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 О ПРАКТИЧЕСКИМ РАБОТАМ

по дисциплине

«Проектирование клиент-серверных систем»

на тему

«Информационная система Электронный журнал школьника»

Выполнил студент группы ИКБО-20-21 Фомичев Р.А.

Принял Мельников Д.А.

Москва

Оглавление

1 Практическая работа №1

Цель работы:

1 Знакомство с графической нотацией формализации и описания бизнес-процессов IDEF0. Знакомство с 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 A0 в нотации IDEF0

В качестве входа по управлению были выбраны:

- Внутренние регламенты
- Законодательство

В качестве входящих потоков были выбраны:

- Заявление о приеме на работу
- Документы о ремонтных работах

В качестве механизмов используются:

- Водители
- Кондукторы
- Трамваи
- Диспетчеры
- Главный инженер
- Экономист
- Директор

В качестве выходов после выполнения ИС получены:

- Отчеты о поломках

- Отчеты о выручке
- Финансовые отчеты

Сама контекстная диаграмма процесса ИС Трамвайно депо школьника представлена на рисунке 1.

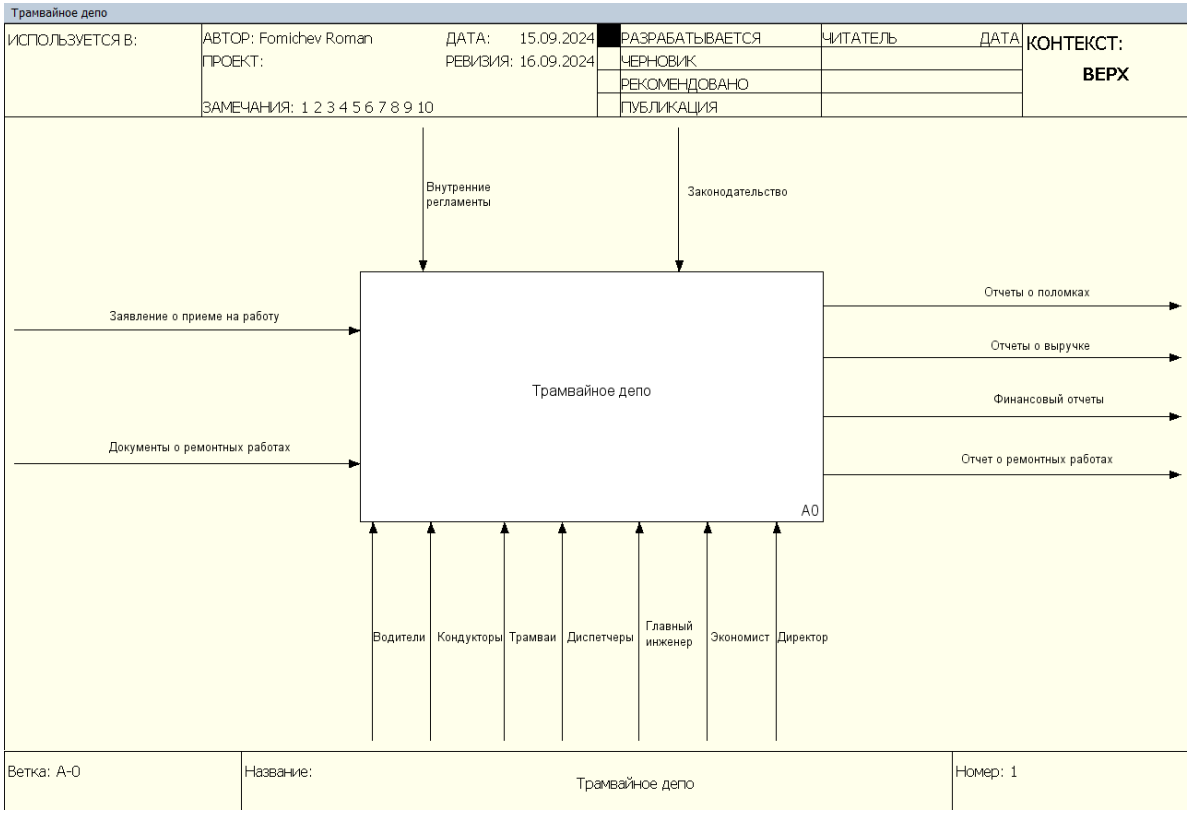

Рисунок 1 – Контекстная диаграмма A0

Далее была произведена декомпозиция основного функционального блока A0 (рисунок 2). Были получены следующие функциональные блоки:

- Управление персоналом – A1
- Планирование маршрутов – A2
- Ремонт и обслуживание трамваев – A3
- Управление финансами – A4

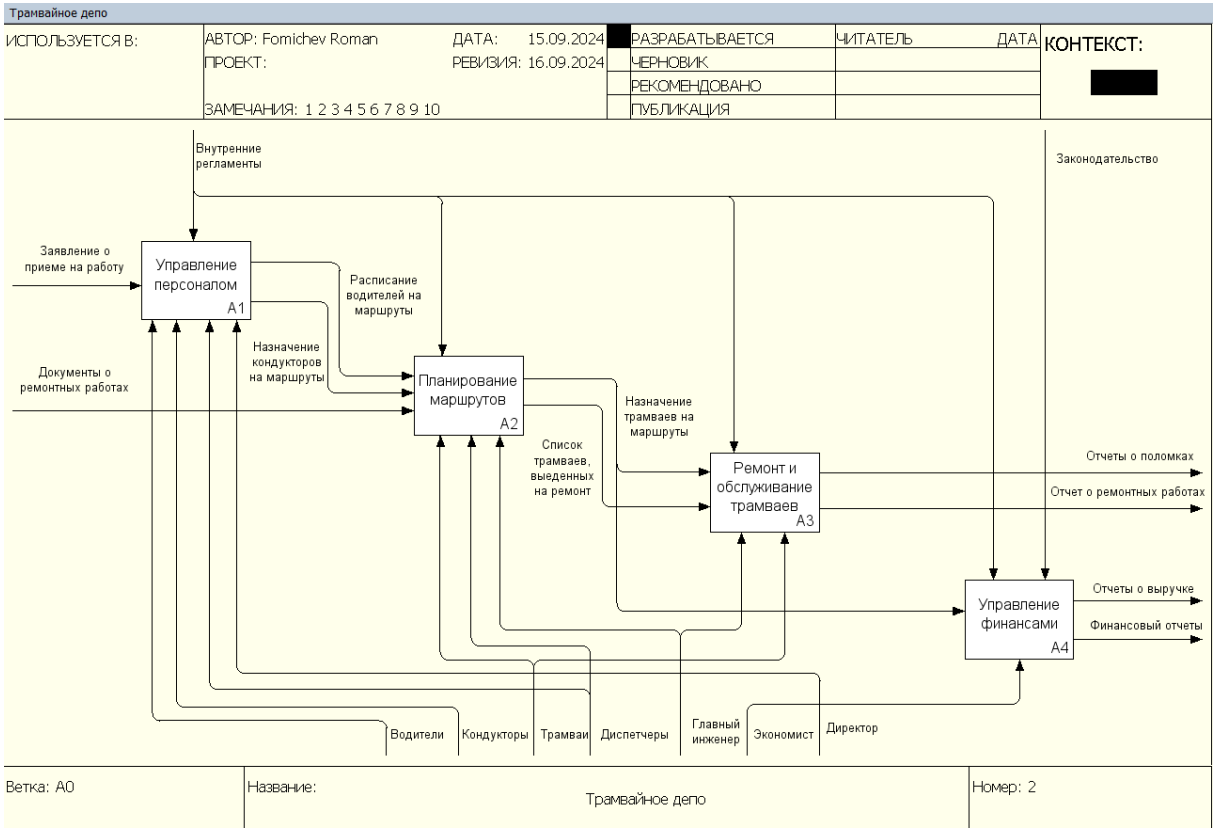

Рисунок 2 – Декомпозиция функционального блока

Декомпозиция блока A1:

- Назначение водителей – A11
- Перенаправление кондукторов – A12

Диаграмма декомпозированного блока представлена на рисунке 3.

