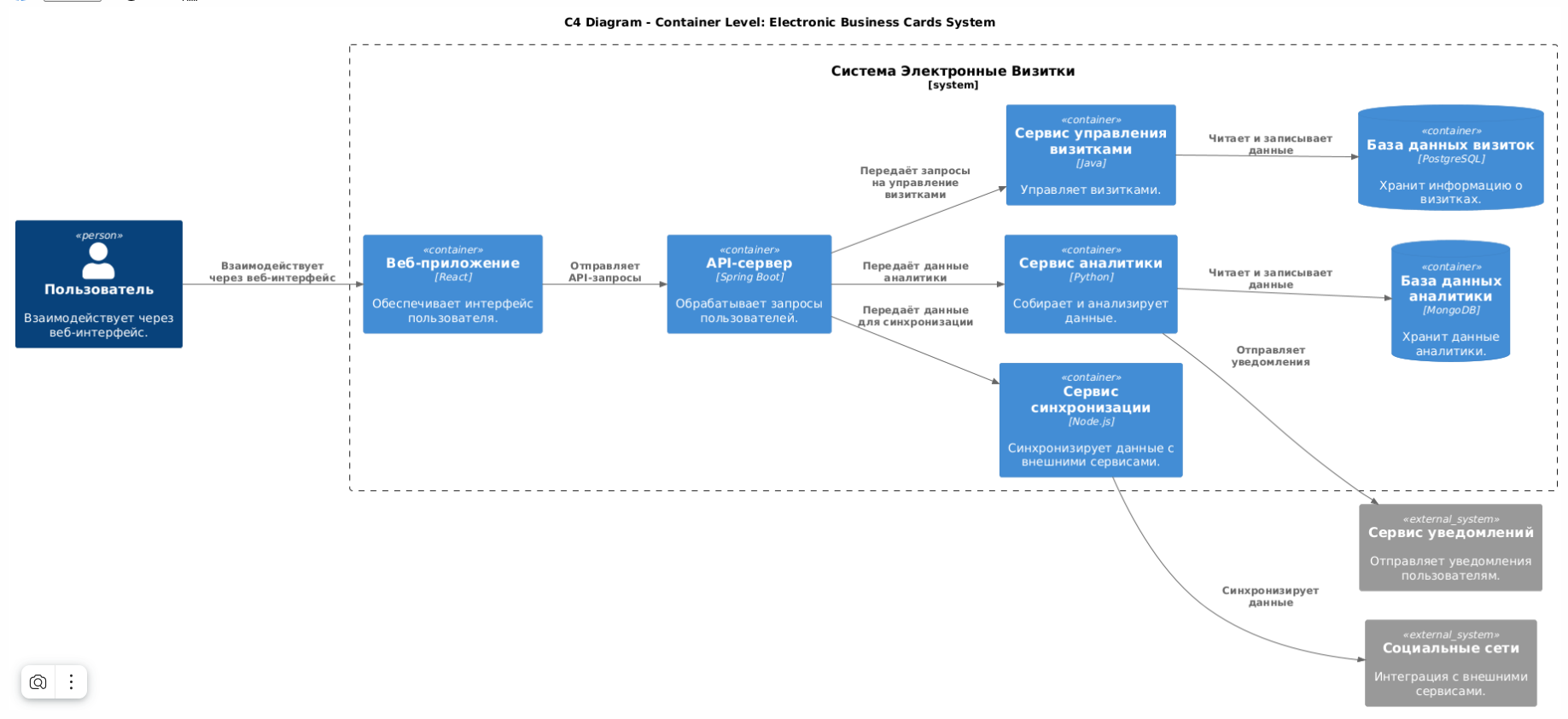


Рисунок 32 – Модель уровня Container

На рисунке 33 представлен исходный код уровня Code.

