


Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования
Кафедра проектирования информационно-компьютерных систем
Рефакторинг и оптимизация программного кода

Отчет
по лабораторной работе №1
на тему:

РЕФАКТОРИНГ ПРОГРАММНОГО КОДА

Проверил	<u>10 (десять)</u> (подпись)	А.В. Шелест
зачтено	<u>11.09.2025</u> (дата защиты)	
Выполнил	<u></u> (подпись)	П.А. Карлюк гр. 214302

Минск, 2025

СОДЕРЖАНИЕ

1 Ссылка на репозиторий <i>GitHub</i>	3
2 Архитектура ПС в нотации C4-модель	3
3 Система дизайна пользовательского интерфейса	9
4 Диаграмма классов и интерфейсов	11
5 Вывод	17

1 ССЫЛКА НА РЕПОЗИТОРИЙ GITHUB

Ссылка на репозиторий сервера: <https://github.com/239fd/wmsProject/>.
Ссылка на репозиторий клиента: <https://github.com/239fd/wmsProjectClient>.

2 АРХИТЕКТУРА ПС В НОТАЦИИ C4-МОДЕЛЬ

Модель *C4* – подход к моделированию архитектуры системы, в котором основной акцент нацелен на простоту нотации и различные уровни абстракции.

На рисунке 1 представлен контейнерный уровень архитектуры программного средства в нотации *C4*.

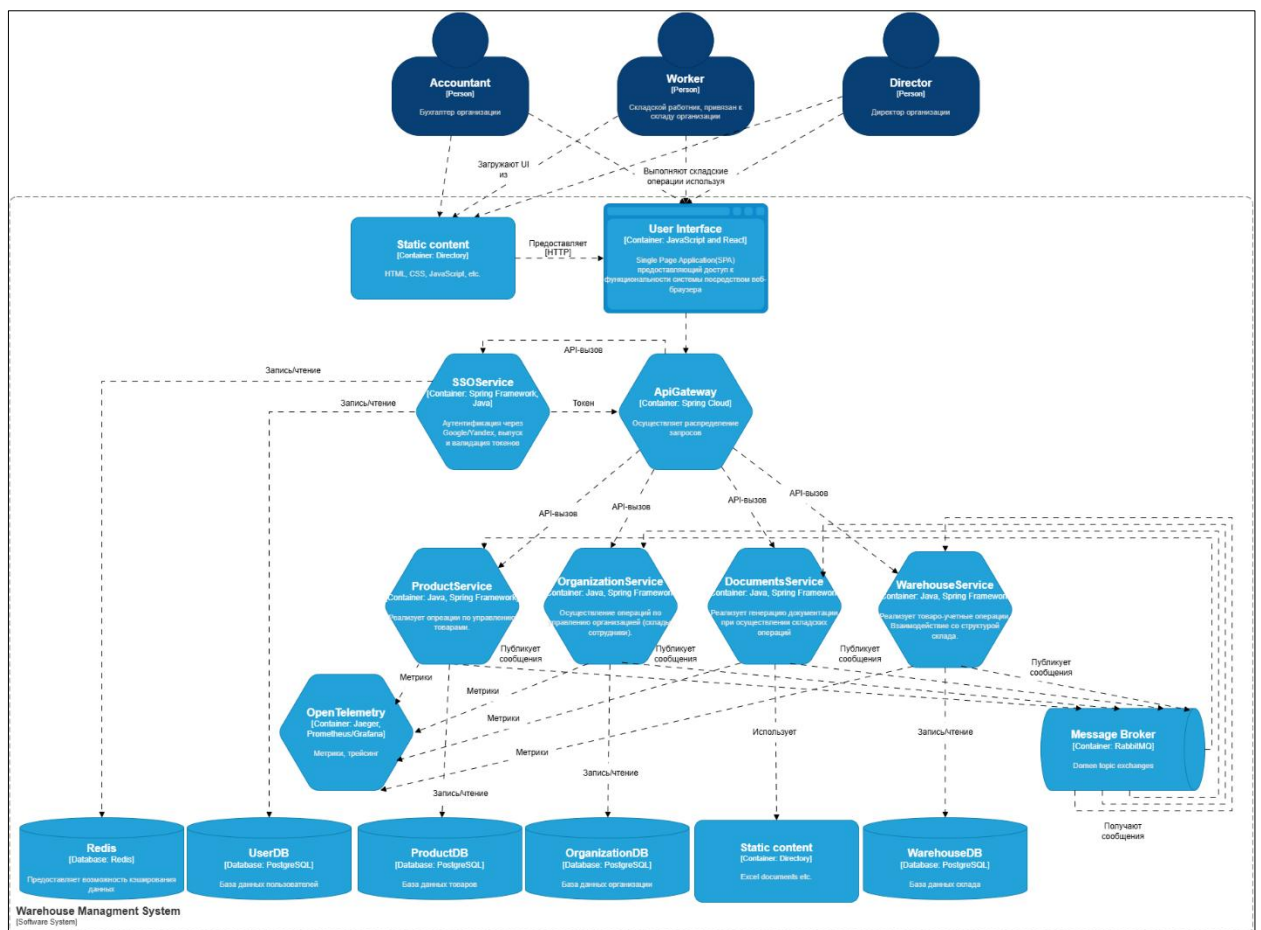


Рисунок 1 – Контейнерный уровень архитектуры программного средства

Далее на рисунке 2 представлен компонентный уровень архитектуры программного средства в нотации *C4*.

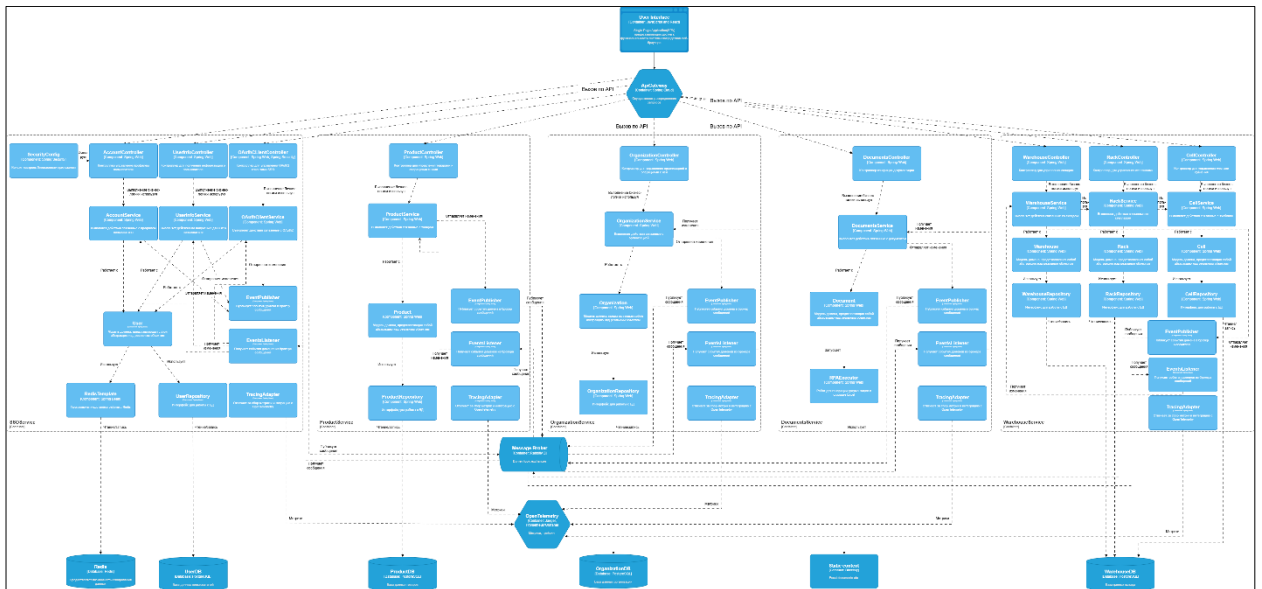


Рисунок 2 – Компонентный уровень архитектуры программного средства

Далее на рисунке 3 более подробно представлен компонентный уровень архитектуры *SSOService* в нотации *C4*.