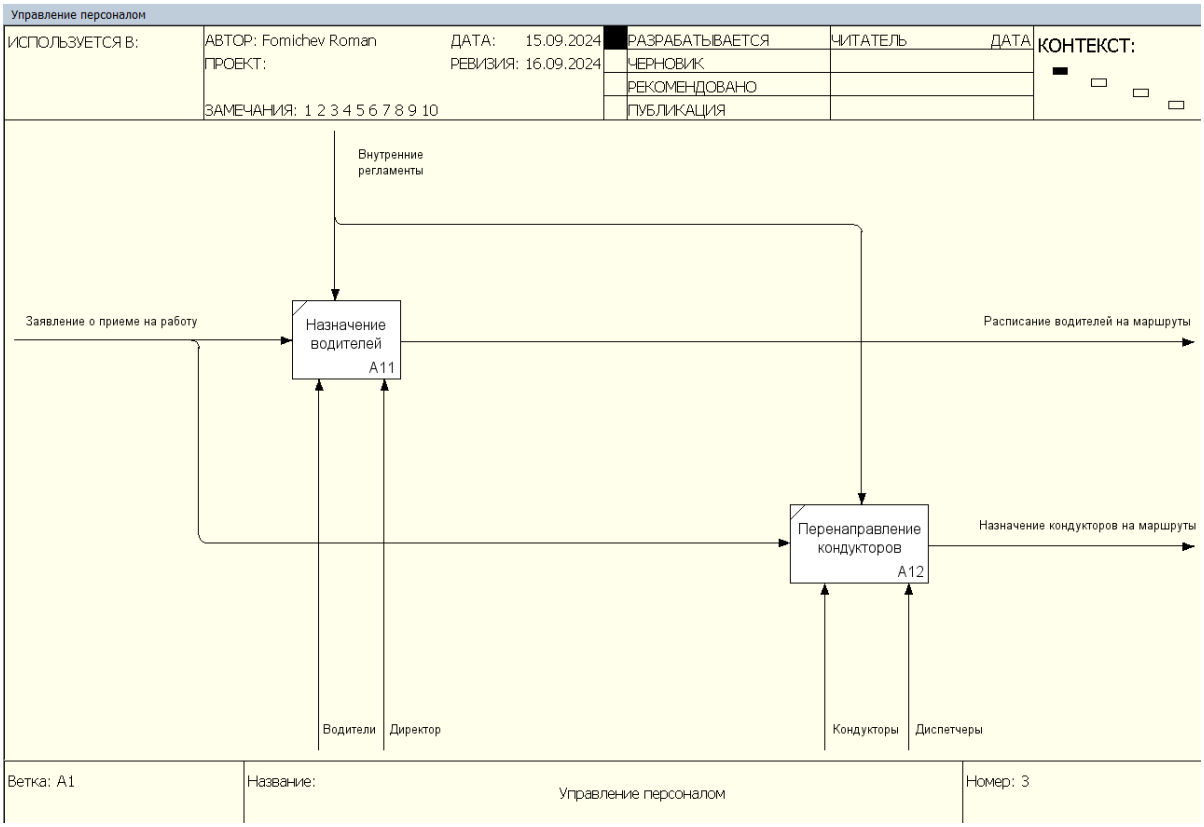

Рисунок 3 – Декомпозиция блока A1

Декомпозиция блока A2:

- Назначение трамваев на маршруты – A21
- Учет поломок и ремонтов – A22

Диаграмма декомпозированного блока представлена на рисунке 4.

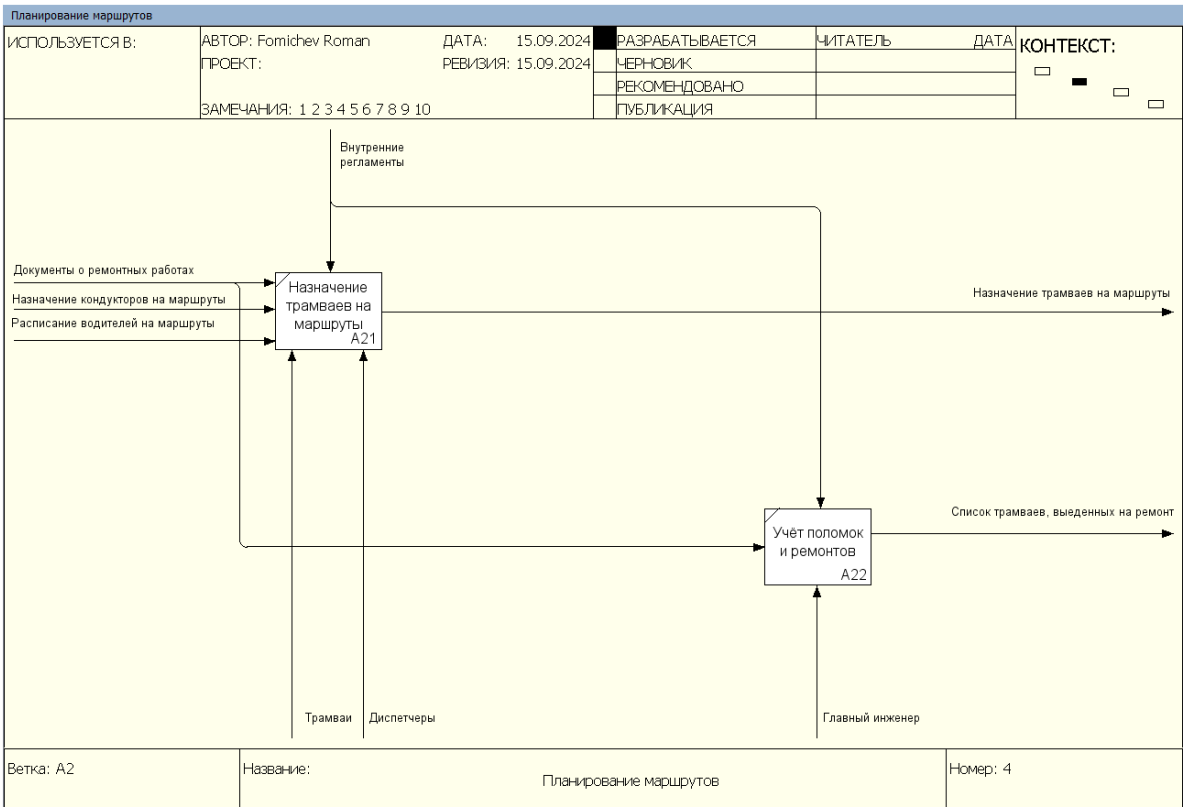

Рисунок 4 – Декомпозиция блока A2

Декомпозиция блока A3:

- Проверка состояния трамваев – A31
- Ремонт неисправных трамваев – A32

Диаграмма декомпозированного блока представлена на рисунке 5.

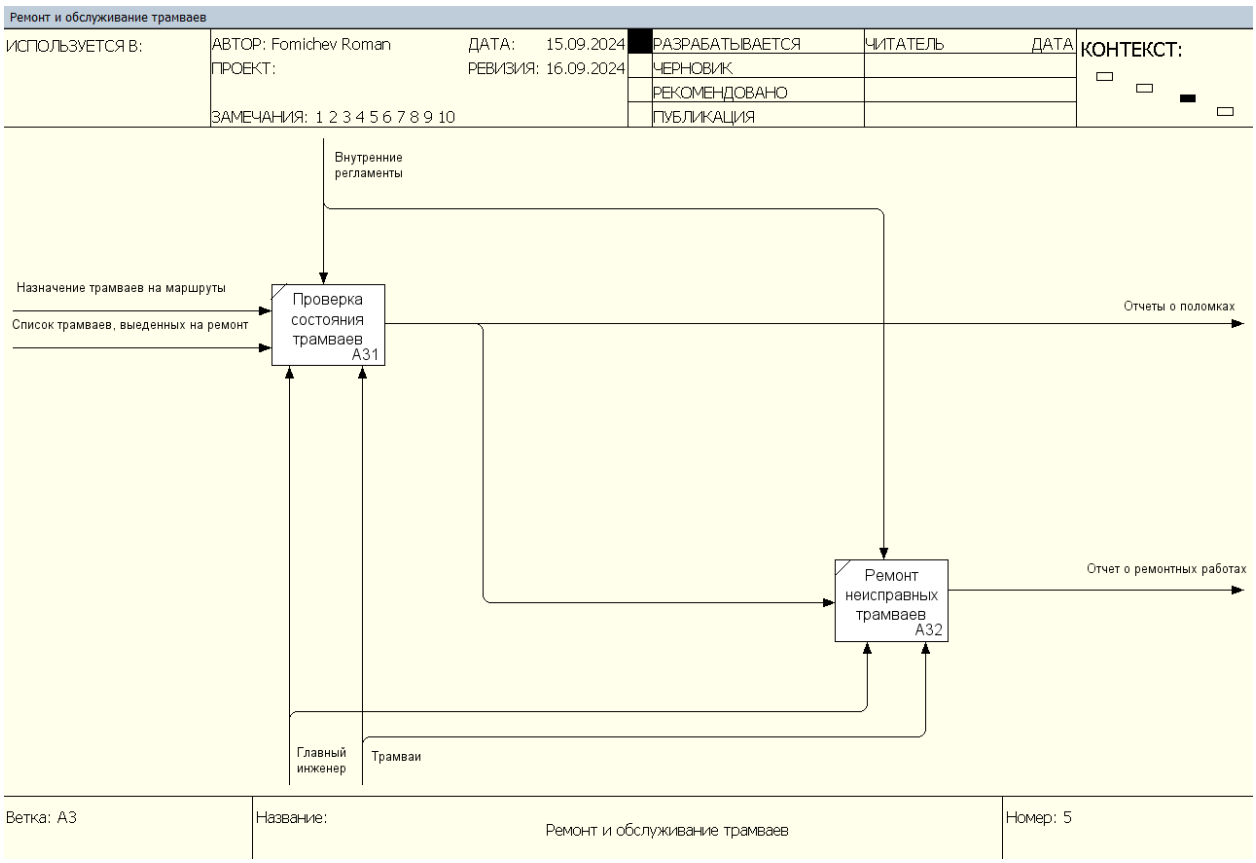

Рисунок 5 – Декомпозиция блока A3

Декомпозиция блока A4:

- Учет выручки с маршрутов – A41
- Финансовая отчетность – A42

Диаграмма декомпозированного блока представлена на рисунке 6.