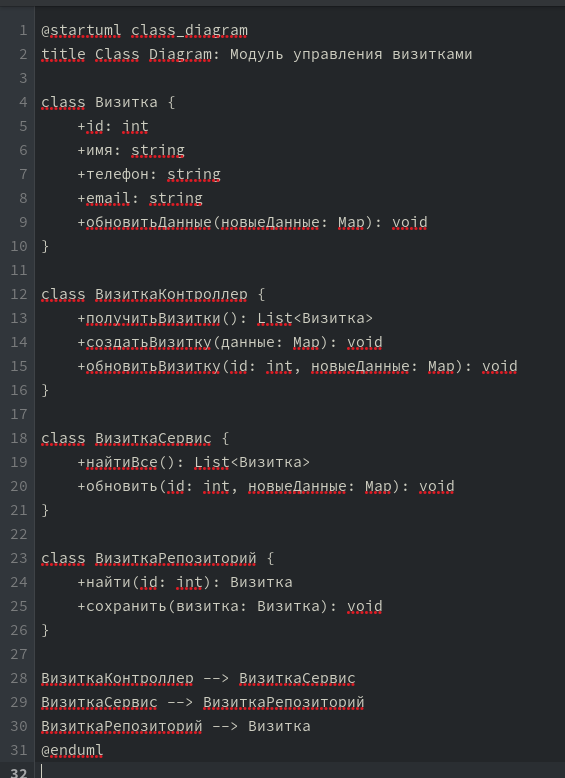


Рисунок 33 – Код уровня Code

На рисунке 34 представлен код модели C4 уровня Code.

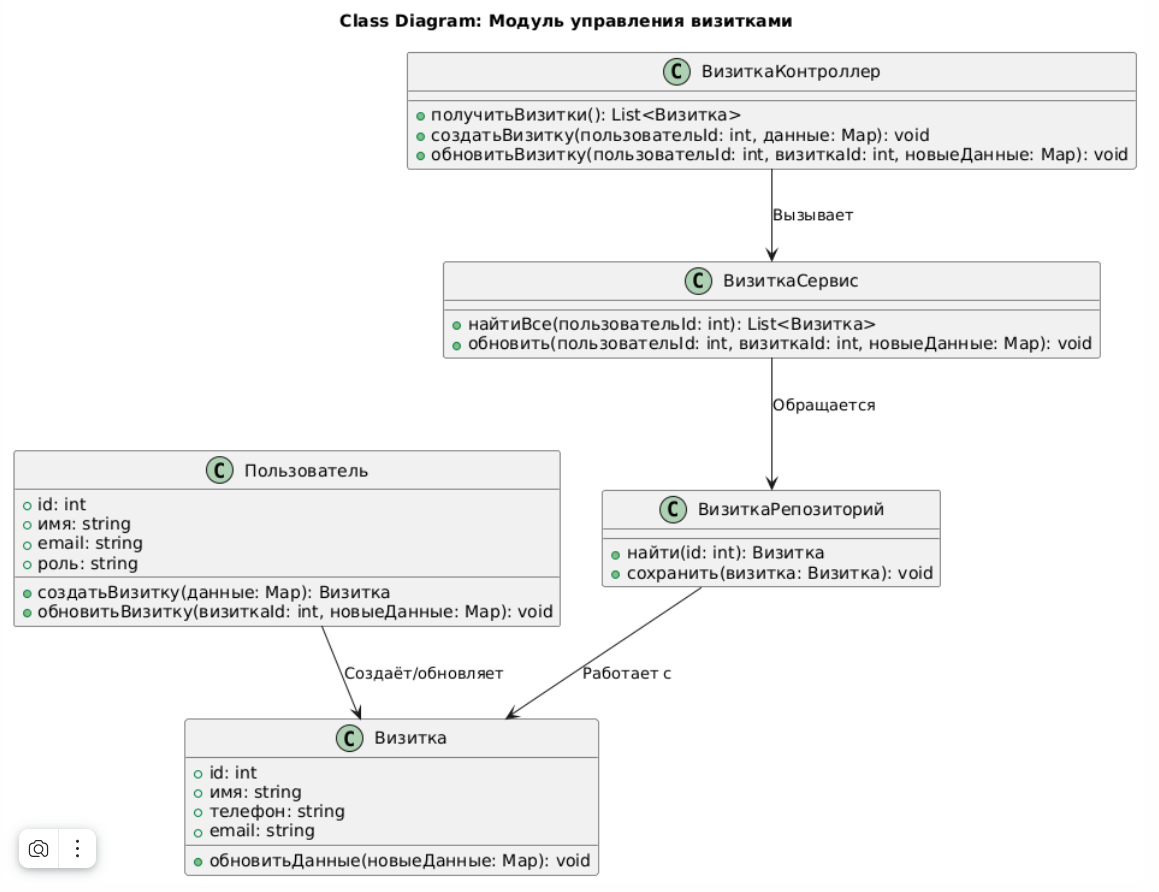


Рисунок 34 – Модель уровня Code

## Вывод

В результате выполнения практической работы были получены навыки работы с моделью C4. Построены диаграммы на уровнях Context, Container, Code модели C4.