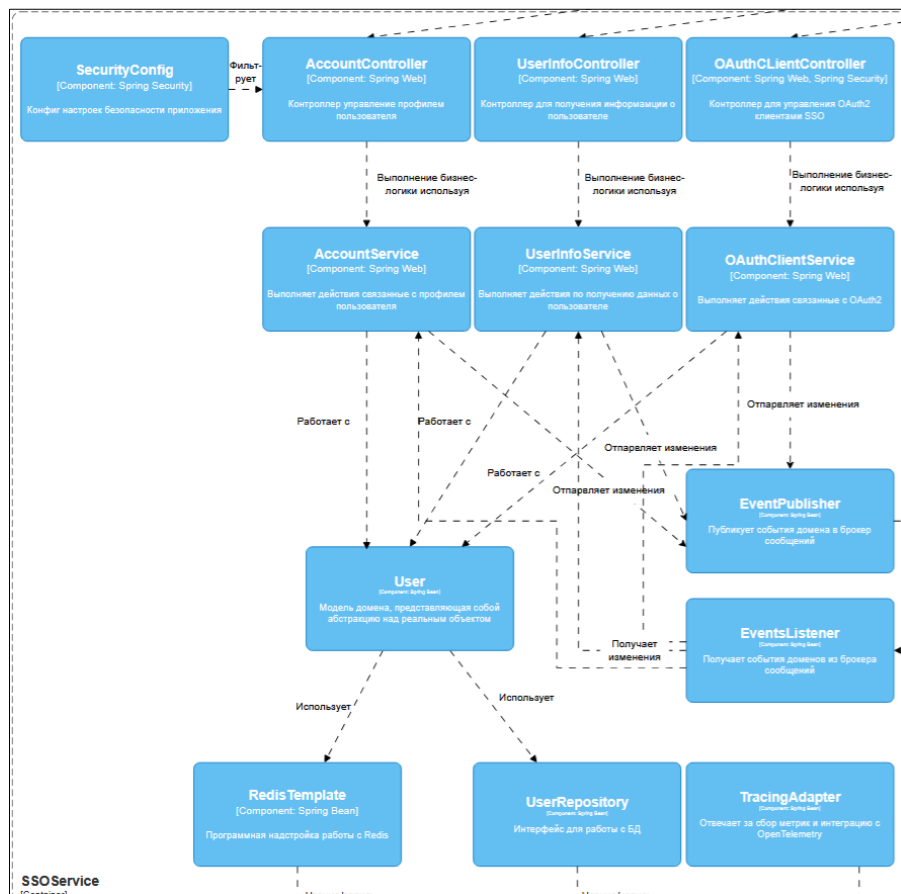


Рисунок 3 – Компонентный уровень архитектуры *SSOService*

Далее на рисунке 4 более подробно представлен компонентный уровень архитектуры *ProductService* в нотации *C4*.

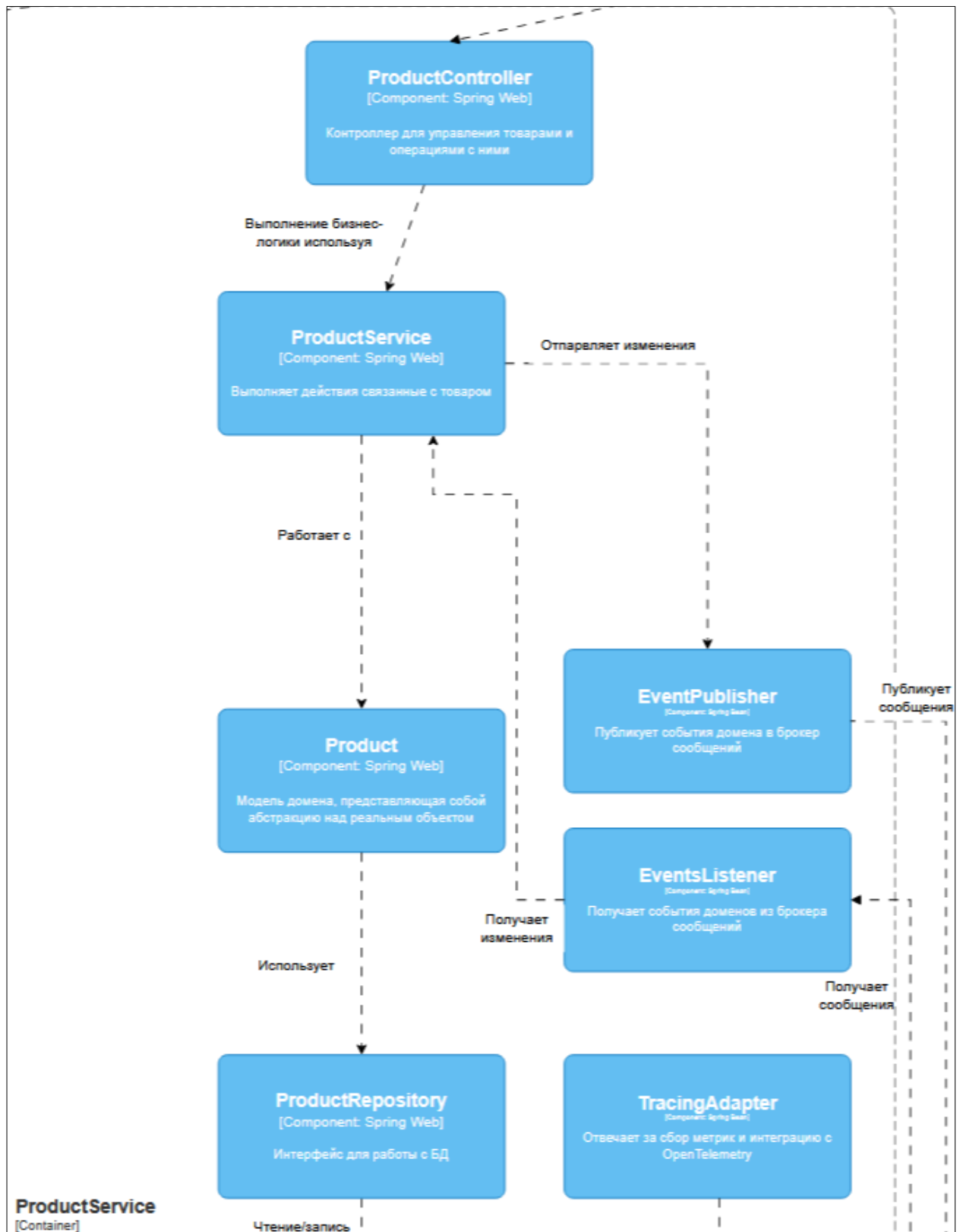


Рисунок 4 – Компонентный уровень архитектуры *ProductService*

Далее на рисунке 5 более подробно представлен компонентный уровень архитектуры *OrganizationService* в нотации C4.