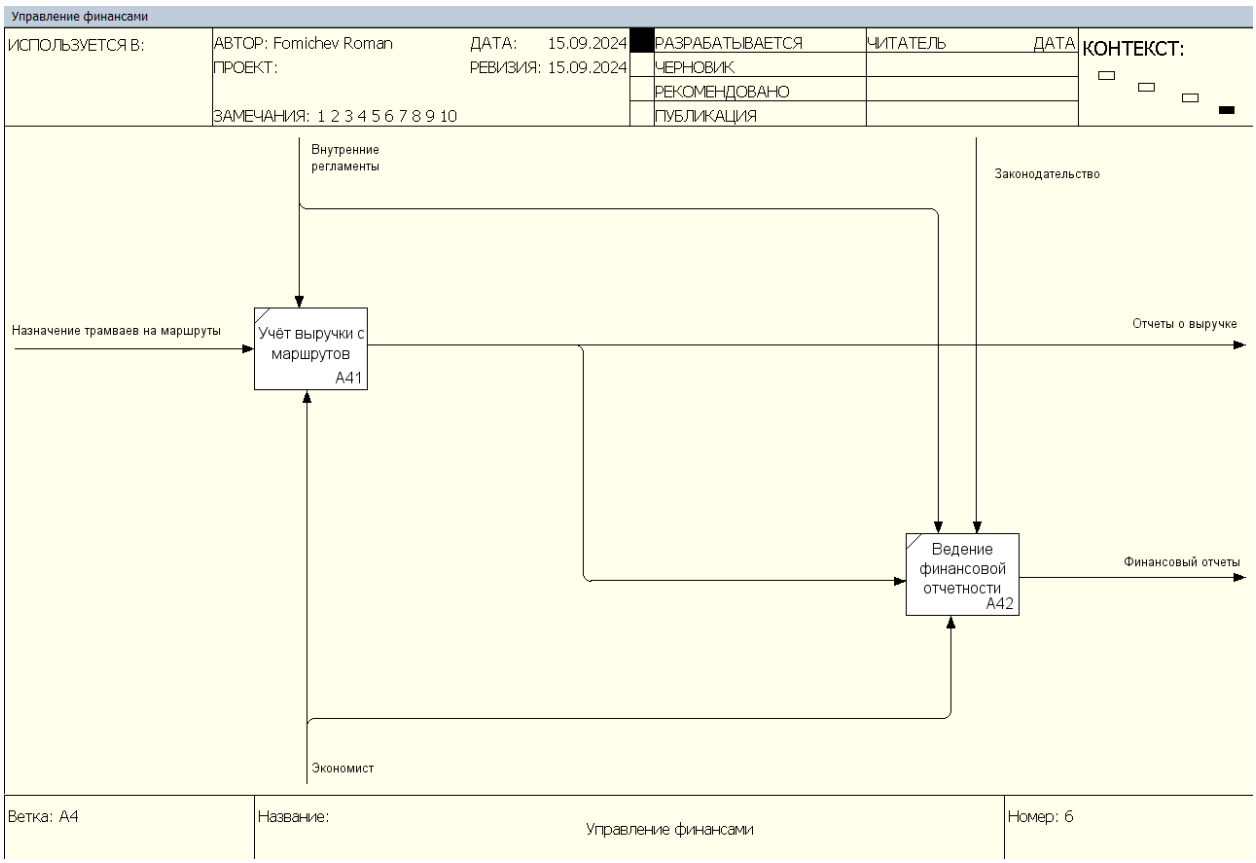

Рисунок 6 – Декомпозиция блока A4

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.

2 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:

В результате анализа функциональной модели AS-IS, были сделаны выводы, как можно преобразовать модель в модель TO-BE.

Необходимо добавить механизм датчиков в трамвае, чтобы ускорить ремонт вышедших их строя трамваев.

В блоке A4 должен формироваться один общий финансовый отчет.

В блоке A0 должны измениться данные выхода и механизмов.

На рисунках 7 – 9 представлены обновленные данные.

