Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ УПРАВЛЕНИЯ»



|  |  |
| --- | --- |
|  | Институт информационных систем |
|  | Кафедра информационных систем |

**Отчет по проектной работе**

|  |  |
| --- | --- |
| **Тема проекта** | Разработка приложения для инвентаризации на базе QR-кодов |
|  | |
| *(тема проекта)* | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Направление подготовки** | | | 09.03.03 Прикладная информатика | | | |
| **Образовательная программа** | | | Информационные технологии и системная аналитика | | | |
| **Курс** | **3** | **Группа** | ПИ3-1 |  | **форма обучения** | очная |
|  |  |  |  |  |  |  |
| **Обучающиеся** | | |  |  | Воротников И. А. | |
| *(подпись)* |  | *(Фамилия и Инициалы)* | |
|  | | |  |  | Нестеров Ф. О. | |
| *(подпись)* |  | *(Фамилия и Инициалы)* | |
|  | | |  |  | Прокопенко А. С. | |
| *(подпись)* |  | *(Фамилия и Инициалы)* | |
|  | | |  |  | Фамилия И.О. | |
|  | | | *(подпись)* |  | *(Фамилия и Инициалы)* | |
|  | | |  |  | Фамилия И.О. | |
|  | | | *(подпись)* |  | *(Фамилия и Инициалы)* | |
| **Куратор проекта** | | |  |  | к.э.н., доцент Терехова А. Е. | |
|  | | | *(подпись)* |  | *(ученая степень, звание, Фамилия и Инициалы)* | |
| **Дата сдачи проекта** | | | |  | 2 мая 2025 г. | |

**Москва – 2025**

**Список исполнителей**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | ФИО (полностью) обучающегося | Дата защиты | Оценка | Подпись руководителя |
|  | Воротников Илья Александрович | 02.05.2025 |  |  |
|  | Нестеров Филипп Олегович | 02.05.2025 |  |  |
|  | Прокопенко Андрей Сергеевич | 02.05.2025 |  |  |

Оглавление

[ВВЕДЕНИЕ 4](#_Toc187354388)

[1. СТЕЙКХОЛДЕРЫ И ОБОСНОВАНИЕ АКТУАЛЬНОСТИ ПРОЕКТА 6](#_Toc187354389)

[1. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОЕКТА 9](#_Toc187354390)

[2. ПЛАНИРОВАНИЕ И РАСПРЕДЕЛЕНИЕ ЗАДАЧ ПО ПРОЕКТУ 28](#_Toc187354391)

[ЗАКЛЮЧЕНИЕ 31](#_Toc187354392)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ 32](#_Toc187354393)

# **ВВЕДЕНИЕ**

Проект посвящен разработке веб-приложения для инвентаризации с использованием QR-кодов. QR-код (сокращение от англ. Quick Response, «быстрый отклик») — это двухмерный штрихкод, который используется для кодирования информации в виде черно-белого квадратного изображения. QR-коды могут содержать текст, ссылки на веб-сайты, контактные данные, геолокацию и многое другое. Решение направлено на упрощение и автоматизацию процессов учёта активов и товарных позиций в компаниях различных организационно-правовых форм. Основная цель проекта — создание интуитивно понятного и функционального инструмента, который позволит ускорить и упростить процесс инвентаризации, предоставив пользователям возможность сканировать QR-коды для быстрой идентификации объектов, внесения данных в базу, их актуализации и анализа. Ключевым результатом станет веб-приложение, которое обеспечит надёжный и удобный способ управления процессами инвентаризации.

Предпосылками к созданию проекта стали современные требования бизнеса к повышению оперативности и точности управления активами. Традиционные методы инвентаризации, основанные на ручном вводе данных, подвержены ошибкам и требуют значительных временных затрат. Развитие технологий автоматизации, таких как QR-коды, создаёт возможности для значительного упрощения и оптимизации этих процессов, минимизируя человеческий фактор.

ООО «Интерпроком» — это компания, выступающая в роли заказчика проекта, которая активно внедряет передовые технологии в процессы управления и оптимизации бизнес-процессов. Основным направлением деятельности компании является разработка и реализация информационных систем, способствующих цифровизации предприятий.

Заказчик заинтересован в создании эффективного и удобного инструмента для управления активами, который бы соответствовал современным стандартам и требованиям рынка.

Актуальность проекта обусловлена необходимостью модернизации процессов учёта активов для повышения эффективности работы организаций. Система инвентаризации на базе QR-кодов позволяет сократить трудозатраты, автоматизировать ввод данных, минимизировать риски потерь и ошибок, а также ускорить процессы управления имуществом. Внедрение подобных решений особенно полезно для компаний любого масштаба и сферы деятельности, которые стремятся к цифровизации и улучшению своих бизнес-процессов.

Новизна проекта заключается в интеграции современных технологий автоматизированного учёта с интуитивным веб-интерфейсом, доступным на любых устройствах с интернет-браузером. Ключевые функциональные новшества включают масштабируемость, что позволяет адаптировать систему для предприятий различных размеров, и использование динамических QR-кодов, содержащих изменяемую информацию. Это даёт возможность обновлять данные в режиме реального времени без необходимости генерации новых кодов.

Проект имеет прикладной характер и тесно связан с направлением подготовки специалистов в области разработки программного обеспечения и информационных систем. Он ориентирован на решение актуальных задач цифровизации бизнеса, создавая новое качество инвентаризации, что делает его востребованным для широкого круга организаций.

# **СТЕЙКХОЛДЕРЫ И ОБОСНОВАНИЕ АКТУАЛЬНОСТИ ПРОЕКТА**

1.1Стейкхолдеры:  
Проект «Разработка веб-приложения для инвентаризации на базе QR-кодов» решает важные задачи для компаний-заказчиков и конечных пользователей, сталкивающихся с необходимостью модернизации учётных процессов. Ключевой заинтересованной стороной является ООО «Интерпроком», занимающееся разработкой прикладного программного обеспечения, системной интеграцией и дистрибуцией решений для автоматизации бизнес-процессов. Для компании критически важно внедрение надёжного и современного инструмента, который позволит автоматизировать инвентаризацию активов, ускорить сбор и обработку данных, сократить трудозатраты и минимизировать ошибки, возникающие из-за человеческого фактора. Использование QR-кодов в данном проекте обеспечивает лёгкость идентификации объектов и быструю актуализацию информации в базе данных, что является важным шагом к достижению высокой эффективности работы.