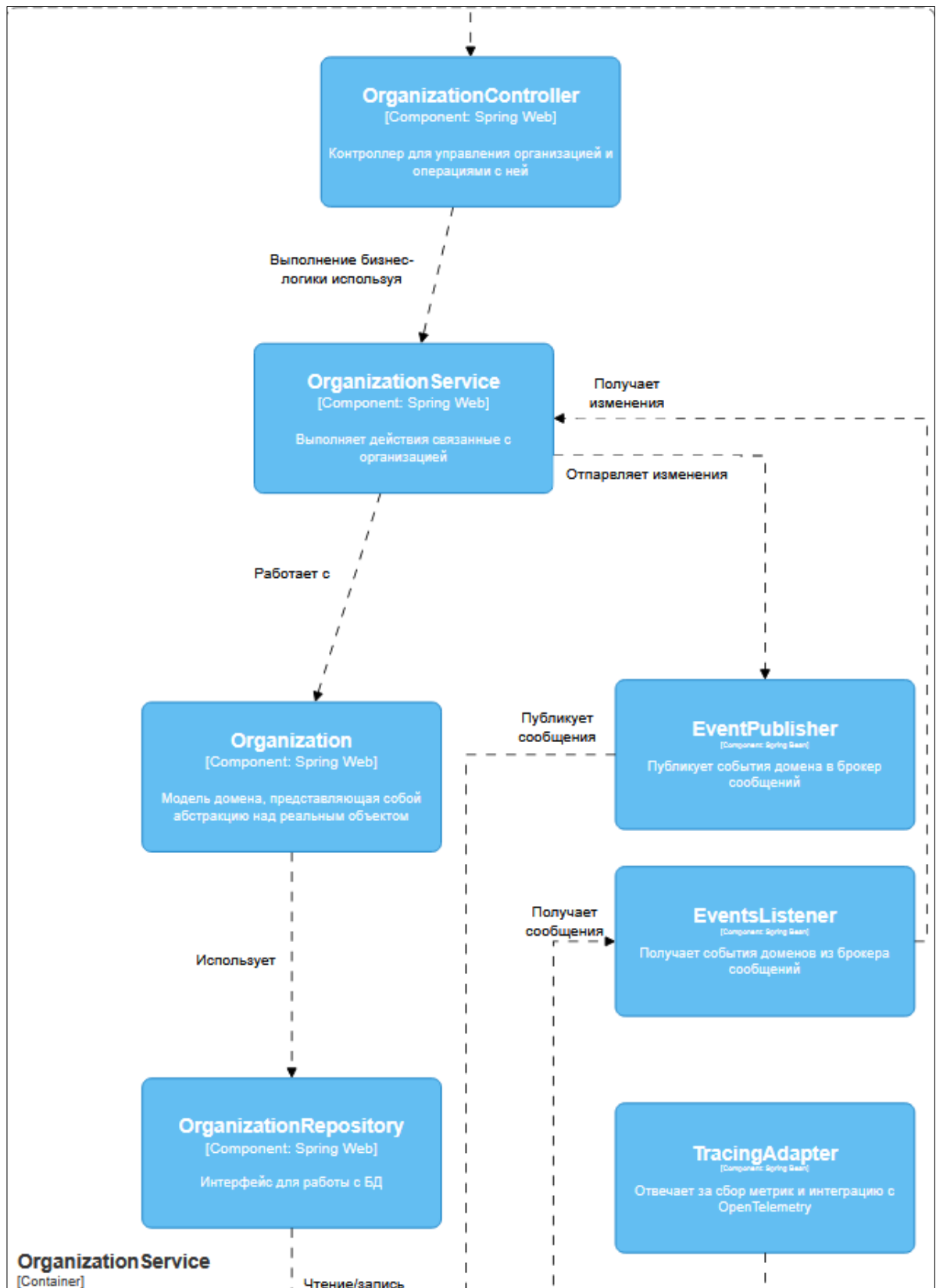


Рисунок 5 – Компонентный уровень архитектуры *OrganizationService*

Далее на рисунке 6 более подробно представлен компонентный уровень архитектуры *DocumentsService* в нотации C4.