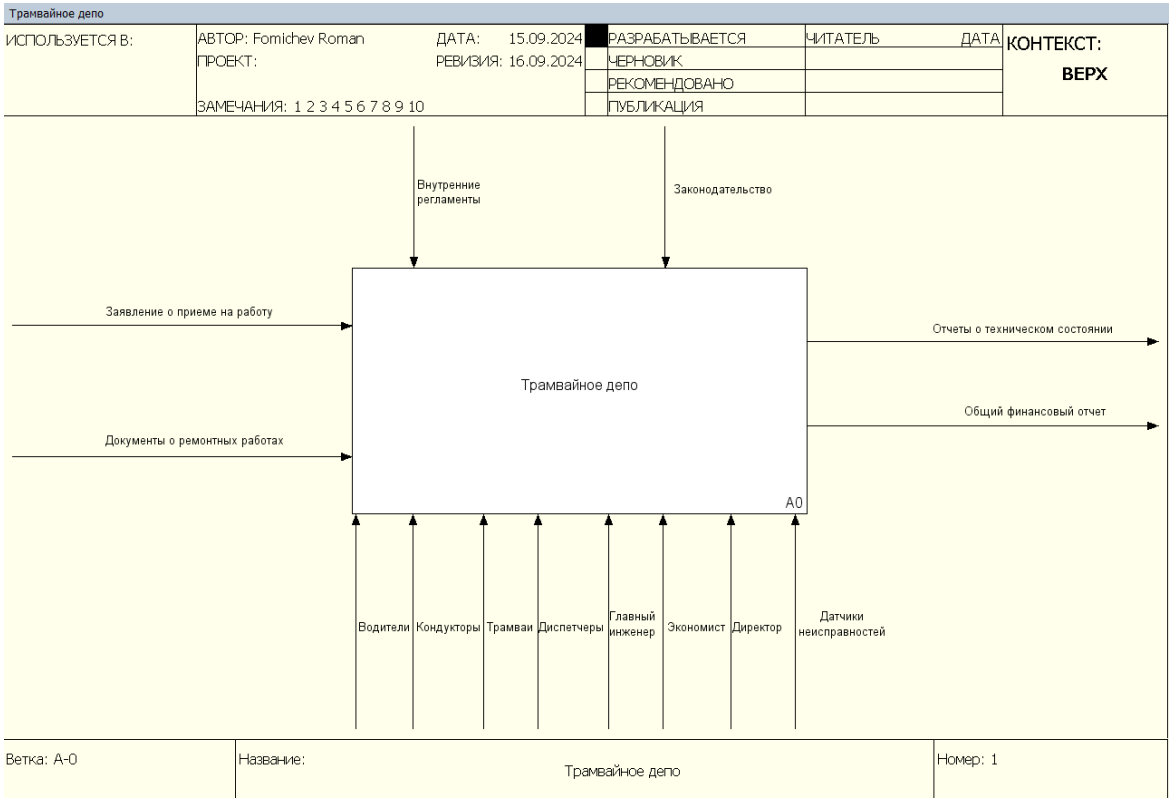

Рисунок 7 – Обновленный блок A0

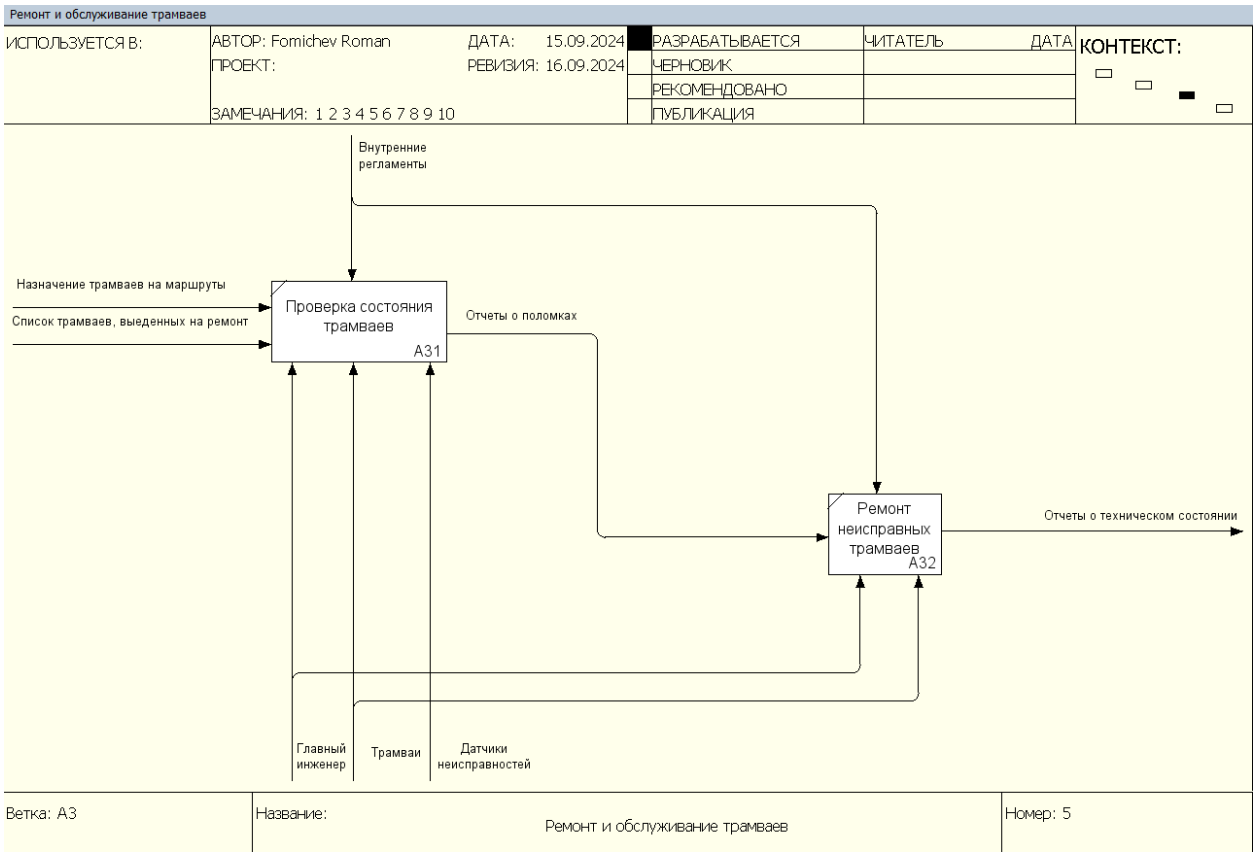

Рисунок 8 – Обновленная декомпозиция блока A3

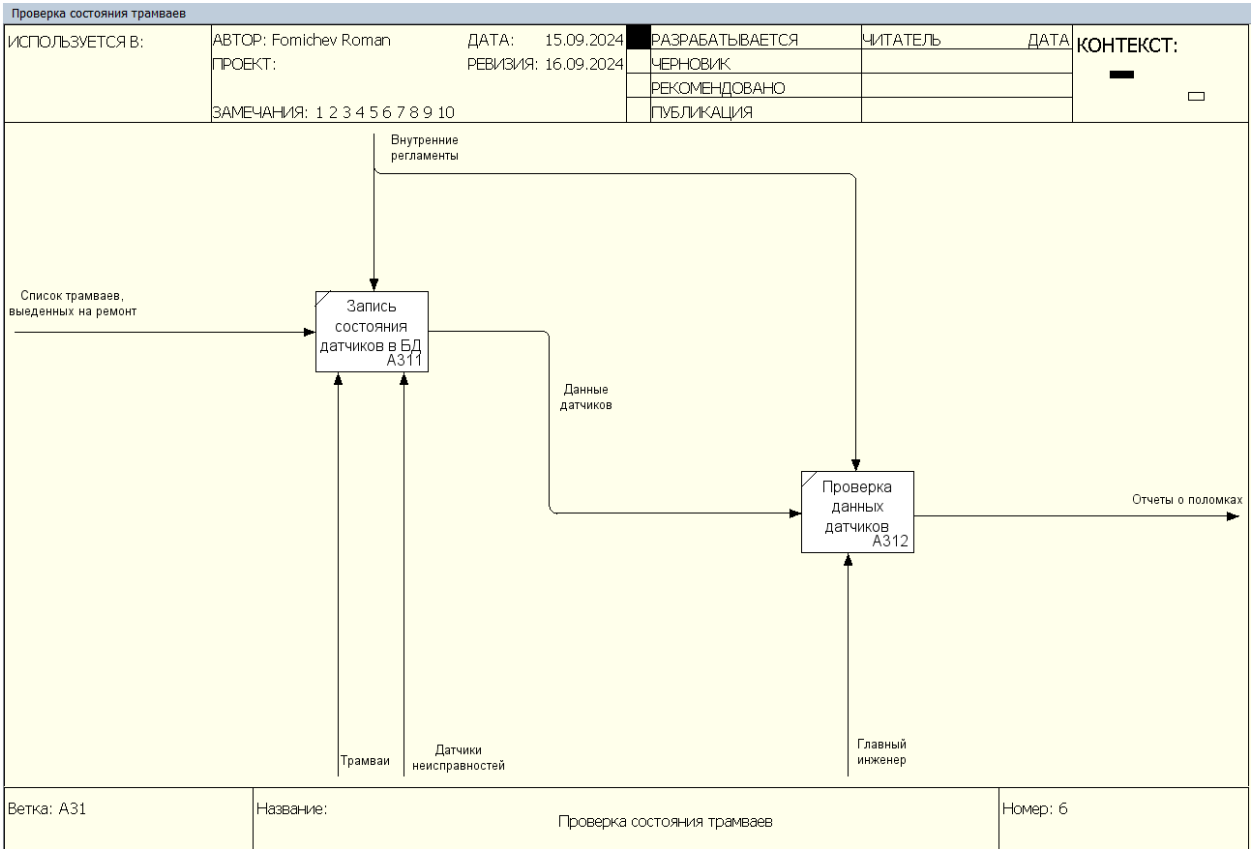

Рисунок 9 – Декомпозиция блока A31

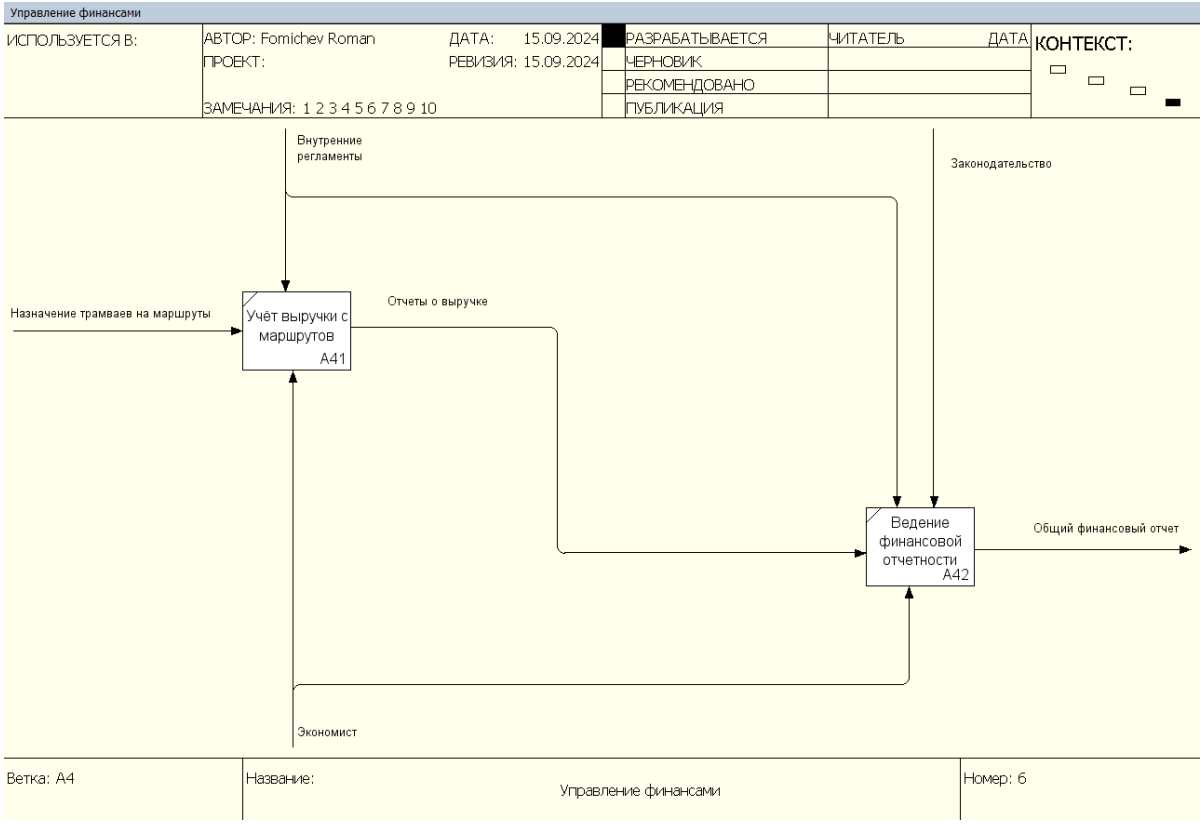

Рисунок 10 – Обновленная декомпозиция блока A4

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.

3 Практическая работа №3

Цель работы:

Получить практические навыки в построении IDEF3-модели процесса.

Постановка задачи:

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

Модель должна содержать:

Перекрестки типа AND, OR, XOR

Ход работы:

Для выполнения практической работы был декомпозирован функциональный блок модели окружения под название «Ремонт и обслуживание трамваев». Была получена модель информационных потоков и взаимоотношений между процессами обработки информации. Модель представлена на рисунке 13.

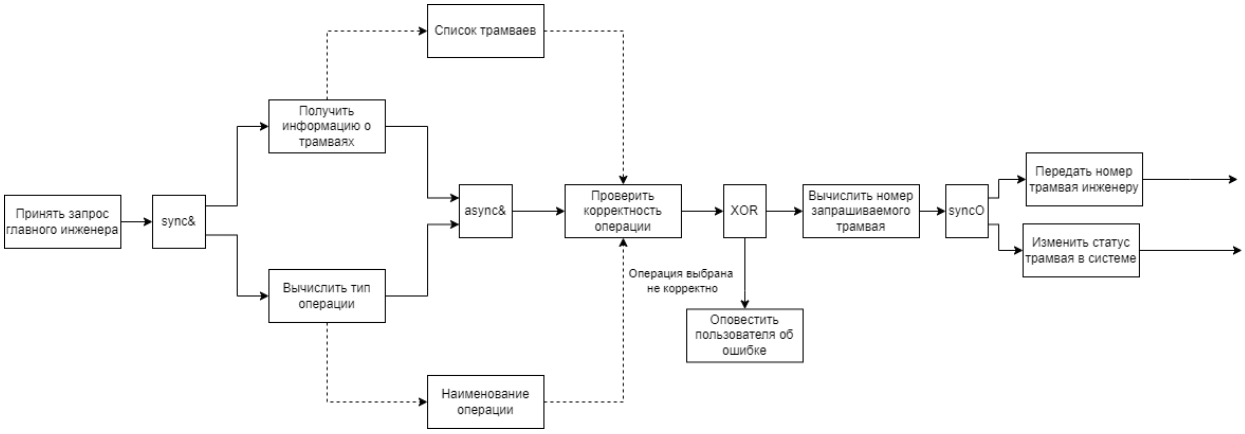

Рисунок 13 – Диаграмма WorkFlow, декомпозиция блока «Ремонт и обслуживание трамваев»

Вывод:

В результате выполнения практической работы были получены теоретические и практические знания в области диаграмм WorkFlow, а также была разработана эта диаграмма в нотации IDEF3.