1.2 Потребительские сегменты:

Включают широкий круг организаций, заинтересованных в улучшении процессов инвентаризации. Среди них магазины, оптовые компании, производственные предприятия, логистические и складские компании, предприятия электронной коммерции, медицинские учреждения, сетевые рестораны, строительные компании и автосервисы. Эти организации часто сталкиваются с проблемами, связанными с устаревшими методами учёта, такими как ручной ввод данных, который приводит к ошибкам, потере времени и увеличению затрат. Кроме того, отсутствие автоматизации затрудняет доступ к актуальной информации, делает инвентаризацию трудоёмкой и снижает точность учётных операций.

1.3 Актуальность:

Проектное решение предусматривает внедрение веб-приложения, которое решает указанные проблемы благодаря автоматизации рутинных операций, созданию единой базы данных для хранения информации об объектах инвентаризации и использованию QR-кодов для их идентификации. Это значительно упрощает процесс учёта, минимизирует влияние человеческого фактора, исключает ошибки ввода данных и ускоряет выполнение инвентаризационных мероприятий. Встроенные аналитические инструменты помогают компаниям прогнозировать потребности и управлять запасами более эффективно.

1.4 Новизна предлагаемого решения:

Заключается в использовании динамических QR-кодов, которые позволяют обновлять данные об объектах в режиме реального времени, что особенно важно для крупных компаний с большими объёмами инвентаризации. Масштабируемость приложения делает его подходящим как для малых предприятий, так и для крупных корпораций. Интуитивно понятный интерфейс обеспечивает быструю адаптацию сотрудников и повышает удобство использования.

1.5 Коммуникация со стейкхолдером

Взаимодействие с ООО «Интерпроком» осуществляется посредством регулярных встреч, которые позволяют уточнять требования, согласовывать изменения и демонстрировать промежуточные результаты работы. Это способствует эффективной реализации проекта и удовлетворению потребностей всех заинтересованных сторон. Реализация данного веб-приложения станет важным шагом к повышению точности и прозрачности учётных операций, что делает проект актуальным и востребованным на рынке автоматизации бизнес-процессов.

1.6 Преимущества использования QR-кодов:

Веб-приложение на базе QR-кодов обладает рядом ключевых преимуществ, которые выгодно отличают его от аналогичных решений. Приложение обеспечивает гибкость за счёт масштабируемости, позволяя настроить систему под потребности компаний различных размеров и отраслей. Использование динамических QR-кодов добавляет функциональность, недоступную в большинстве традиционных систем: обновление данных в реальном времени без необходимости перепечатывать или заменять коды. Кроме того, приложение интегрируется с существующими системами учёта, что облегчает процесс внедрения и минимизирует затраты на обучение сотрудников.

1.7 Риски:

Несмотря на очевидные преимущества, проект может столкнуться с рядом рисков. Во-первых, технические сложности при интеграции с уже существующими системами учёта. Для минимизации этого риска предусмотрено использование стандартизированных API и модулей интеграции. Во-вторых, риск низкой адаптации конечных пользователей к новому инструменту. Это может быть решено за счёт создания интуитивно понятного интерфейса и проведения обучающих мероприятий. Также могут возникнуть риски, связанные с обеспечением безопасности данных, которые будут минимизированы с использованием современных технологий шифрования и строгих политик доступа.

# **ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОЕКТА**

Разработка проекта велась поэтапно, что позволило структурировать процесс и эффективно распределить задачи. Каждый этап был направлен на достижение конкретных результатов, необходимых для создания функциональной системы.

Выбор технологий для разработки проекта

Figma была выбрана для разработки дизайн-прототипа нашего приложения для инвентаризации благодаря её уникальным преимуществам по сравнению с другими платформами. Во-первых, Figma предоставляет облачную среду, которая позволяет работать над проектом в реальном времени, обеспечивая совместную работу команды независимо от географического положения участников. Это особенно актуально для нашей команды, работающей удалённо. Во-вторых, платформа отличается простотой в использовании и интуитивно понятным интерфейсом, что значительно ускоряет процесс разработки и согласования дизайна. Кроме того, Figma поддерживает интеграцию с другими инструментами и предоставляет широкий набор функций для создания интерактивных прототипов, что облегчает тестирование пользовательского интерфейса на ранних этапах проекта. Эти преимущества сделали Figma оптимальным выбором для реализации нашего проекта, обеспечив эффективность и удобство в процессе проектирования.

Flutter/Dart для разработки приложения:

Кроссплатформенность: Flutter позволяет разрабатывать приложения одновременно для Android и iOS, что значительно снижает трудозатраты и упрощает поддержку.

Высокая производительность: благодаря использованию языка Dart и архитектуре Flutter приложения компилируются в нативный код, обеспечивая высокую скорость работы и плавность интерфейса.

Широкие возможности интерфейса: Flutter предоставляет гибкие виджеты, которые позволяют создавать современные, удобные и интуитивно понятные пользовательские интерфейсы, соответствующие требованиям нашего проекта.

Открытый исходный код: Flutter и Dart также являются бесплатными инструментами, что снижает затраты на разработку.

Экосистема и сообщество: Широкая экосистема пакетов и активное сообщество разработчиков упрощают внедрение функционала, например, сканирования QR-кодов.

Почему связка PostgreSQL + FastAPI не подошла

Причина Объяснение

1. Сложность инфраструктуры PostgreSQL и FastAPI требуют настройки и поддержки сервера (или VPS), а также конфигурации HTTPS, SSL, безопасности и резервного копирования. Это замедляет разработку, особенно на старте.

2. Отдельное API и БД Мы должны были разрабатывать отдельный API (FastAPI) и обеспечивать его доступность — что добавляет слой сложности и ошибок.

3. Отсутствие "из коробки" авторизации и ролей. Для авторизации/аутентификации пришлось бы либо интегрировать OAuth/JWT вручную, либо использовать сторонние решения (например, Auth0). Firebase решает это встроенными средствами.

4. Слабая интеграция с Flutter FastAPI и PostgreSQL не предоставляют готовых решений для Flutter — всё взаимодействие приходится писать с нуля через HTTP-клиенты (например, Dio), в том числе обработку ошибок, форматов и т.п.

5. Сложная настройка масштабируемости и офлайн-режима PostgreSQL сам по себе не поддерживает офлайн-доступ. Нужно было бы вручную писать кэш или синхронизацию.

Почему мы выбрали Firebase

1. Отличная интеграция с Flutter cуществует официальный пакет cloud\_firestore и firebase\_auth, которые дают нативную, простую и стабильную интеграцию. Поддержка от Google.

2. Backend как сервис (BaaS) Firebase предоставляет базу данных, хранилище, авторизацию, хостинг и аналитику — всё в одном, без необходимости разворачивать сервер.

3. Поддержка реального времени Firestore обновляет данные на всех устройствах автоматически без опроса сервера. Это идеально для синхронизации между несколькими пользователями.