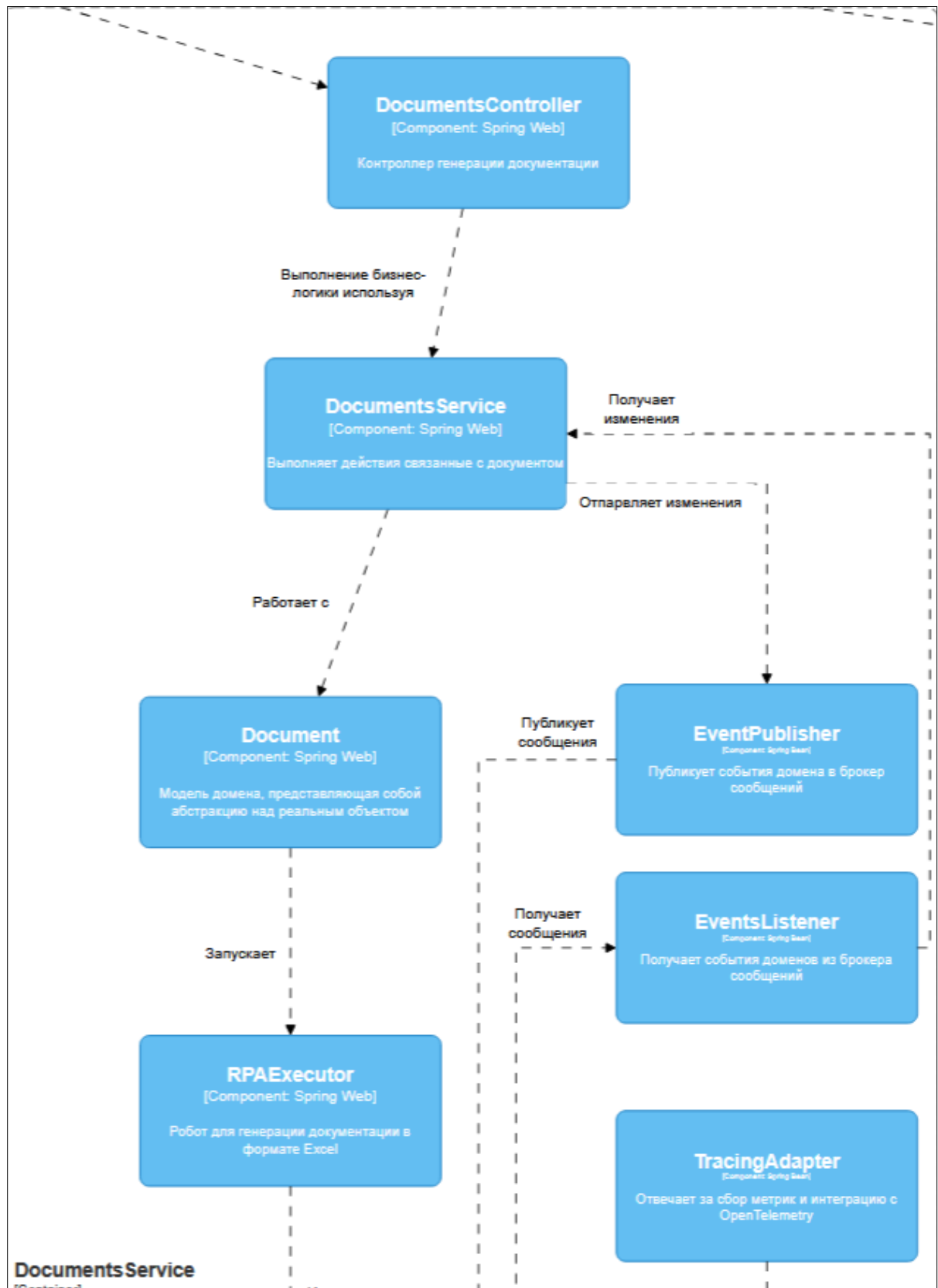


Рисунок 6 – Компонентный уровень архитектуры *DocumentsService*

Далее на рисунке 7 более подробно представлен компонентный уровень архитектуры *WarehouseService* в нотации *C4*.

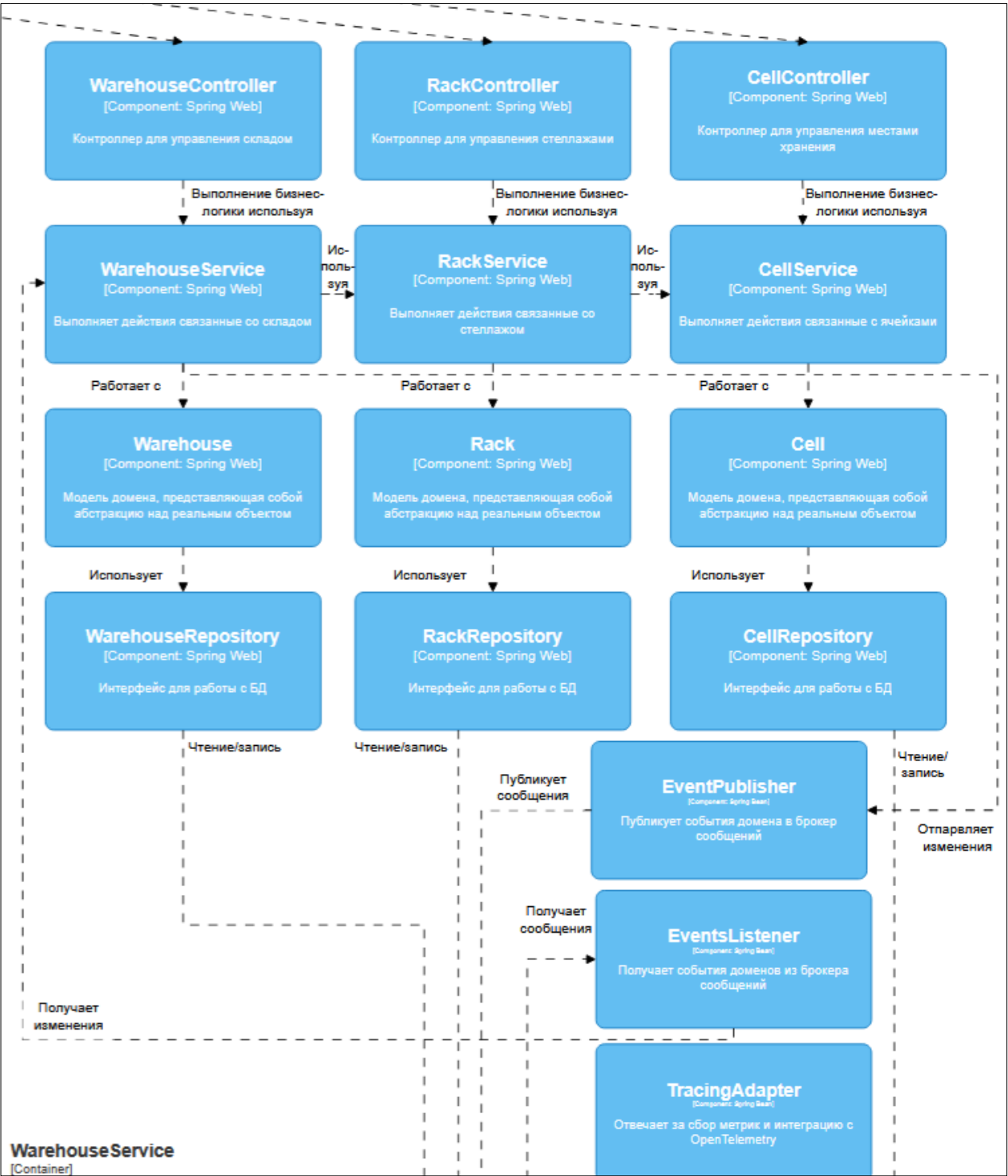


Рисунок 7 – Компонентный уровень архитектуры *WarehouseService*

Таким образом, было выполнено проектирование контейнерного и компонентного уровней архитектуры программного средства в нотации *C4*.

3 СИСТЕМА ДИЗАЙНА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Система дизайна представляет собой набор правил и компонентов, которые формируют единый визуальный язык интерфейсов.

В её основе лежит тщательно продуманная цветовая палитра, включающая как акцентные, так и вспомогательные оттенки. Основные цвета жёлтые и синие. Они используются для привлечения внимания и построения акцентов. Вспомогательные серые оттенки служат для текста, границ и фона, обеспечивая баланс и читаемость. Кроме того, в палитре предусмотрены семантические цвета: красный для ошибок, оранжевый для предупреждений, зелёный для успеха и голубой для информационных сообщений. Это позволяет пользователю на интуитивном уровне различать типы уведомлений и состояния системы.

Иконки в дизайн-системе выступают как универсальный язык символов. Они помогают сократить текстовые элементы, делая интерфейс более лаконичным и интуитивным.

Неотъемлемой частью системы дизайна является типографика. Выделяется строгая иерархия заголовков и текстовых стилей: от крупных заголовков до мелкого текста.

Ключевым элементом системы являются *UI*-компоненты, которые формируют основу интерфейса. Кнопки представлены в нескольких вариантах: стандартные, активные и выделенные, что позволяет гибко выстраивать сценарии взаимодействия. Поля ввода снабжены подписями и значениями, обеспечивая удобство работы с формами. Таблицы позволяют структурировать большие объёмы информации и поддерживают постраничную навигацию, что особенно важно в корпоративных системах. Для уведомлений предусмотрены четыре цветовых состояния, отражающие тип сообщения: ошибка, предупреждение, информационное сообщение и успешное действие. Навигационная панель в верхней части экрана обеспечивает быстрый доступ к разделам, а календарь с возможностью выбора диапазонов дат делает работу с временными интервалами наглядной и удобной.

Система дизайна представлена на рисунке 8.