4 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 выполнения практической работы был декомпозирован процесс «Ремонт и обслуживание трамваев». В результате была получена модель бизнес-процесса, описывающая потоки информации, перемещающие между различными процессами в рамках обработки запроса пользователя. Модель представлена на рисунке 14.

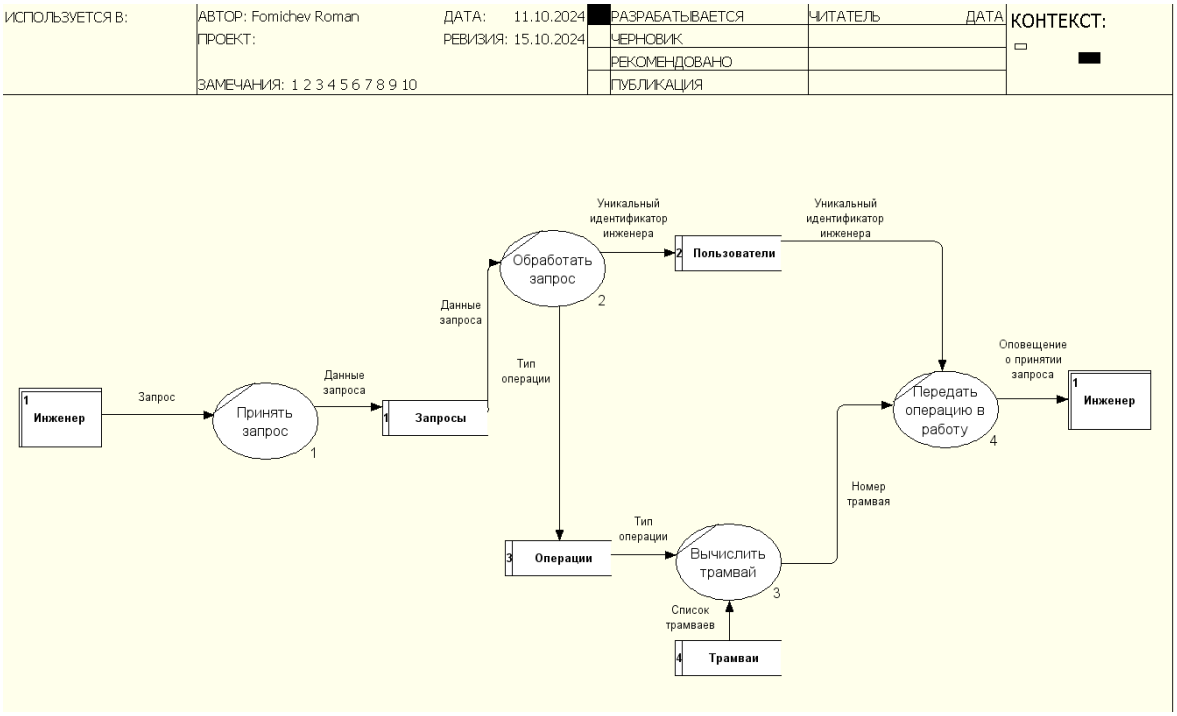

Рисунок 14 – DFD-модель бизнес-процесса «Ремонт и обслуживание трамваев»

Вывод:

В результате выполнения практической работы были получены теоретические и практический знания в области диаграмм потоков данных, а также была разработана модель бизнес-процесса с помощью методологии DFD.

5 Практическая работа №5

Цель работы: получить практические навыки в построении прецедентной UML-модели бизнес-процесса.

Порядок выполнения работы.

Выберите бизнес-процесс, для которого будете формировать модель. Вы можете выбрать один из вариантов процессов, описанных в приложении, или предложить свой вариант. Можно выбрать один из процессов, для которого на предыдущих лабораторных работах строилась модель по одной из структурных методологий. Желательно, чтобы процесс имел различные версии, т.е. альтернативные потоки событий.

Для заданной предметной области:

- построить диаграмму классов;
- построить диаграмму последовательности;
- построить диаграмму взаимодействий (диаграмму коммуникаций);
- построить диаграмму пакетов;

Ход работы:

Для выполнения данной практической работы был выбран процесс ремонта и обслуживания трамвая.

При выполнении данной работы была построена диаграмма классов, используемых в данном процессе, изображенная на рисунке 15.

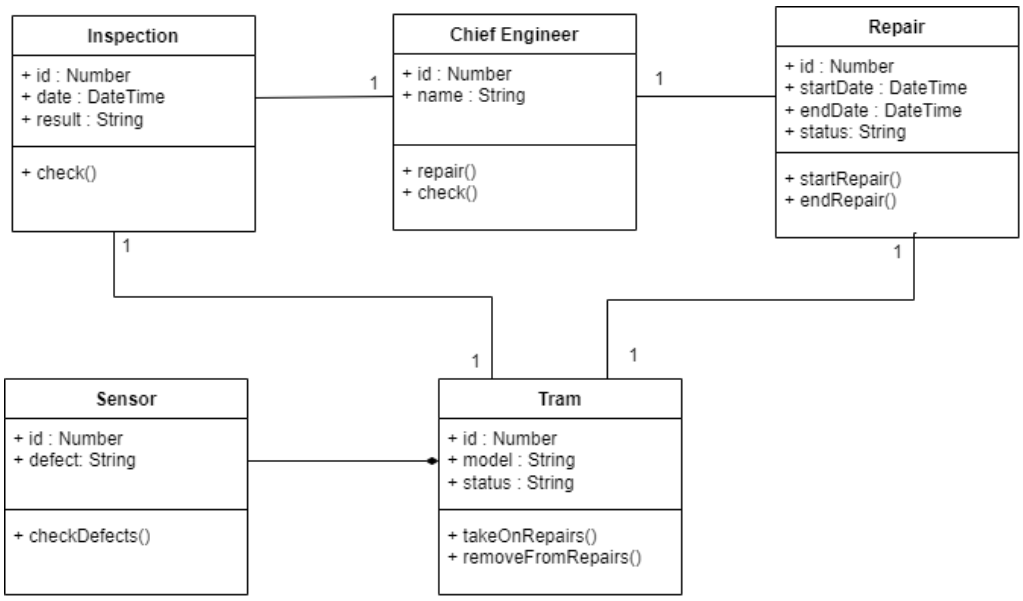

Рисунок 15 – Диаграмма классов

Также была построена диаграмма последовательности для этого процесса, изображенная на рисунке 16.