4. Простота хранения структурированных данных Firestore позволяет хранить вложенные документы и коллекции — это удобно для структуры "локация → предметы".

5. Встроенная авторизация Firebase поддерживает email/password, Google, а также анонимный вход — можно быстро добавить проверку прав доступа.

6. Безопасность на уровне правил Мы можем ограничивать доступ к данным с помощью Firebase Security Rules без необходимости писать сложную бизнес-логику на сервере.

7. Бесплатный тариф на этапе MVP Firebase предлагает бесплатную квоту, которой достаточно для разработки и демонстрации, что уменьшает издержки.

Использование NoSQL подхода в даёт гибкость в хранении данных, которую сложно получить в традиционных реляционных базах данных.

Конкретно в нашем проекте это дало нам

Быстрый старт: мы за день подняли хранилище, подключили его к Flutter и начали сохранять объекты.

Упрощённую архитектуру: без FastAPI и серверного кода мы работаем только с фронтендом, что ускоряет итерации.

Меньше багов и меньше точек отказа.

Гибкость в представлении данных: легко группируем по "этаж/комната", фильтруем и храним доп. атрибуты (описание, QR и т.д.).

Удобное масштабирование: при необходимости можно добавить авторизацию, аналитику или облачные функции без смены платформы..

Стадии выполнения проекта:

1.Моделирование процессов учета товаров в организации в целом и инвентаризации в частности в нотации IDEF0 для лучшего понимания разработки приложения.

2. Разработка Use Case: создание сценариев использования для определения основных функций и взаимодействий в системе.

3. Создание дизайн-макетов в Figma: разработка визуальной концепции интерфейса для обеспечения удобства работы пользователей.

4. Прототипирование интерфейса в Android Studio: реализация экранов приложения и базовой навигации между ними.

5. Построение ER-диаграммы и дальнейшая разработка прототипа базы данных: проектирование структуры базы данных для хранения и обработки информации.

6. Доработка интерфейса приложения, разработка backend-а, интеграция Firebase инструментов.

Такой подход позволил последовательно выполнить все необходимые этапы, от определения требований до создания прототипа, закладывая прочную основу для реализации проекта.

**Стадия 1.** Моделирование процессов учета товаров

Для лучшего понимания задач разработки приложения проведено моделирование процессов учёта товаров в организации в целом и инвентаризации в частности с использованием методологии IDEF0. Это позволило детализировать основные этапы, связанные с инвентаризацией, и выявить ключевые взаимосвязи между элементами системы.

Основные диаграммы:

Общий процесс проведения инвентаризации (Рис. 1):

Данная диаграмма описывает весь цикл инвентаризации, начиная с подготовки к учёту и заканчивая анализом данных.

Отображены входные данные, такие как текущая база активов и методы идентификации (включая QR-коды).

Основные этапы включают сбор информации о товарах, внесение данных в систему, проверку актуальности и корректности записей, а также генерацию итогового отчёта.

Этап с присвоением и наклейкой QR-кодов (Рис. 2):

Более детализированная диаграмма, фокусирующаяся на процессе идентификации активов с помощью QR-кодов.

Включает этапы: генерация QR-кодов для каждого объекта, их печать и наклейка на соответствующие активы.

Особое внимание уделено корректному внесению данных в систему для привязки QR-кода к определённому активу, что является основой для последующего быстрого доступа к информации.

Отображены входные данные (уникальные идентификаторы активов), выходные данные (готовые QR-коды) и механизмы (оборудование для печати, интерфейсы системы).