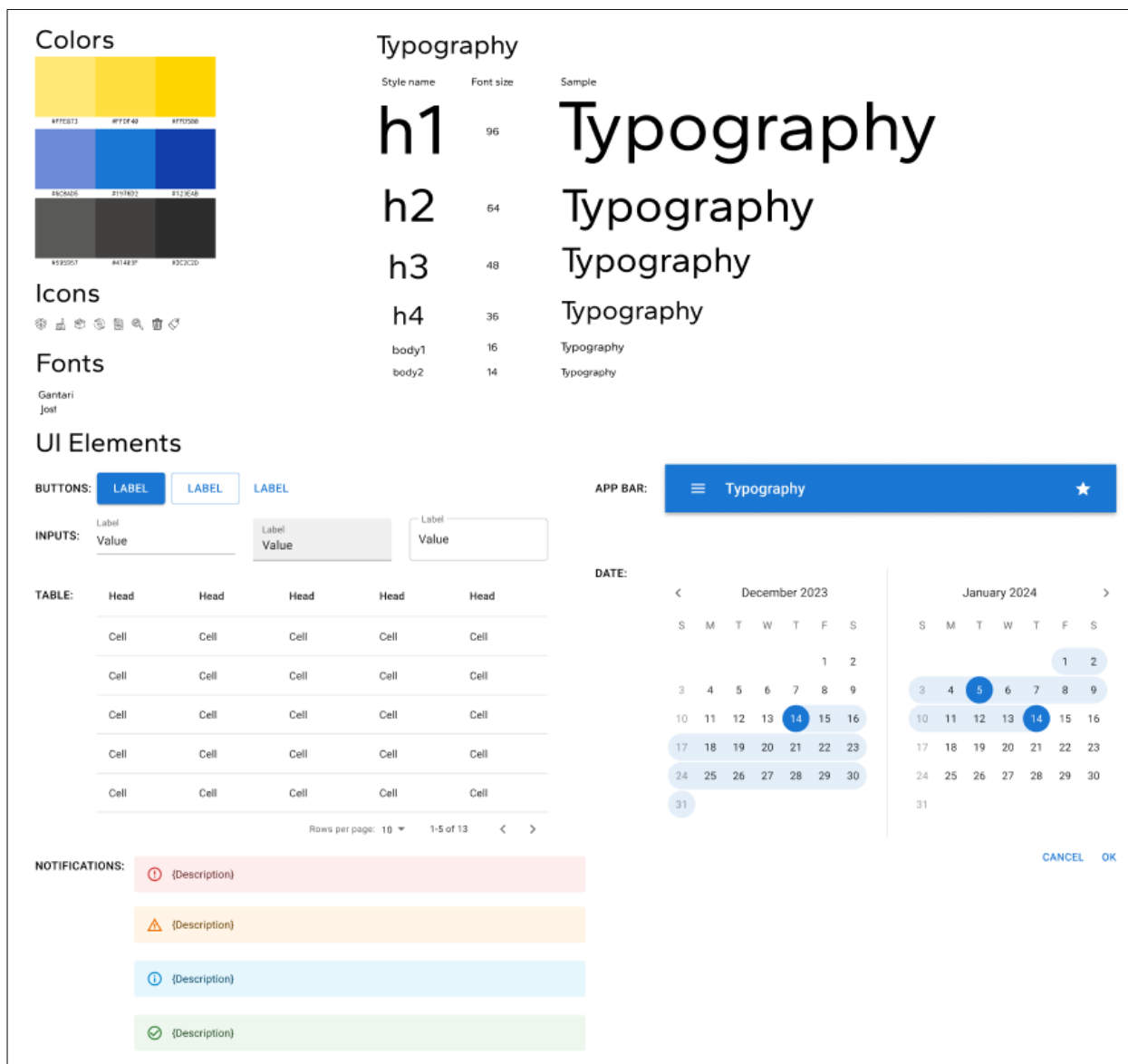


Рисунок 8 – Система дизайна

Ссылка на систему дизайна в Figma: www.figma.com/design/z3Wg78EHGyQn1m5km7OTPn/Design-system?node-id=0-1&p=f&t=GISZRkA9vbl6Q9W7-0

Таким образом, было выполнено проектирование системы дизайна пользовательского интерфейса.

4 ДИАГРАММА КЛАССОВ И ИНТЕРФЕЙСОВ

Архитектура проектируемого программного средства реализована с опорой на принципы *Clean Architecture* и учитывает положения предметно-ориентированного проектирования (*DDD*), а также паттерны *CQRS*, *Saga* и *Database-per-Service*. Основной замысел заключается в разделении системы на изолированные микросервисы, каждый из которых отвечает за свой ограниченный контекст и взаимодействует с другими исключительно посредством обмена событиями.

Внутренняя структура сервисов унифицирована и соответствует слоям *Clean Architecture*. На уровне домена (*Entities*) определяются сущности, обеспечивающие строгую модель предметной области. Так, в сервисе учётных записей ключевым элементом является пользователь, связанный с набором ролей и прав доступа; в сервисе продуктов доменную основу составляют товар и его жизненный цикл, отражённый в статусах; в сервисе документов реализованы сущности акта переоценки, инвентаризационного списка и прочие документы, а также *RPA* для автоматизированного формирования документов; в сервисе организаций центральным элементом выступает организация как субъект; в сервисе склада определены склад, стеллажи и ячейки, формирующие основу складской модели.

Над доменной моделью располагается уровень прикладных сценариев (*Use Cases*), реализующий обработку команд и запросов. Здесь проявляется паттерн *CQRS*: команды изменяют состояние агрегатов и инициируют публикацию доменных событий, тогда как запросы обеспечивают доступ к материализованным представлениям данных. Например, в сервисе документов команда запускает генерацию акта переоценки, а запрос позволяет получить готовый документ; в сервисе склада команды формируют структуру склада, а запросы возвращают сведения о стеллажах и ячейках.

Интеграция с внешним миром и взаимодействие сервисов организованы через уровень адаптеров. Контроллеры обеспечивают доступ к сервисам, репозитории инкапсулируют операции с данными, а специальные инфраструктурные компоненты отвечают за публикацию и приём событий, а также за трассировку и мониторинг. Все реализации инфраструктуры изолированы в отдельном слое, что также соответствует требованию *Clean Architecture*.

Большинство сервисов обладает собственной базой данных в соответствии с принципом *Database-per-Service*. Это исключает межсервисные зависимости на уровне хранилищ и позволяет масштабировать систему фрагментарно, сохраняя согласованность данных посредством обмена событиями. Для координации долгоживущих бизнес-процессов применяется паттерн *Saga*. Например, при оформлении отгрузки инициируются события, которые затрагивают сервис товара, сервис склада, сервис документов и сервис организаций. Последовательность действий управляется сагой, что

обеспечивает как прямое выполнение транзакции, так и компенсацию при возникновении ошибок.

Диаграмма классов и интерфейсов представлена на рисунке 9.