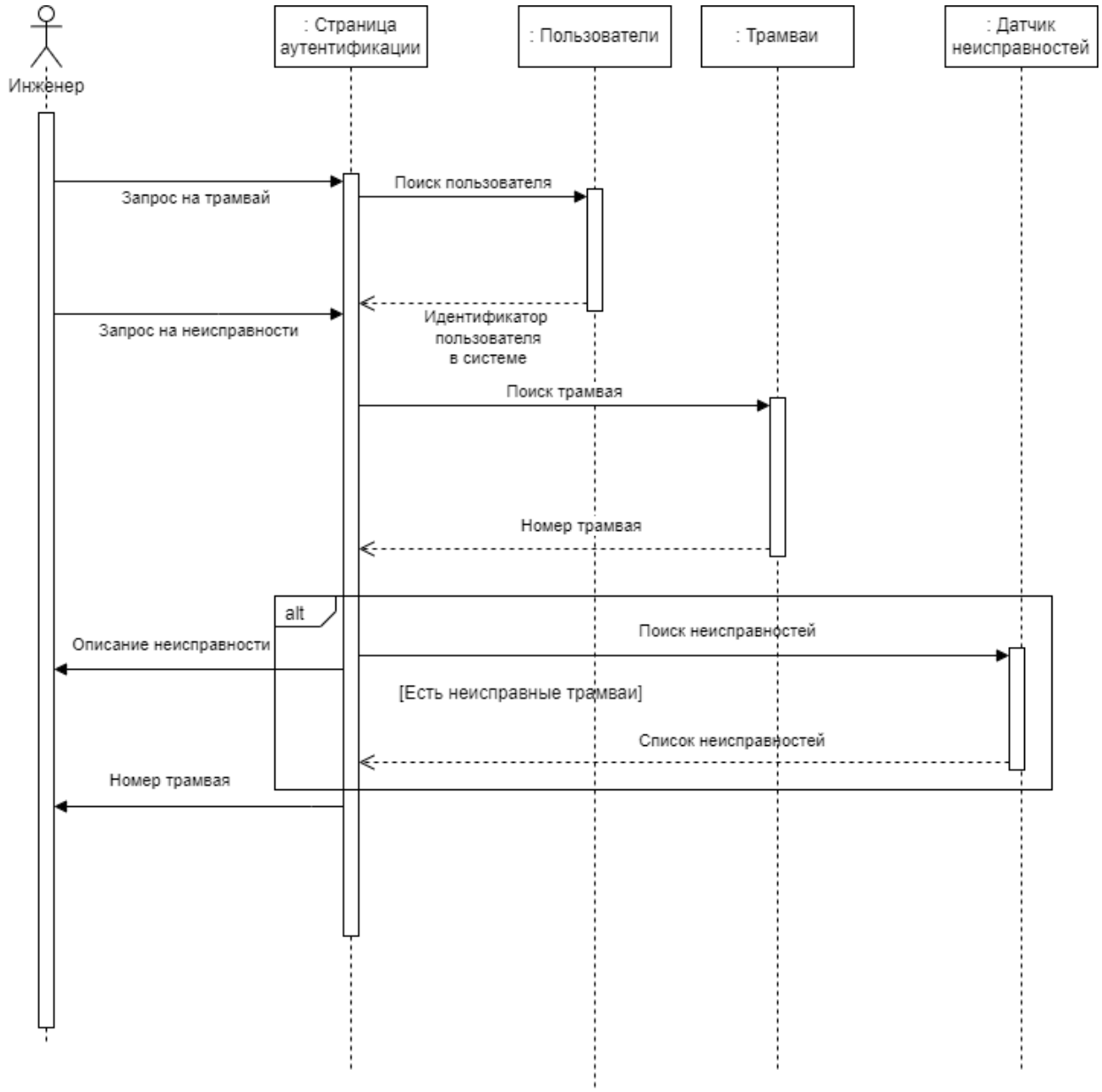

Рисунок 16 – Диаграмма последовательности процесса обработки запроса посетителя

Также для данного процесса были сформированы диаграммы обзора взаимодействий и пакетов, представленные на рисунках 17 – 18 соответственно.

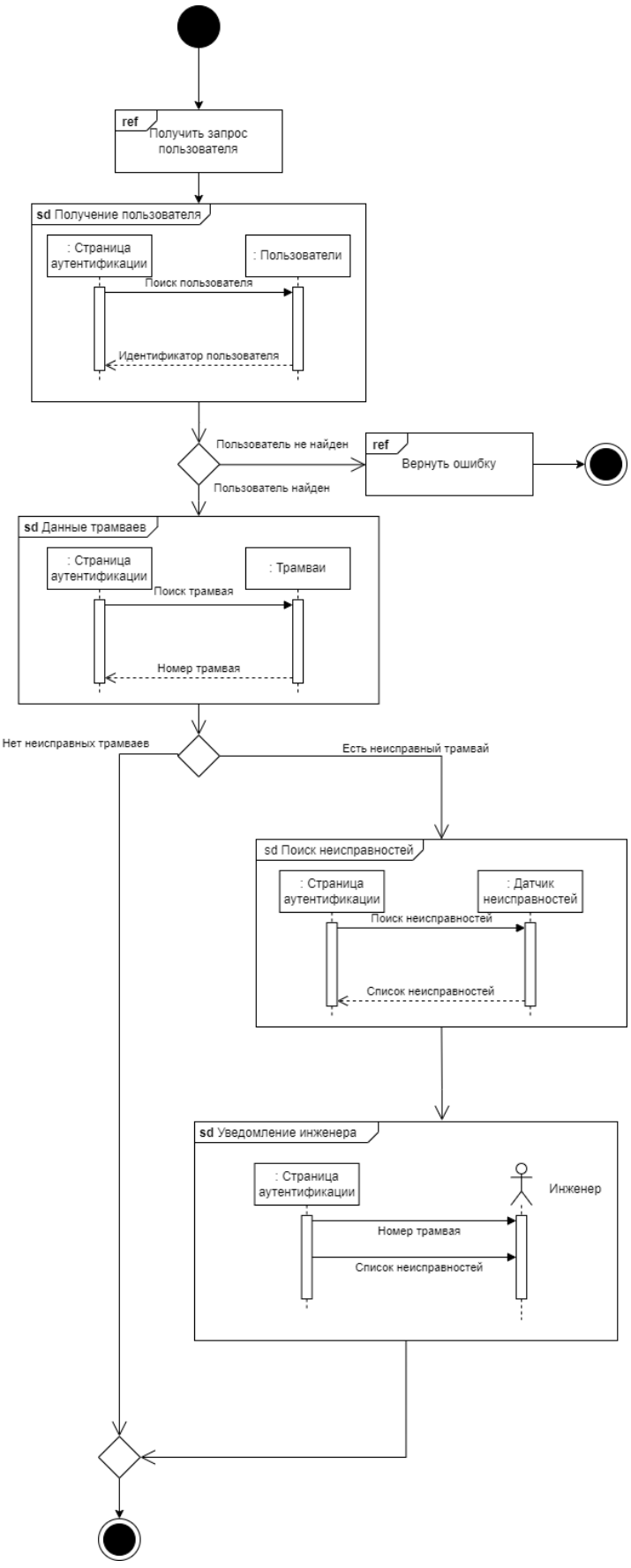

Рисунок 17 – Диаграмма обзора взаимодействий

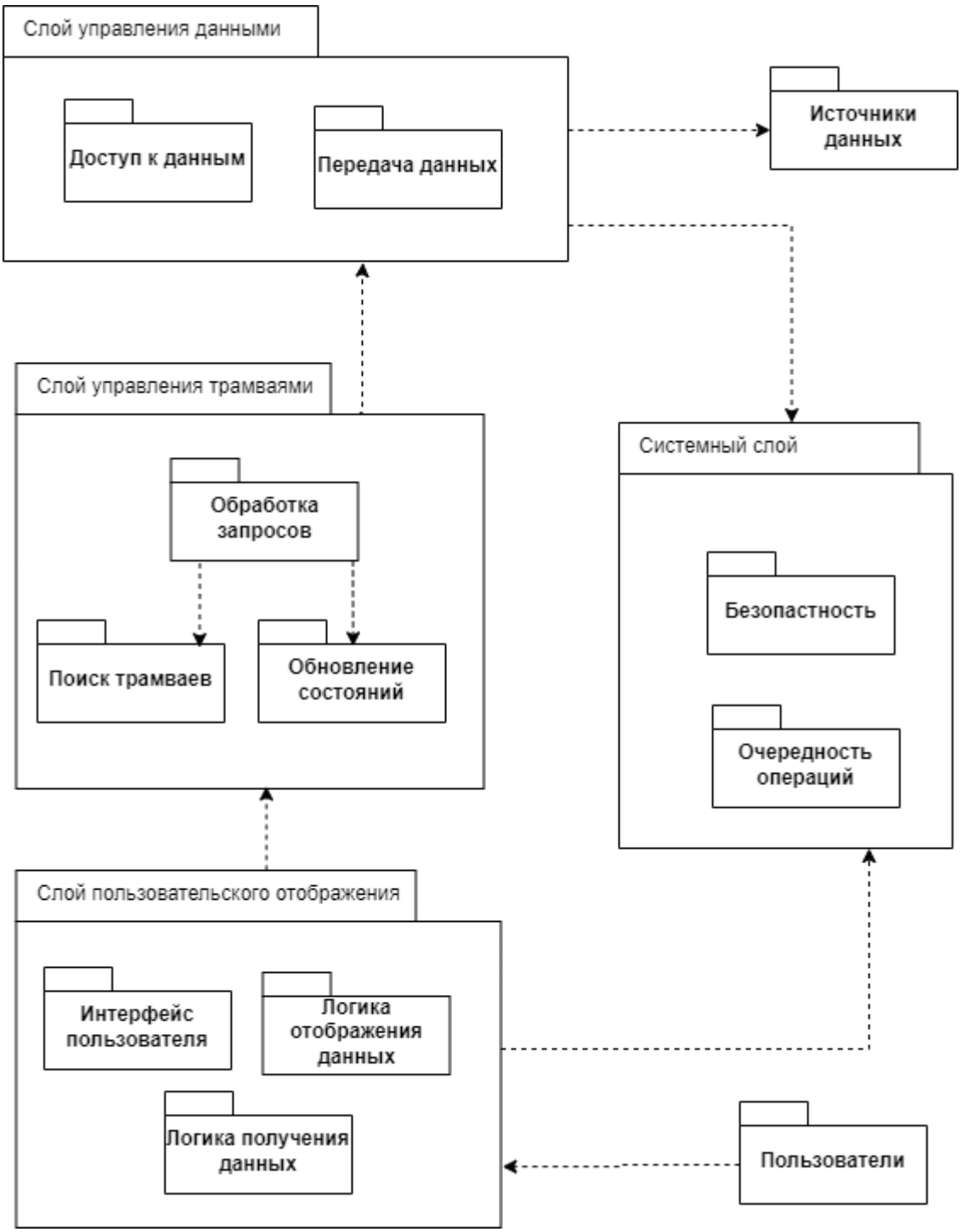

Рисунок 18 – Диаграмма пакетов

Вывод:

В ходе выполнения данной практической работы были получены навыки создания различных UML диаграмм, которые были применены на практике при создании диаграмм пакетов, обзора взаимодействий, последовательности, классов.

6 Практическая работа №6

Задание:

Рассмотрите факторы, которые могут повлиять на масштабируемость системы, такие как архитектура, базы данных, алгоритмы, и другие.

Рассмотрите, какие изменения в архитектуре или инфраструктуре могут потребоваться для обеспечения требуемой масштабируемости.

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

Предложите стратегии масштабируемости, такие как использование облачных ресурсов, контейнеризация, кэширование, балансировка нагрузки и другие могут быть применены в вашем проекте.

Раскройте необходимые риски непосредственно в рамках вашего проекта.