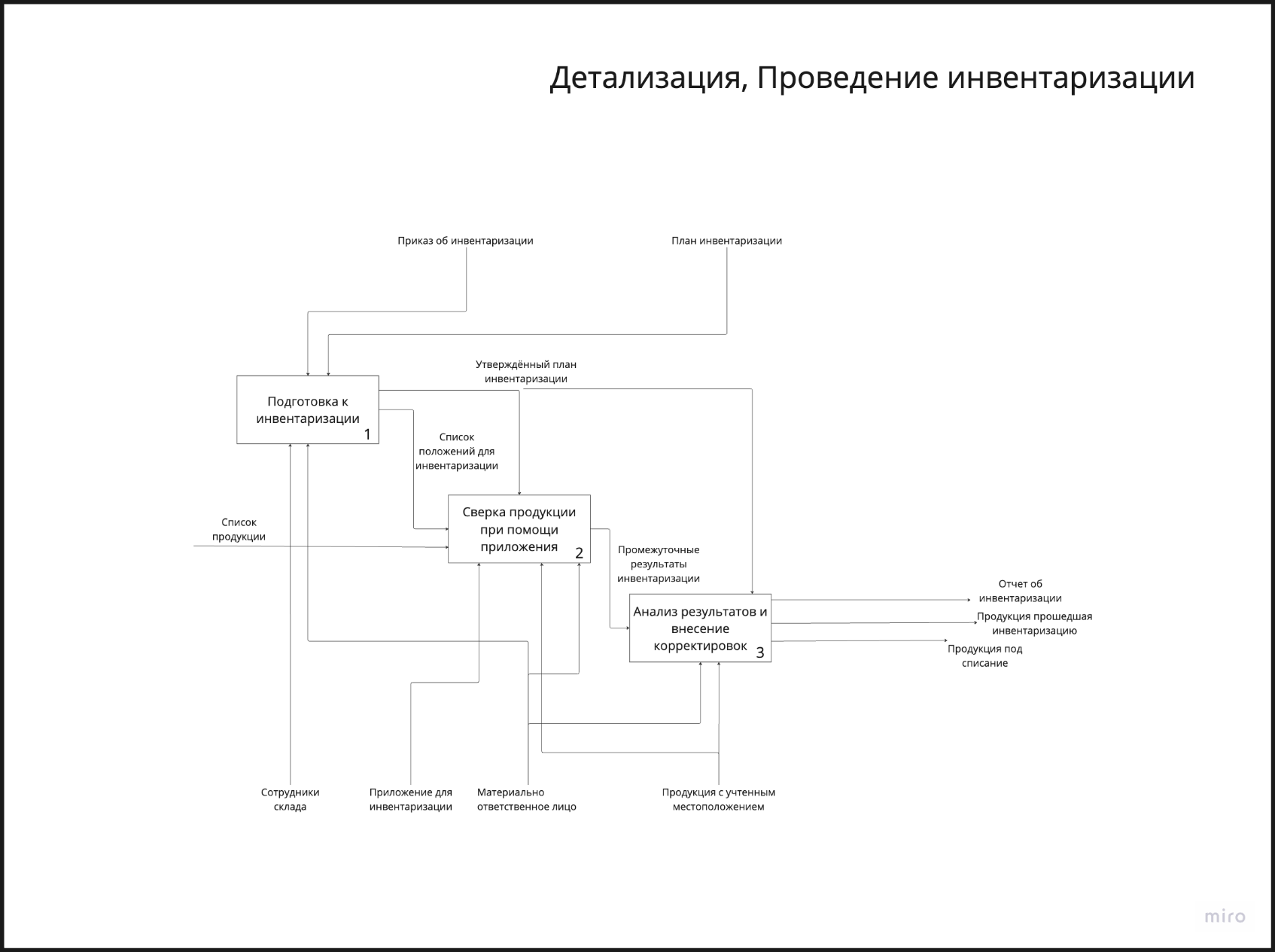


Рис. 1 – Проведение инвентаризации в нотации IDEF0

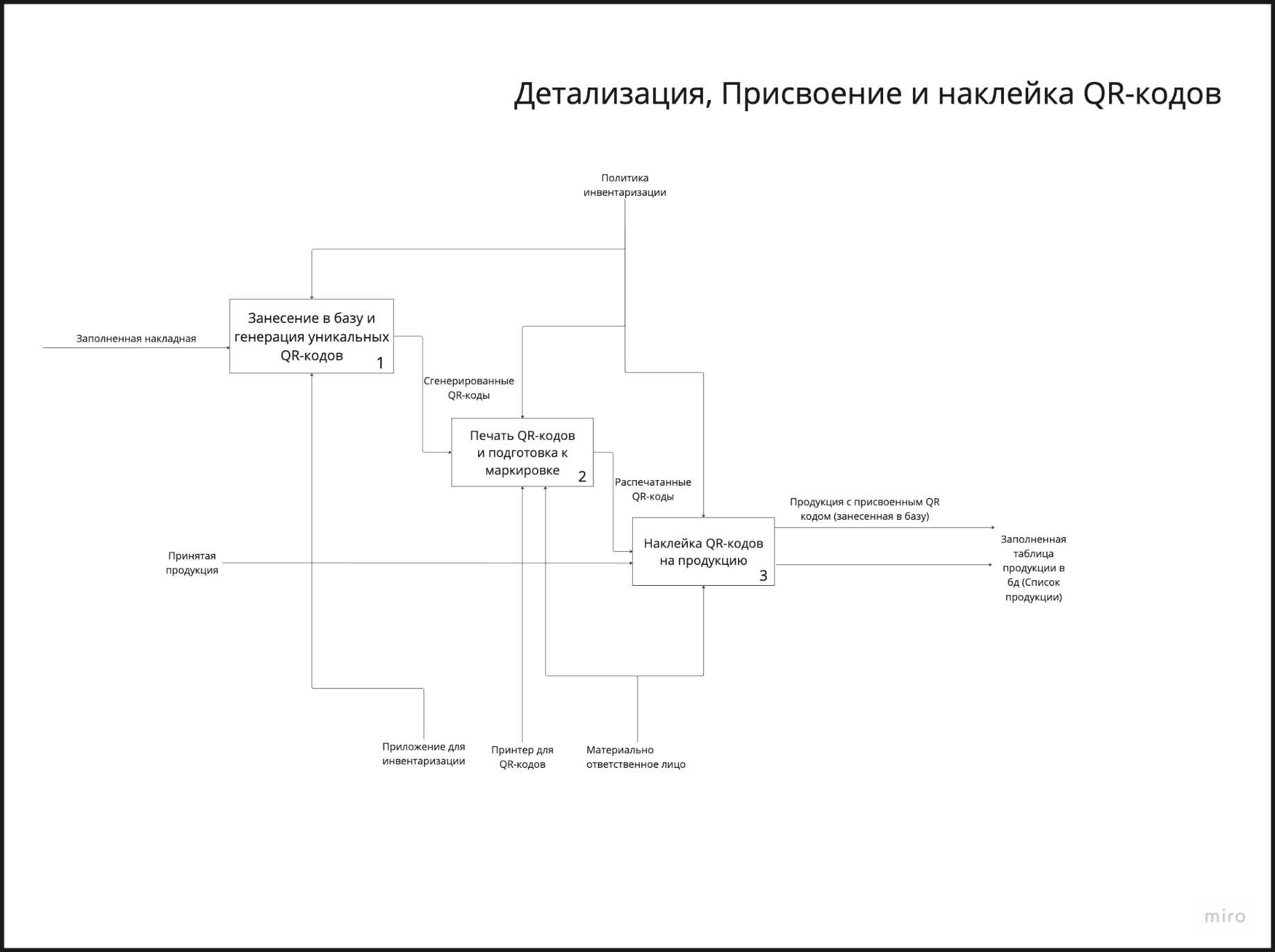


Рис. 2 – Присвоение и наклейка QR-кодов в нотации IDEF0

**Стадия 2.** Use Case (Рис.3) были выполнены следующие задачи:

1. Анализ и идентификация акторов системы:

Были определены ключевые пользователи системы, такие как обычные пользователи (для инвентаризации) и администраторы (для управления доступом и данными).

2. Определение основных функций системы:

Выделены основные сценарии использования: управление активами, генерация и использование QR-кодов, фильтрация и поиск данных, управление правами доступа, импорт и экспорт данных.

3. Описание сценариев взаимодействия:

- Для каждого сценария был составлен основной поток событий, включающий шаги пользователя и реакции системы.

- Были разработаны альтернативные потоки для обработки ошибок или отклонений от стандартного поведения.

4. Документирование Use Case:   
Описания сценариев были структурированы в удобной форме, что позволило использовать их для дальнейших этапов проектирования, таких как создание макетов интерфейса и архитектуры системы.

Рис. 3 – Use Case

Стадия 3. Проектирование пользовательского интерфейса

Спроектированы дизайн-макеты приложения в Figma. Работа велась последовательно и охватывала следующие шаги:

На основе ранее разработанных сценариев использования выделены ключевые экраны и элементы интерфейса, которые необходимы для реализации функций системы.

Разработаны основные экраны (Рис. 4, 5) такие как: Главная страница с навигацией по приложению, Страницы управления активами и положениями, Интерфейс для генерации QR-кодов, Экран для просмотра и фильтрации данных, Страница настройки прав доступа, Создана общая структура интерфейса, включая меню, формы ввода и элементы навигации.