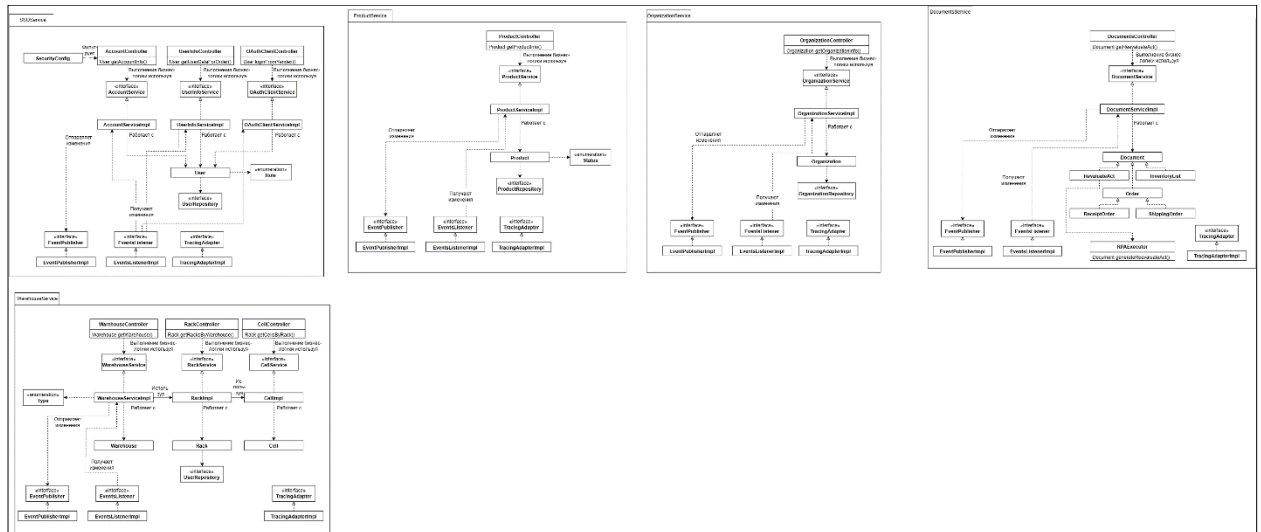


Рисунок 9 – Диаграмма классов и интерфейсов

На рисунке 10 представлена диаграмма классов и интерфейсов компонента *SSOService*.

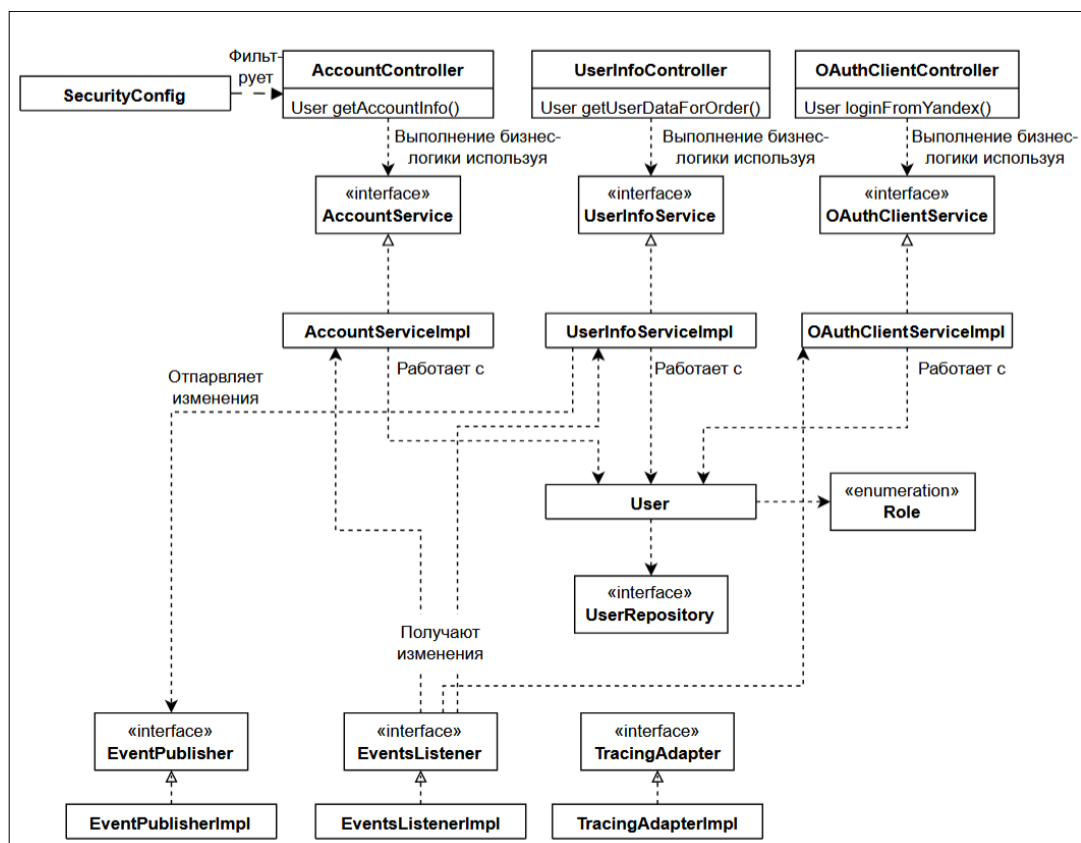


Рисунок 10 – Диаграмма классов и интерфейсов компонента *SSOService*

На рисунке 11 представлена диаграмма классов и интерфейсов компонента *ProductService*.