Рассчитайте какое количество пользователей или запросов система должна обслуживать в вашем проекте?

Ход работы:

В процессе анализа работы системы управления ремонтами и обслуживанием трамваев были рассмотрены различные стратегии её функционирования, а также проведён расчет предполагаемых нагрузок на систему. Было выявлено, что система в депо среднего размера должна обеспечивать обработку до 5000 активных операций в сутки, связанных с управлением ремонтами и проверкой состояния трамваев. Каждая операция пользователя (инженера) включает минимум два дополнительных запроса – принятие данных о неисправностях и отправку уведомлений о выполнении работ.

Для обеспечения стабильной работы системы в крупных депо, где требуется быстрая обработка большого количества заявок, особенно в пиковые часы нагрузки (например, утренние и вечерние часы), важно поддерживать цикл обработки каждой операции в пределах 30-60 секунд. Это позволит значительно повысить эффективность системы по сравнению с традиционными методами управления процессами ремонта и обслуживания.

Основным фактором увеличения ежедневной нагрузки на систему является подключение новых депо и увеличение парка трамваев. Это может привести к резкому росту числа пользователей и запросов, что может вызвать перегрузку системы, если прогнозы по нагрузке будут занижены. Ограничивающими факторами масштабируемости являются архитектура системы, влияющая на скорость выполнения запросов, медленная работа баз данных, а также нехватка технических ресурсов (например, серверов).

Одним из решений для повышения производительности системы является переход на микросервисную архитектуру. Разделение задач системы (например, распределение трамваев по ремонтам, аутентификация пользователей, контроль технического состояния) на отдельные сервисы позволит равномерно распределить нагрузку между разными серверами и уменьшить зависимость от недостатка вычислительных мощностей.

Однако микросервисная архитектура должна быть устойчива к возможным сбоям, таким как выход из строя отдельных сервисов или нехватка ресурсов для их работы. Для этого предлагается использовать контейнеризацию, которая позволит оперативно разворачивать нужные сервисы и быстро восстанавливать их работоспособность. Контейнеры также могут содержать локальные базы данных, что ускорит доступ к важной информации и снизит время отклика.

Кроме того, для повышения масштабируемости системы рекомендуется реализовать кэширование часто запрашиваемых данных (например, списки трамваев, сотрудников депо или запчастей), распределение нагрузки между

несколькими экземплярами сервисов с помощью таких инструментов, как Kubernetes или Docker, а также, при необходимости, размещение части сервисов на серверах депо. Это повысит безопасность данных и снимет часть нагрузки с централизованной системы, позволяя депо самостоятельно управлять ресурсами.

Несмотря на очевидные преимущества предложенных решений, они также влекут за собой риски, связанные с повышением затрат на обслуживание и обновление системы, особенно в случае её развертывания на локальных серверах. Это может привести к ошибкам в обработке данных или увеличению времени на выполнение операций, что снизит общую эффективность системы по сравнению с ручным управлением.

Вывод:

В рамках выполнения данной практической работы были рассмотрены различные варианты масштабирования системы. В результате была выбрана микросервисная архитектура, с возможностью on-premise установки.

7 Практическая работа №7

В ходе анализа архитектуры системы управления ремонтом и обслуживанием трамваев были рассмотрены три типа архитектур: монолитная, модульная и микросервисная. Каждая из них имеет свои особенности, преимущества и недостатки, которые влияют на возможность модификации и масштабирования системы.

Монолитная архитектура объединяет все компоненты системы (учет неисправностей трамваев, планирование ремонтов, управление техническим обслуживанием) в одно приложение. Основной недостаток такого подхода заключается в сложности внесения изменений и ограничениях в масштабируемости. При росте нагрузки необходимо развертывать несколько полных копий системы, что может привести к перегрузке серверов и затруднить управление ресурсами. Это делает сложнее обновление отдельных функций системы и увеличивает вероятность ошибок при внедрении изменений.

Модульная архитектура, в отличие от монолитной, позволяет разделить систему на независимые модули, такие как управление техническим состоянием, учет неисправностей и ремонтные работы. Каждый модуль может функционировать самостоятельно, что упрощает внесение изменений и снижает риски, связанные с обновлениями. Однако модули по-прежнему остаются частью одного приложения, и для масштабирования требуется запуск дополнительных копий всей системы. Это ограничивает гибкость и может привести к неравномерному распределению нагрузки между модулями.

Микросервисная архитектура, выбранная для анализа, обеспечивает наибольшую гибкость и независимость компонентов системы. Суть микросервисного подхода заключается в том, что система делится на отдельные микросервисы, каждый из которых отвечает за конкретную функцию, например, диагностику неисправностей, учет трамваев или

управление ремонтами. Эти микросервисы взаимодействуют между собой через API, что позволяет масштабировать каждый из них независимо. Например, при увеличении запросов на диагностику неисправностей можно масштабировать только соответствующий микросервис, не затрагивая другие компоненты системы.

Основной риск, связанный с микросервисной архитектурой, заключается в повышенной сложности координации и управления микросервисами. Для решения этой проблемы необходимо использовать системы оркестрации, такие как Kubernetes или Docker Swarm, а также инструменты мониторинга и логирования для отслеживания состояния каждого микросервиса. Также важно внедрить отказоустойчивую архитектуру, которая сможет справляться с выходом из строя отдельных сервисов без нарушения работы всей системы.

Для успешного внедрения микросервисной архитектуры и обеспечения ее масштабируемости предложены следующие стратегии: использование контейнеризации, которая позволит быстро развертывать новые копии сервисов и минимизировать время простоя; внедрение кэширования для часто используемых данных; балансировка нагрузки между несколькими экземплярами сервисов с помощью оркестрации; возможность установки системы на серверы депо для крупных предприятий, что обеспечит снижение времени отклика и повысит уровень безопасности данных.

Выводы:

Таким образом, микросервисная архитектура была выбрана как наилучшее решение для системы трамвайного депо, так как она позволяет эффективно управлять ростом количества пользователей, поддерживает независимое масштабирование сервисов и упрощает модификацию системы.

8 Практическая работа №8

Цель задания:

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

Задание:

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 со здания 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:

Для анализа применения шаблонов и паттернов проектирования в контексте системы трамвайного депо был рассмотрен ряд структурных, поведенческих и порождающих паттернов. Основной целью являлось выявление наиболее подходящих решений, которые помогут повысить гибкость системы, упростить взаимодействие между компонентами и обеспечить эффективное масштабирование.

Поскольку система трамвайного депо построена на микросервисной архитектуре, основное внимание уделено паттернам, которые помогают управлять взаимодействием между независимыми сервисами и обеспечивают надежное и безопасное хранение данных.

1. **Шаблон "Singleton" (Одиночка):** Этот паттерн используется для управления подключением к базе данных, обеспечивая наличие единственного экземпляра подключения. Это позволяет исключить возможность создания нескольких подключений к базе данных, что минимизирует риск конфликтов и избыточных запросов. Реализация Singleton позволяет централизовать управление доступом к базе данных, повысить производительность системы и снизить затраты на ресурсы.

2. **Шаблон "Factory Method" (Фабричный метод):** В системе трамвайного депо этот паттерн применяется для создания объектов, таких как учетные записи пользователей или записи о ремонтных работах. Паттерн

позволяет создавать объекты на основе переданных параметров, обеспечивая гибкость в расширении системы. Например, при необходимости добавления нового типа трамвая или категории предмета не потребуется вносить изменения в код логики, а достаточно будет добавить новую фабрику, которая будет создавать соответствующие объекты. Это упрощает расширение системы и повышает её гибкость.

3. **Шаблон "Observer" (Наблюдатель):** Данный паттерн используется для отслеживания изменения состояния трамваев и уведомления всех заинтересованных микросервисов о событиях. Например, при изменении состояния трамвая (отремонтирован или нуждается в ремонте) соответствующие микросервисы (управление тех обслуживания трамваев, уведомление главного инженера) получают уведомления и могут предпринять дальнейшие действия. Паттерн "Наблюдатель" позволяет минимизировать взаимодействие между сервисами, так как каждый микросервис может подписаться только на те изменения, которые его касаются, что снижает нагрузку на систему и увеличивает её отзывчивость.

4. **Шаблон "Adapter" (Адаптер):** Использование паттерна "Adapter" позволяет интегрировать систему трамвайного депо с внешними API и системами, такими как учетные системы пользователей. Адаптеры предоставляют единый интерфейс для взаимодействия с различными системами, что упрощает добавление новых интеграций и минимизирует изменения в исходном коде при замене внешних сервисов.

5. **Паттерн "Facade" (Фасад):** Для упрощения взаимодействия между различными микросервисами используется паттерн "Facade". Он объединяет несколько мелких сервисов в единую точку входа, предоставляя упрощенный интерфейс для выполнения комплексных операций. Это снижает количество прямых взаимодействий между микросервисами, упрощает координацию и уменьшает вероятность ошибок при вызове сервисов. Например, фасад может объединять операции авторизации новых пользователей и проверки состояния трамваев в одном запросе.

6. **Паттерн "Circuit Breaker" (Размыкатель цепи):** Паттерн "Circuit Breaker" используется для управления сбоями и ошибками при взаимодействии между микросервисами. Если один из сервисов временно недоступен или работает некорректно, "Circuit Breaker" позволяет временно остановить отправку запросов к этому сервису и перенаправить их на резервный сервис или вернуть сообщение об ошибке пользователю. Это снижает риск перегрузки системы и предотвращает цепную реакцию отказов.