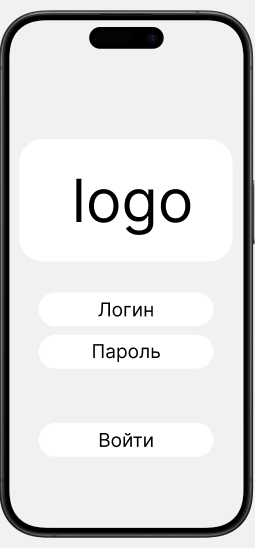
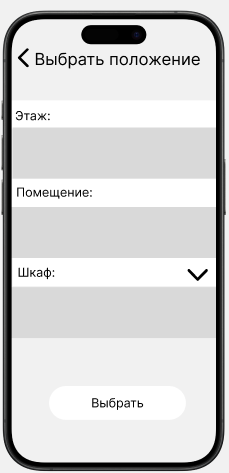
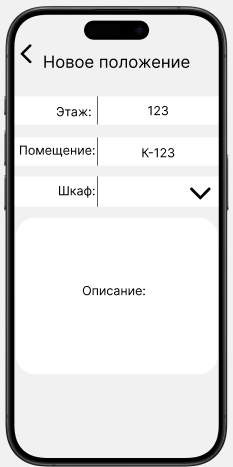
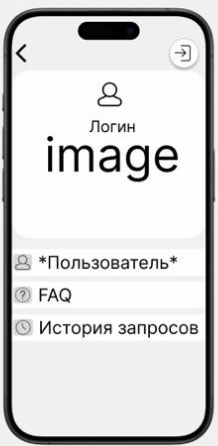
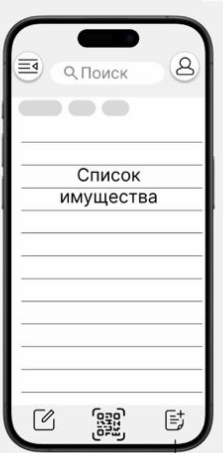


Рис. 4 – Макеты экранов в Figma (экран авторизации, главный экран, экран профиля пользователя, экран инвентаризации, экран добавления нового положения, экран выбора положения)

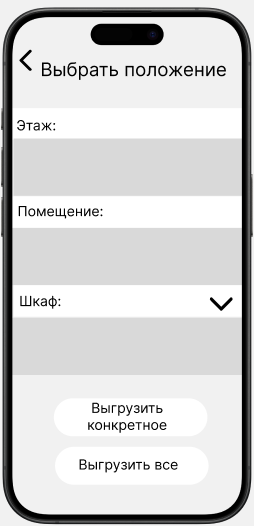
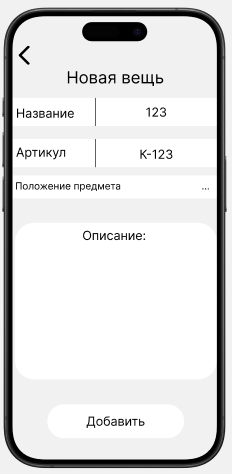


Рис. 5 – Макеты экранов в Figma (экрана добавления новой вещи, экран локального сканирования, экран выбора положения)

Тестирование макетов:

Проведена проверка макетов на соответствие пользовательским требованиям и сценарию работы. Внесены улучшения на основе обратной связи для повышения интуитивности интерфейса.

**4 стадия.** Прототипирование интерфейса в Android Studio.

Проект представляет собой мобильное приложение, реализованное с использованием Flutter, предназначенное для инвентаризации на основе QR-кодов. Структура проекта делится на несколько ключевых компонентов и папок, каждый из которых выполняет свою задачу.

#### **Assets:**

Содержит статические ресурсы, такие как изображения и иконки, используемые в приложении.

**1.1. icons:**

**1.1.1. profile**: Иконки и изображения, связанные с профилем пользователя.

**1.1.2. home\_page**: Иконки и изображения, используемые на главной странице приложения.

1. build:

Сгенерированные файлы при сборке проекта, которые не требуют изменений пользователем.

#### **Fonts:**

Шрифты, используемые в приложении. Сюда можно добавлять собственные шрифты, которые будут использоваться в UI.

#### **ios:**

Папка, содержащая файлы для сборки проекта под iOS, включая настройки и конфигурации платформы.

#### **Lib:**

Основная папка с кодом приложения.

**5.1. features:**

**5.1.1. view:**

**5.1.1.1. choose.dart**: Экран выбора положения (например, для выбора этажа, офиса или шкафа для инвентаризации).

**5.1.1.2. home\_screen.dart**: Главная страница приложения, где отображается информация о текущем инвентаре и есть доступ к основным функциям.

**5.1.1.3. login\_screen.dart**: Экран авторизации, где пользователь вводит логин и пароль для входа в приложение.

**5.1.1.4. new\_thing.dart**: Экран для добавления новой вещи в базу данных.

**5.1.1.5. uploading\_data.dart**: Экран, отображающий процесс выгрузки данных, например, экспорт инвентаризации.

**5.1.1.6. view.dart**: Экспорт всех экранов из папки **view**, чтобы их можно было использовать в других частях приложения.

**5.2. Widgets:**

**5.2.1. list\_property.dart**: Виджет для главного экрана, который отображает список имущества и положений, а также дополнительные кнопки для взаимодействия с данными.

**5.2.2. new\_pos.dart**: Экран для добавления нового положения (например, добавление нового этажа, офиса или шкафа для инвентаризации).

**5.2.3. profile.dart**: Экран профиля пользователя, который не поддается редактированию, только для просмотра информации.

**5.2.4. Widgets.dart**: Экспорт всех виджетов из папки **Widgets**, чтобы их можно было использовать в других частях приложения.

* 1. **repositories:** Репозитории, отвечающие за работу с данными. Содержат логику для получения, сохранения и обновления данных. Например, репозитории для работы с сервером или локальной базой данных.
  2. **router:**

**5.4.1. routes.dart**: Содержит маршруты (роуты) приложения, связывая экраны с их адресами для навигации между экранами.

* 1. **theme:**

**5.5.1. theme.dart**: Определяет основную тему приложения, включая цвета, шрифты и другие параметры, которые обеспечивают единый стиль интерфейса.

1. **main.dart:**

Главный файл, в котором инициализируется приложение, настраиваются маршруты, темы и другие глобальные параметры.

1. **QR\_Code\_App.dart:**

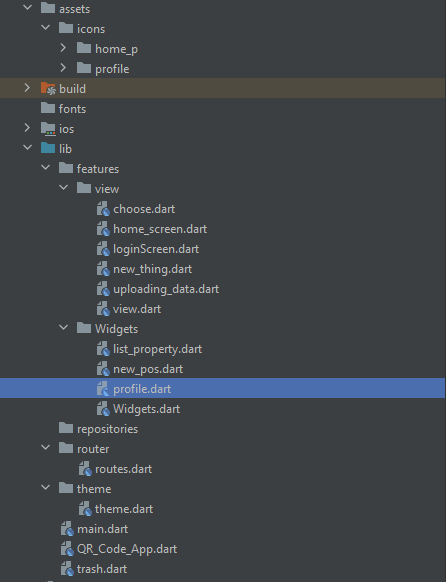
Главный файл приложения, который запускает и инициализирует все компоненты.