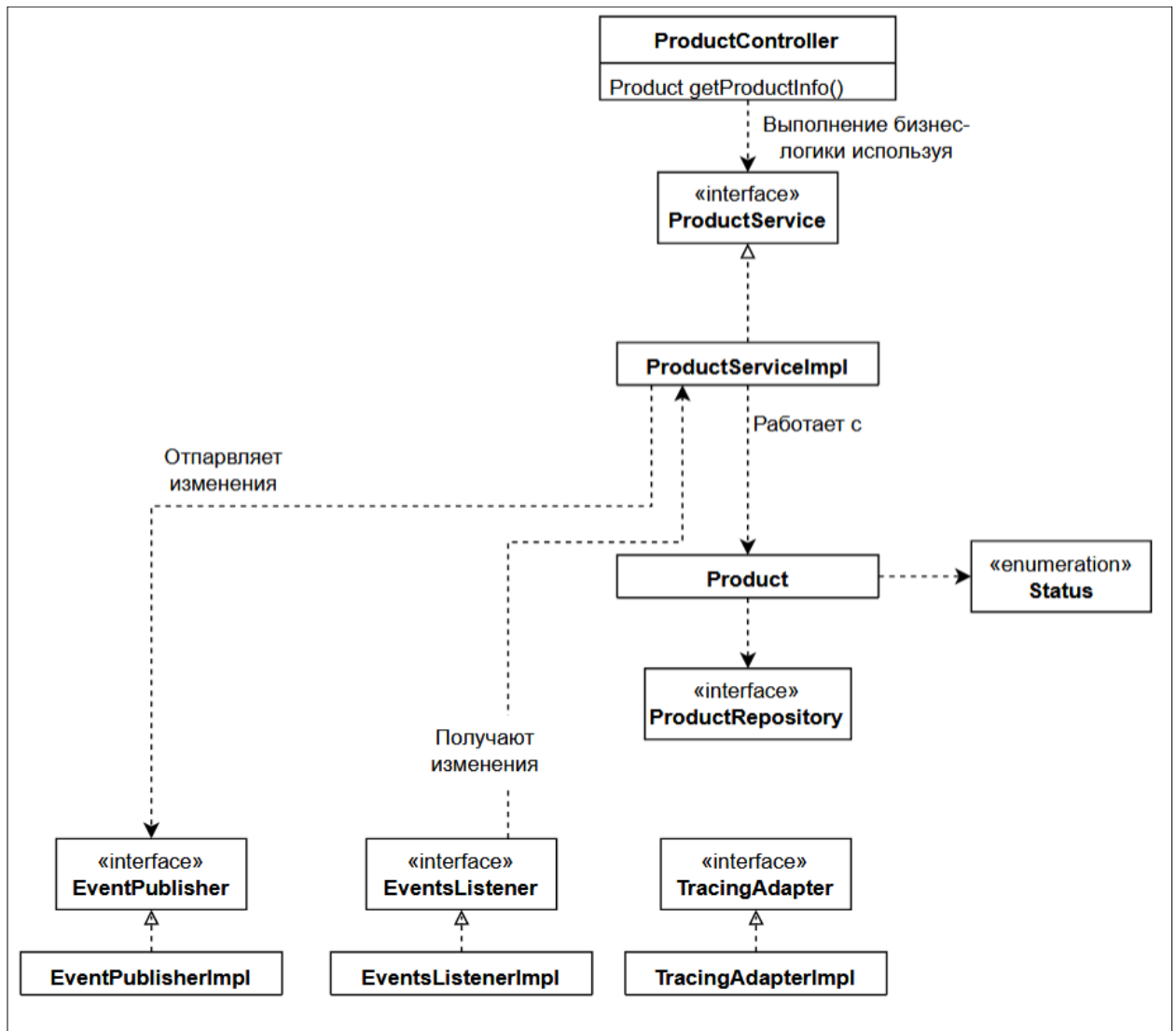


Рисунок 11 – Диаграмма классов и интерфейсов компонента *ProductService*

На рисунке 12 представлена диаграмма классов и интерфейсов компонента *OrganizationService*.

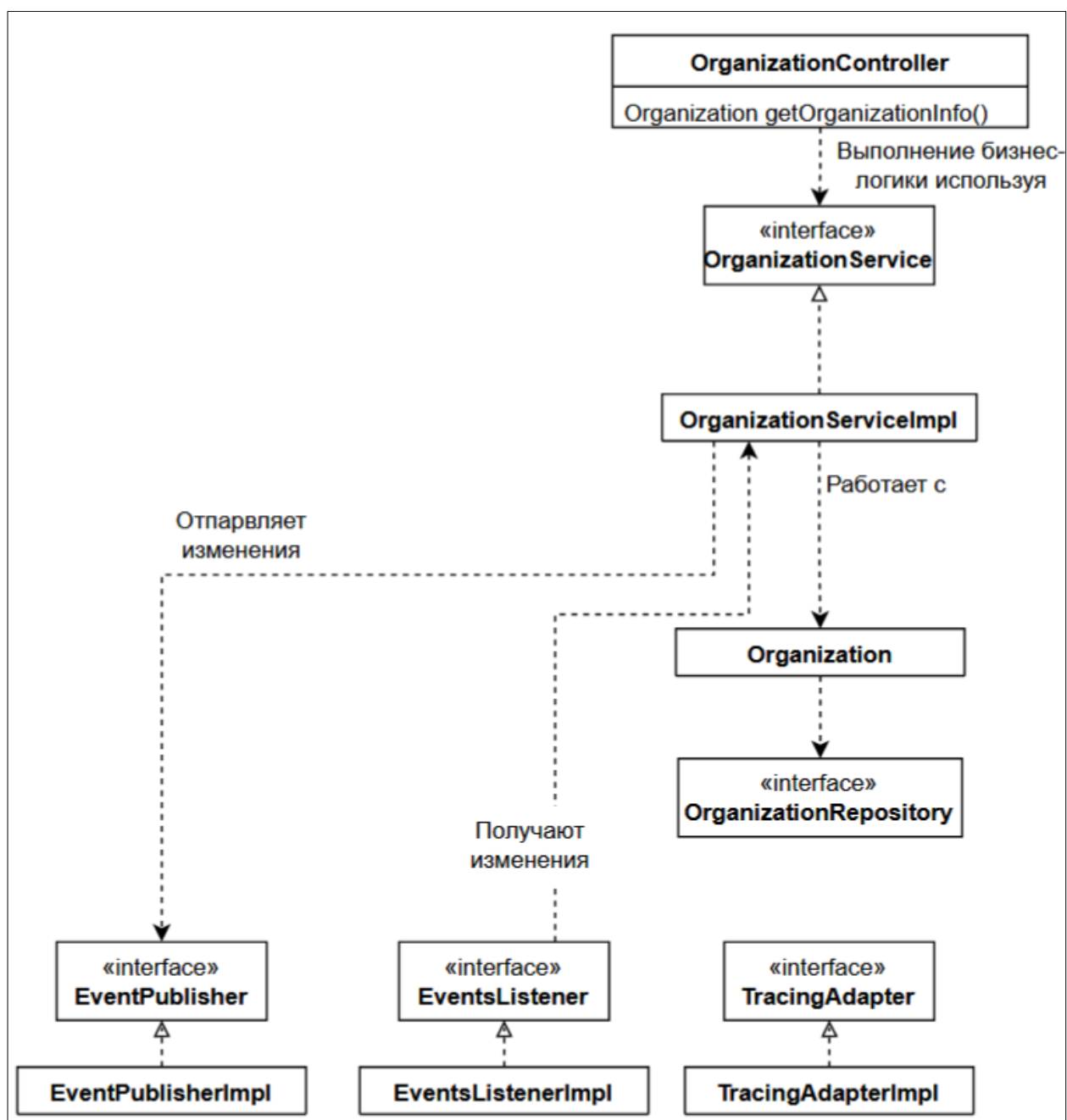


Рисунок 12 – Диаграмма классов и интерфейсов компонента *OrganizationService*

На рисунке 13 представлена диаграмма классов и интерфейсов компонента *DocumentsService*.