Для реализации вышеописанных паттернов в системе трамвайного депо предлагается следующая структура:

- **Singleton:** использовать для создания единого подключения к базе данных, которое будет управлять всеми операциями чтения и записи.
- **Factory Method:** применить для создания объектов при регистрации новых пользователей или добавлении новых трамваев.
- **Observer:** реализовать для уведомления микросервисов о событиях изменения состояния трамваев.
- **Adapter:** внедрить для взаимодействия с внешними системами.
- **Facade:** создать фасадные сервисы, объединяющие несколько микросервисов, чтобы сократить количество запросов и улучшить координацию между ними.
- **Circuit Breaker:** включить в реализацию микросервисов для управления сбоями и повышения отказоустойчивости системы.

Диаграммы представлены на рисунках 19 - 21.

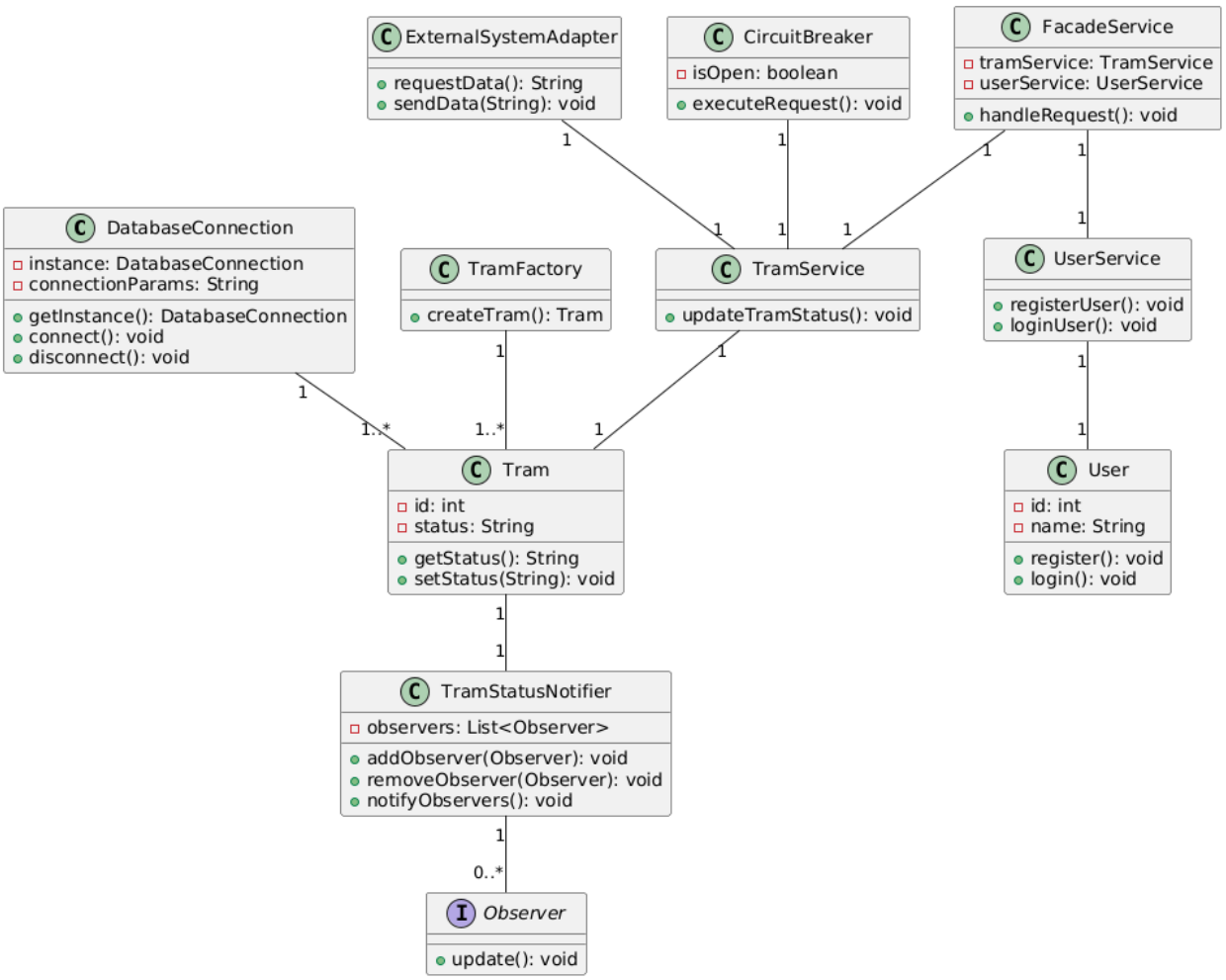

Рисунок 19 – Диаграмма классов

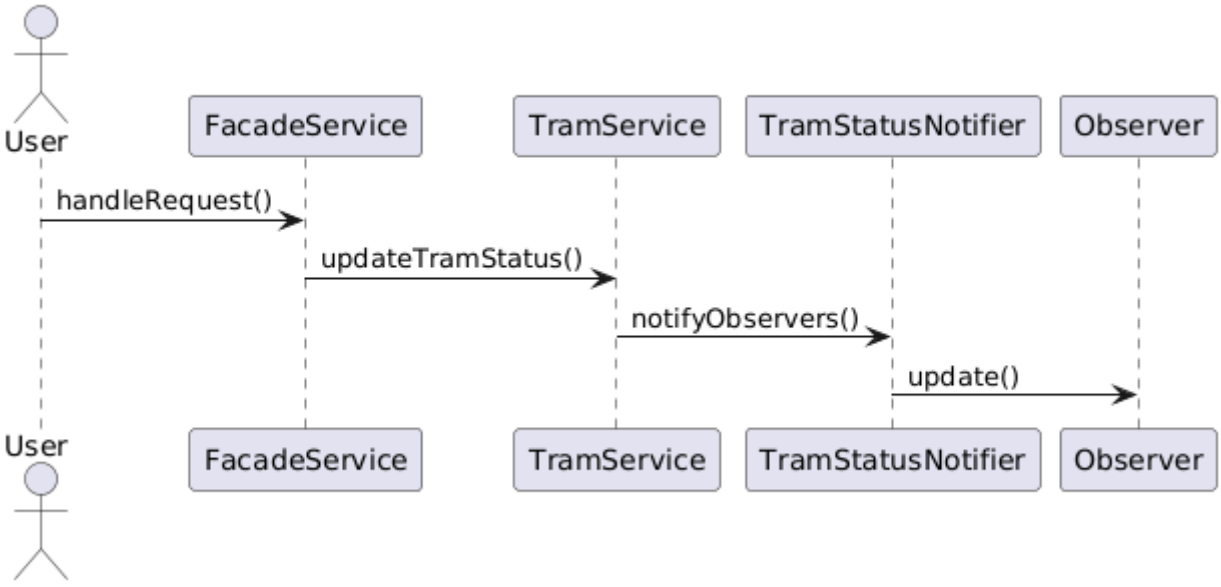

Рисунок 20 – Диаграмма последовательности

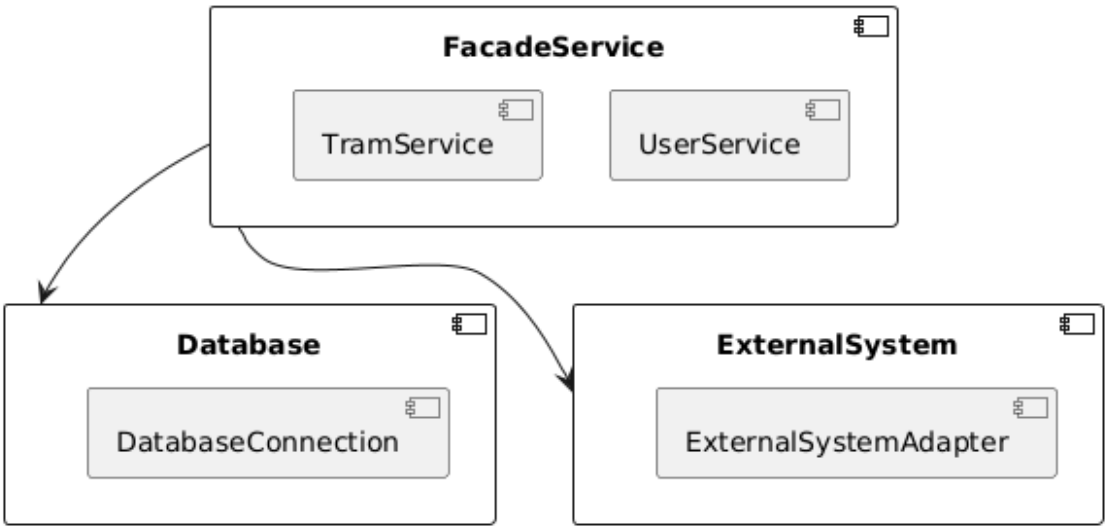

Рисунок 21 – Диаграмма компонентов

Вывод:

Применение данных паттернов позволит существенно упростить архитектуру системы, обеспечить её гибкость и масштабируемость, а также снизить риски при интеграции с внешними сервисами и разработке новых функциональных компонентов. В результате выполнения всех практических работ была спроектирована система для проекта трамвайного депо, ядром которой станет микросервисная архитектура с использованием различных паттернов для облегчения функционирования, поддерживания и обновления.

9 Список литературы

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.