Рис. 6 – директории проекта в Android Studio(структура приложения)

В дальнейшем структура проекта будет расширяться.

HomeScreen(Главный экран):

import 'package:flutter/material.dart';

import 'package:flutter\_svg/flutter\_svg.dart';

import '/features/Widgets/Widgets.dart';

import '/router/routes.dart';

class HomePage extends StatefulWidget {

const HomePage({super.key});

@override

State<HomePage> createState() => \_HomePageState(); }

class \_HomePageState extends State<HomePage> {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

elevation: 0,

// Убираем автоматическую кнопку для бокового меню

automaticallyImplyLeading: false,

title: Row(

mainAxisAlignment: MainAxisAlignment.spaceBetween,

crossAxisAlignment: CrossAxisAlignment.center,

children: [

// Кнопка для открытия бокового меню

Builder(

builder: (context) => IconButton(

icon: SvgPicture.asset('assets/icons/home\_p/sidebar.svg'),

iconSize: 60,

onPressed: () {

// Открытие бокового меню

Scaffold.of(context).openDrawer(); }, ), ),

SizedBox(width: 13),

// Поле поиска

Container(

height: 40,

width: 190,

child: TextField(

decoration: InputDecoration(

hintStyle: TextStyle(fontSize: 20),

hintText: 'Поиск',

prefixIcon: SvgPicture.asset(

'assets/icons/home\_p/search.svg',

height: 15,

width: 15, ),

border: OutlineInputBorder(

borderRadius: BorderRadius.circular(35.0),

borderSide: BorderSide.none, ),

filled: true,

fillColor: Colors.white,

contentPadding: EdgeInsets.symmetric(vertical: 10, horizontal: 15), ), ), ),

SizedBox(width: 13),

// Кнопка профиля

IconButton(

icon: SvgPicture.asset('assets/icons/home\_p/profile.svg'),

iconSize: 60,

onPressed: () {

Navigator.pushNamed(context, "/profile"); }, ), ], ), ),

// Боковое меню

drawer: Drawer(

child: ListView(

padding: EdgeInsets.zero,

children: [

// Уменьшаем высоту синего поля с надписью "Меню"

// SizedBox(height: 40,),

DrawerHeader(

padding: EdgeInsets.all(20), // Уменьшаем внутренние отступы

child: Text(''),

decoration: BoxDecoration(

color: Colors.blue, ), ),

Container(

decoration: BoxDecoration(

color: Colors.blue[50], // Фон для элемента

borderRadius: BorderRadius.circular(12), // Закругление углов ),

child: ListTile(

title: Text('Инвентаризация',style: TextStyle(fontSize: 20,color: Colors.black)),

onTap: () {

Navigator.pushNamed(context, "/inventory"); }, ), ),

SizedBox(height: 12,),

Container(

decoration: BoxDecoration(

color: Colors.blue[50], // Фон для элемента

borderRadius: BorderRadius.circular(12), // Закругление углов ),

child: ListTile(

title: Text('Выгрузка данных',style: TextStyle(fontSize: 20,color: Colors.black)),

onTap: () {

Navigator.pushNamed(context, "/uploading\_thing"); }, ), ), ], ), ),

body: InventoryScreen(), ); } }

Суммарно на данный момент приложение включает в себя свыше 1 150 строк кода.

**5 стадия.** Построение ER-диаграммы и дальнейшая разработка прототипа базы данных

На этапе моделирования базы данных была спроектирована структура, обеспечивающая эффективное хранение и обработку информации для системы инвентаризации. Работа включала в себя следующие шаги:

Анализ требований к данным:

Изучены потребности системы в хранении информации об активах, их местоположении, связанных QR-кодах, ролях пользователей и истории изменений.

Определены ключевые сущности, их атрибуты и взаимосвязи, которые необходимы для корректной работы системы.

Основные сущности и их связь с процессом инвентаризации:

items (объекты):

Таблица хранит информацию о предметах (активах) с их уникальными идентификаторами, названиями и описанием. Это ключевая сущность, так как инвентаризация ориентирована на учёт активов.

storage\_locations (места хранения):

Таблица связывает объекты с местами их хранения. Поля включают:

item\_id: идентификатор объекта.

location\_type: тип места хранения.

location\_id, floor\_id, room\_id: ссылки на конкретные местоположения.

qr\_code: уникальный QR-код для каждого места хранения или объекта.

floors (этажи):

Содержит номера этажей для систематизации местоположений в зданиях.

rooms (комнаты):

Таблица для учёта комнат, связанных с этажами через floor\_id.

closet (шкафы):

Таблица для хранения информации о шкафах, находящихся в комнатах. Связь с комнатами осуществляется через room\_id.

desks (столы):

Дополнительная таблица для учёта столов, с указанием номера стола и ссылки на комнату через room\_id.

Связи между сущностями:

Диаграмма показывает чёткие отношения между местами хранения, этажами, комнатами, шкафами и столами. Например:

Шкафы и столы привязаны к конкретным комнатам.

Комнаты связаны с этажами.

Места хранения объединяют все эти данные для обеспечения гибкости и точности инвентаризации.

Использование QR-кодов:

Поле qr\_code в таблице storage\_locations отвечает за привязку QR-кодов к объектам или местам хранения. Это упрощает идентификацию объектов и позволяет ускорить процессы инвентаризации.

Преимущества структуры:

Модульность: База данных организована так, что её можно масштабировать, добавляя новые категории мест хранения или изменяя структуру объектов.

Гибкость: Возможность связывать активы с различными уровнями местоположений (этажи, комнаты, шкафы, столы) делает систему адаптируемой под разные сценарии. Упрощение поиска: Чёткая организация данных облегчает поиск активов по их местоположению или QR-коду.