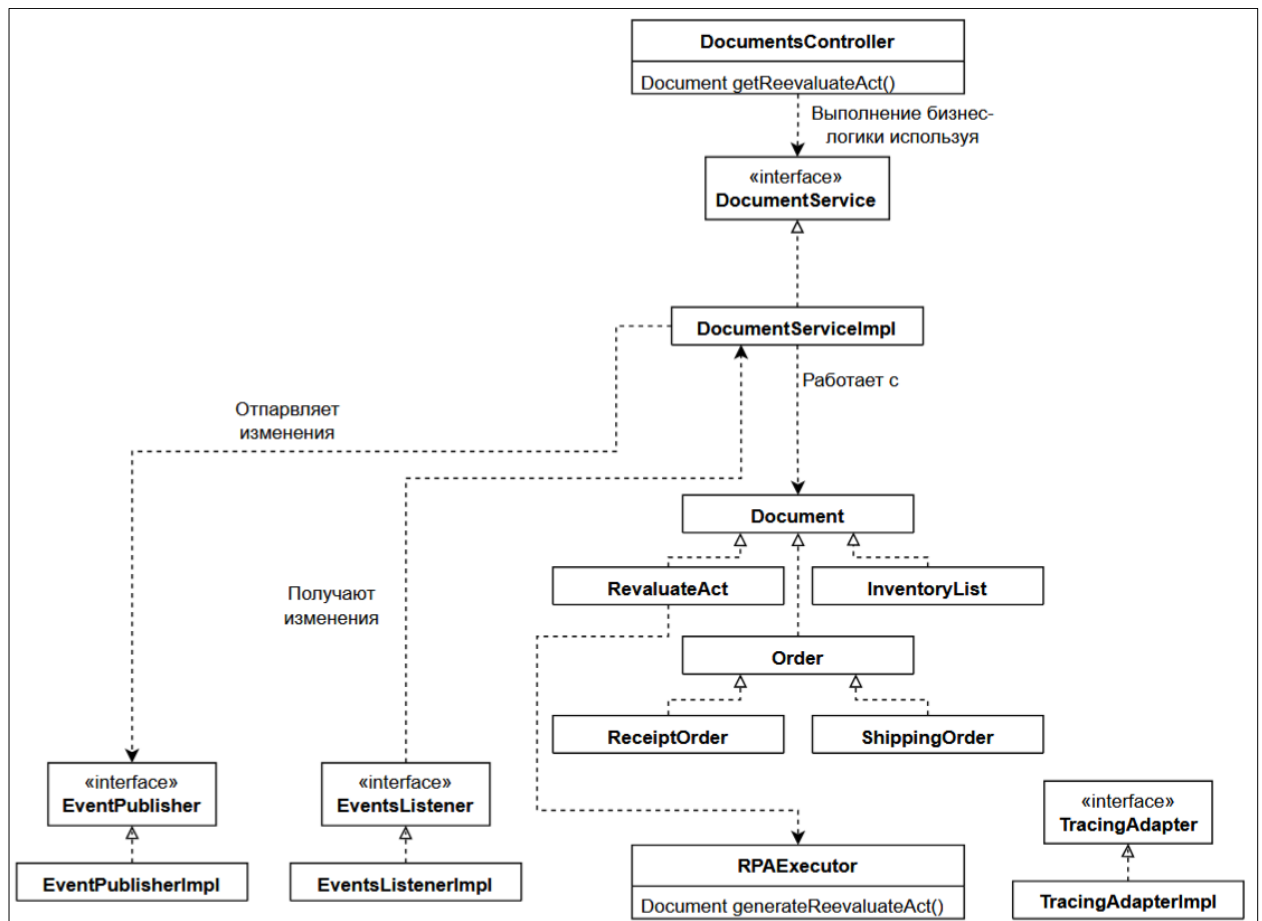


Рисунок 13 – Диаграмма классов и интерфейсов компонента *DocumentsService*

На рисунке 14 представлена диаграмма классов и интерфейсов компонента *WarehouseService*.

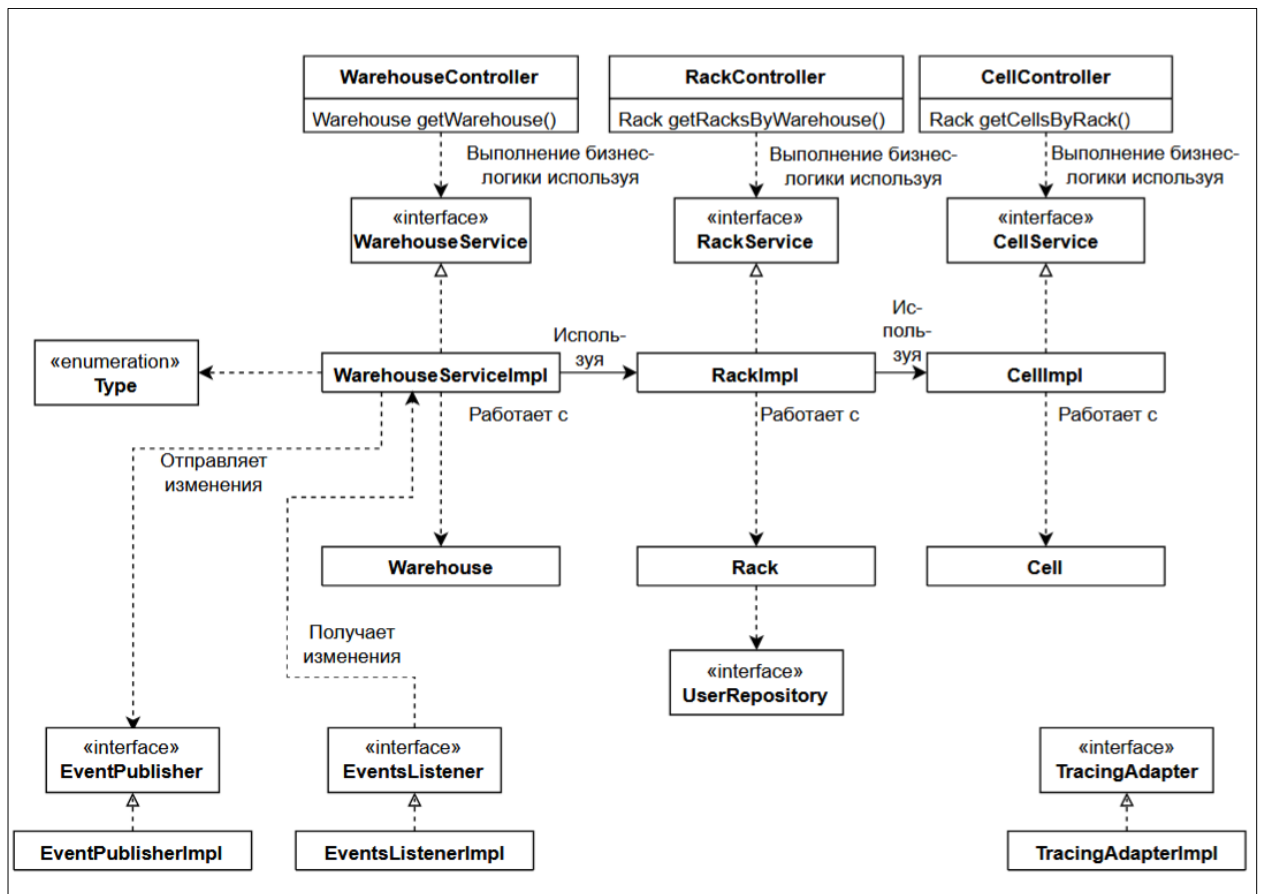


Рисунок 14 – Диаграмма классов и интерфейсов компонента *WarehouseService*

Таким образом, спроектированная архитектура отражает строгие границы контекстов, соответствующие основным направлениям бизнес-деятельности: учёт пользователей, управление товарами, документооборот, работа с организациями и управление складской инфраструктурой. Выбранные архитектурные решения формируют основу для модульной, масштабируемой и отказоустойчивой системы, способной надёжно поддерживать ключевые бизнес-операции в условиях распределённой микросервисной среды.

5 ВЫВОД

В результате выполнения лабораторной работы была спроектирована архитектура программного средства с использованием нотации *C4*, что позволило формализовать представление о структуре и взаимодействии компонентов на различных уровнях детализации. Архитектура описана в соответствии с *Clean Architecture* с учётом подхода *DDD*, что обеспечило разделение ответственности, изоляцию бизнес-логики и возможность дальнейшего масштабирования системы.

В рамках выполнения была создана система дизайна пользовательского интерфейса, включающая перечень элементов управления и их визуальное оформление (типографика, цветовая схема, формы элементов), что формирует единый стиль и повышает удобство взаимодействия пользователя с системой.

Для уточнения архитектурных решений были построены диаграммы классов и интерфейсов, отражающие структуру микросервисов. При этом были учтены принципы *SOLID* и лучшие практики языка *Java*, что позволило повысить гибкость и сопровождаемость проектируемого решения.