# Практическая работа №12

## Цель работы

Освоить синтаксис и возможности Mermaid для создания различных типов диаграмм в документах Markdown.

Научиться интегрировать диаграммы Mermaid в Markdown-документы.

Применить полученные знания для визуализации структур и процессов в рамках своей индивидуальной темы проекта, используя новые типы диаграмм.

**Ход работы**

Рисунок 35 – Диаграмма путешествия пользователя

Рисунок 36 – Ментальная карта функциональных возможностей

Рисунок 37 – Квадрант-граф

Рисунок 38 – Гит граф

Документацию можно найти в репозитории - <https://github.com/SmknSe/pkss12/blob/main/README.md>.

## Выводы

В ходе данной работы освоены синтаксис и возможности Mermaid для создания различных типов диаграмм в документах Markdown.

# Список использованных источников

1. Гвоздева Т. В., Баллод Б. А.. Проектирование информационных систем. Стандартизация [Электронный ресурс]:учебное пособие. - Санкт-Петербург: Лань, 2019. - 252 с. – Режим доступа: https://e.lanbook.com/book/115515
2. Рочев К. В.. Информационные технологии. Анализ и проектирование информационных систем [Электронный ресурс]:учебное пособие. - Санкт-Петербург: Лань, 2019. - 128 с. – Режим доступа: https://e.lanbook.com/book/122181
3. Вейцман В. М.. Проектирование информационных систем [Электронный ресурс]:учебное пособие. - Санкт- Петербург: Лань, 2019. - 316 с. – Режим доступа: https://e.lanbook.com/book/122172
4. Остроух А. В., Суркова Н. Е.. Проектирование информационных систем [Электронный ресурс]:монография. - Санкт-Петербург: Лань, 2019. - 164 с. – Режим доступа: https://e.lanbook.com/book/118650
5. Информационно-правовой портал ГАРАНТ http:// www.garant.ru
6. Консультант Плюс http:// www.consultant.ru
7. Научная электронная библиотека eLIBRARY.ru
8. Российский технологический журнал
9. Тестирование и контроль программных систем: сайт. – URL: https://xreferat.com/33/2759-1-sushnost-i-osobennosti-ispol-zovaniya-instrumental-nogo-programmnogo-obespecheniya.html (дата обращения 15.03.2020). – Текст: электронный.
10. Контроль и корректировка кода: сайт. – URL: https://studfile.net/preview/2790134/page:3/ (дата обращения 15.03.2020). – Текст: электронный.
11. Object Constraints Language: сайт. – URL: https://ami.nstu.ru/~vms/lecture/lecture12/lecture12.htm (дата обращения 15.03.2020). – Текст: электронный.
12. Эксплуатация и сопровождение проекта: сайт. – URL: http://kgau.ru/istiki/umk/pis/l7.htm (дата обращения 15.03.2020). – Текст: электронный.
13. Типовые модели систем: сайт. – URL: http://kgau.ru/istiki/umk/pis/l37.htm (дата обращения 15.03.2020). – Текст: электронный.
14. Проектирование и разработка Java приложений и систем: сайт. – URL: https://dic.academic.ru/dic.nsf/ruwiki/608820 (дата обращения 15.03.2020). – Текст: электронный.
15. Современные CASE средства проектирования систем: сайт. – URL: http://window.edu.ru/resource/616/73616/files/kulyabov-korolkova\_formal-methods.pdf (дата обращения 15.03.2020). – Текст: электронный.
16. Реляционные СУБД: сайт. – URL: https://compress.ru/article.aspx?id=10082 (дата обращения 15.03.2020). – Текст: электронный.
17. Проектирование ER-диаграмм: сайт. – URL: http://citforum.ru/cfin/prcorpsys/infsistpr\_09.shtml (дата обращения 15.03.2020). – Текст: электронный.
18. Стандарты разработки: сайт. – URL: http://venec.ulstu.ru/lib/disk/2017/460.pdf (дата обращения 15.03.2020). – Текст: электронный.
19. Архитектура ЭВМ и систем: сайт. – URL: https://arxiv.org/ftp/arxiv/papers/1802/1802.06769.pdf (дата обращения 15.03.2020). – Текст: электронный.