**В процессе разработки было принято решение перейти на Nosql базу данных платформы Firebase.**

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

Рис. 7– ER-диаграмма

**Стадия 6.**

1) **Доработка интерфейса(новые/изменённые файлы):**

* list\_property.dart
* qr\_code\_generator.dart
* qr\_scan\_dialogs.dart
* list\_pos\_for\_choose.dart

**list\_property.dart:**

import 'package:flutter/material.dart';  
import 'package:flutter\_svg/flutter\_svg.dart';  
import 'Widgets.dart';  
import '/services/services.dart';  
import '/features/view/view.dart';  
class InventoryScreen extends StatefulWidget {  
 @override  
 \_InventoryScreenState createState() => \_InventoryScreenState();  
}  
class \_InventoryScreenState extends State<InventoryScreen> {  
 String currentFilter = 'Положения'; // 'Положения' или 'Имущество'  
 String? selectedFloor;  
 String? selectedRoom;  
 Location? selectedLocation;  
 final List<String> floors = ['1', '2', '3', '4', '5', '6'];  
 Map<String, Map<Location, List<Item>>> locationsWithItems = {};  
 @override  
 void initState() {  
 super.initState();  
 }  
 List<String> get currentList {  
 // Если выбран этаж, подгружаем данные  
 if (selectedFloor == null) {  
 return floors;  
 }  
 // Получаем данные для выбранного этажа  
 final data = locationsWithItems[selectedFloor];  
 if (data == null) return []; // Если данных нет, возвращаем пустой список  
 // Если выбран только этаж, показываем комнаты  
 if (selectedRoom == null) {  
 final rooms = data.keys.map((loc) => loc.room).toSet().toList();  
 rooms.sort();  
 return rooms;  
 }  
 // Если выбран только кабинет, показываем его описание  
 if (selectedLocation == null) {  
 final roomLocations = data.keys.where((loc) => loc.room == selectedRoom).toList();  
 return roomLocations.map((loc) => loc.type).toList(); // Используем поле type для отображения  
 }  
 // Если выбрана вещь, показываем её имя  
 final items = data[selectedLocation] ?? [];  
 return items.map((item) => item.name).toList();  
 }  
 Future<void> fetchLocationsAndItems(String floor) async {  
 try {  
 final fetched = await FirebaseService().getLocationsWithItemsByFloor(floor);  
 setState(() {  
 locationsWithItems[floor] = fetched;  
 });  
 } catch (e) {  
 print('Ошибка при получении данных: $e');  
 }  
 }  
 void handleTap(String tapped) {  
 setState(() {  
 if (selectedFloor == null) {  
 selectedFloor = tapped;  
 fetchLocationsAndItems(tapped);  
 } else if (selectedRoom == null) {  
 selectedRoom = tapped;  
 } else if (selectedLocation == null) {  
 final loc = locationsWithItems[selectedFloor]!.keys.firstWhere(  
 (l) => l.room == selectedRoom && (l.type) == tapped, // Исправлено на type  
 );  
 selectedLocation = loc;  
 }  
 });  
 }  
 void handleBack() {  
 setState(() {  
 if (selectedLocation != null) {  
 selectedLocation = null;  
 } else if (selectedRoom != null) {  
 selectedRoom = null;  
 } else if (selectedFloor != null) {  
 selectedFloor = null;  
 }  
 });  
 }  
 String get currentPath {  
 if (selectedLocation != null) return selectedLocation!.type; // Используем type для отображения  
 if (selectedRoom != null) return selectedRoom!;  
 return selectedFloor ?? '';  
 }  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 body: Column(  
 children: [  
 // Панель фильтров  
 Padding(  
 padding: const EdgeInsets.all(8.0),  
 child: Row(  
 mainAxisAlignment: MainAxisAlignment.spaceAround,  
 children: [  
 \_buildFilterButton('Положения'),  
 \_buildFilterButton('Имущество'),  
 SizedBox(width: 50),  
 IconButton(  
 icon: Icon(Icons.*filter\_list*),  
 iconSize: 35,  
 onPressed: () {  
 if (selectedFloor != null)  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 8.0),  
 child: Row(  
 children: [  
 IconButton(  
 icon: Icon(Icons.*arrow\_back*, size: 30),  
 onPressed: handleBack,  
 ),  
 Text(  
 currentPath,  
 style: TextStyle(fontSize: 18, fontWeight: FontWeight.*w500*),  
 ),  
 ],  
 ),  
 ),  
  
 Expanded(  
 child: ListView.separated(  
 itemCount: currentList.length,  
 itemBuilder: (context, index) {  
 final item = currentList[index];  
 String? subtitle;  
  
 // Получаем описание для элемента, если это последний уровень  
 if (selectedLocation != null) {  
 final items = locationsWithItems[selectedFloor]?[selectedLocation];  
 final selectedItem = items?.firstWhere(  
 (i) => i.name == item,  
 orElse: () => Item(id: '', name: '', location: selectedLocation!, description: ''),  
 );  
 subtitle = selectedItem?.description;  
 }  
  
 return InkWell(  
 onTap: () => handleTap(item),  
 borderRadius: BorderRadius.circular(16.0),  
 child: Container(  
 padding: const EdgeInsets.all(16),  
 decoration: BoxDecoration(  
 color: selectedFloor != null && item == currentPath  
 ? Colors.*blue*.shade100  
 : Colors.*white*,  
 borderRadius: BorderRadius.circular(16.0),  
 border: Border.all(  
 color: selectedFloor != null && item == currentPath  
 ? Colors.*blue* : Colors.*grey*.shade300,  
 width: 1.5,  
 ),  
 boxShadow: [  
 BoxShadow(  
 color: Colors.*black12*,  
 blurRadius: 4,  
 offset: Offset(0, 2),  
 ),  
 ],  
 ),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text(  
 selectedFloor == null  
 ? 'Этаж $item'  
 : item,  
 style: const TextStyle(  
 fontSize: 18,  
 fontWeight: FontWeight.*bold*,  
 color: Colors.*black*,  
 ),  
 ),  
 if (subtitle != null && subtitle.isNotEmpty)  
 Padding(  
 padding: const EdgeInsets.only(top: 4.0),  
 child: Text(  
 subtitle!,  
 style: TextStyle(fontSize: 15, color: Colors.*black54*),],  
  
 separatorBuilder: (context, index) => Divider(color: Colors.*grey*.shade300, thickness: 0.4),  
 ),  
 ),  
  
 // Кнопки навигации  
 Padding(  
 padding: const EdgeInsets.all(20.0),  
 child: Row(  
 mainAxisAlignment: MainAxisAlignment.spaceAround,  
 children: [  
 IconButton(  
 onPressed: () {},  
 icon: SvgPicture.asset('assets/icons/home\_p/edit.svg'),  
 iconSize: 35,  
 ),  
 IconButton(  
 onPressed: () {  
 showDialog(  
 context: context,  
 builder: (context) => Dialog(  
 shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),  
 child: ClipRRect(  
 borderRadius: BorderRadius.circular(16),  
 child: SizedBox(  
 width: double.*infinity*,  
 height: 400,  
 child: QRScanScreen(),  
  
 icon: SvgPicture.asset('assets/icons/home\_p/qr\_code.svg'),  
 iconSize: 58,  
 ),  
 IconButton(  
 onPressed: () {  
 showModalBottomSheet(  
 context: context,  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.vertical(top: Radius.circular(20)),  
 ),  
 builder: (context) {  
 return Container(  
 padding: EdgeInsets.all(16.0),  
 decoration: BoxDecoration(  
 color: Colors.*white*,  
 borderRadius: BorderRadius.vertical(top: Radius.circular(20)),  
 ),  
 child: Column(  
 mainAxisSize: MainAxisSize.min,  
 children: [  
 ListTile(  
 leading: Icon(Icons.*location\_on*, color: Colors.*blue*),  
 title: Text("Положение", style: TextStyle(fontSize: 18)),  
 onTap: () {  
 Navigator.*pop*(context);  
 Navigator.*pushNamed*(context, '/new\_pos');  
 },  
 ),  
 Divider(thickness: 1),  
 ListTile(  
 leading: Icon(Icons.*add\_circle\_outline*, color: Colors.*green*),  
 title: Text("Вещь", style: TextStyle(fontSize: 18)),  
 onTap: () {  
 Navigator.*pop*(context);  
 Navigator.*pushNamed*(context, '/new\_thing');  
  
 icon: SvgPicture.asset('assets/icons/home\_p/add.svg'),  
 iconSize: 35,  
  
  
 Widget \_buildFilterButton(String filter) {  
 return ElevatedButton(  
 onPressed: () {  
 setState(() {  
 currentFilter = filter;  
 selectedFloor = null;  
 selectedRoom = null;  
 selectedLocation = null;  
 });  
 },  
 child: Text(  
 filter,  
 style: TextStyle(  
 fontSize: 15,  
 color: currentFilter == filter ? Colors.*white* : Colors.*black*,  
 ),  
 ),  
 style: ElevatedButton.*styleFrom*(  
 backgroundColor: currentFilter == filter ? Colors.*black87* : Colors.*white*,  
 shape: RoundedRectangleBorder(  
 borderRadius: BorderRadius.circular(50.0),  
 ),  
 padding: EdgeInsets.symmetric(horizontal: 15, vertical: 10),

**qr\_code\_generator.dart:**

import 'package:flutter\_zxing/flutter\_zxing.dart';  
import 'dart:convert';  
import '/features/view/view.dart';  
String generateQRCode(Location location) {  
 final Map<String, dynamic> locationData = location.toJson();  
 final String jsonString = jsonEncode(locationData);  
 return jsonString;  
}

**qr\_scan\_dialogs.dart:**

import 'package:flutter/material.dart';  
import '/features/view/view.dart';  
  
void showItemDialog(BuildContext context, Item item) {  
 showDialog(  
 context: context,  
 builder: (context) => AlertDialog(  
 title: Text('Информация о вещи'),  
 content: Column(  
 mainAxisSize: MainAxisSize.min,  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text('Название: ${item.name}'),  
 Text('Описание вещи: ${item.description}'),  
 SizedBox(height: 10),  
 Text('Расположение:'),  
 Text('Этаж: ${item.location.floor}'),  
 Text('Кабинет: ${item.location.room}'),  
 Text('Тип: ${item.location.type}'),  
 if (item.location.description != null)  
 Text(' Описание положения: ${item.location.description}'),  
 ],  
 ),  
 actions: [  
 TextButton(  
 onPressed: () => Navigator.of(context).pop(),  
 child: Text('ОК'),  
 ),  
 ],  
 ),  
 );  
}  
  
void showLocationDialog(BuildContext context, Location location) {  
 showDialog(  
 context: context,  
 builder: (context) => AlertDialog(  
 title: Text('Информация о положении'),  
 content: Column(  
 mainAxisSize: MainAxisSize.min,  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text('Этаж: ${location.floor}'),  
 Text('Кабинет: ${location.room}'),  
 Text('Тип: ${location.type}'),  
 if (location.description != null)  
 Text('Описание: ${location.description}'),  
 ],  
 ),  
 actions: [  
 TextButton(  
 onPressed: () => Navigator.of(context).pop(),  
 child: Text('ОК'),  
 ),  
 ],  
 ),  
 );  
}

**list\_pos\_for\_choose.dart:**  
import 'package:flutter/material.dart';  
import '../../services/firebase\_service.dart'; // правильный импорт твоего сервиса  
import '../view/Items.dart'; // правильный импорт модели Location  
class LocationSelector extends StatefulWidget {  
 final String floor;  
 final String room;  
 final ValueChanged<Location> onLocationSelected;  
 const LocationSelector({  
 Key? key,  
 required this.floor,  
 required this.room,  
 required this.onLocationSelected,  
 }) : super(key: key);  
 @override  
 \_LocationSelectorState createState() => \_LocationSelectorState();  
}  
class \_LocationSelectorState extends State<LocationSelector> {  
 Location? \_selectedLocation;  
 List<Location> \_locations = [];  
  
 @override  
 void initState() {  
 super.initState();  
 \_loadLocations();  
 }  
 void \_loadLocations() async {  
 final firebaseService = FirebaseService();  
 final locations = await firebaseService.getFilteredLocations(  
 widget.floor,  
 widget.room,  
 );  
 setState(() {  
 \_locations = locations;  
 });  
 }  
 void \_selectLocation(Location location) {  
 setState(() {  
 \_selectedLocation = location;  
 });  
 widget.onLocationSelected(location); // уведомляем родителя  
 }  
 @override  
 Widget build(BuildContext context) {  
 return ListView.separated(  
 shrinkWrap: true,  
 physics: NeverScrollableScrollPhysics(),  
 itemCount: \_locations.length,  
 separatorBuilder: (context, index) => const SizedBox(height: 8), // Меньший отступ  
 itemBuilder: (context, index) {  
 final location = \_locations[index];  
 final isSelected = \_selectedLocation?.id == location.id;  
  
 return InkWell(  
 onTap: () => \_selectLocation(location),  
 borderRadius: BorderRadius.circular(16.0),  
 child: Container(  
 padding: const EdgeInsets.all(16),  
 decoration: BoxDecoration(  
 color: isSelected ? Colors.blue.shade100 : Colors.white,  
 borderRadius: BorderRadius.circular(16.0),  
 border: Border.all(  
 color: isSelected ? Colors.blue : Colors.grey.shade300,  
 width: 1.5,  
 ),  
 boxShadow: [  
 BoxShadow(  
 color: Colors.black12,  
 blurRadius: 4,  
 offset: Offset(0, 2),  
 )  
 ],  
 ),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text(  
 location.type,  
 style: const TextStyle(  
 fontSize: 18, // Чуть меньше, чем в текстовом поле  
 fontWeight: FontWeight.bold,  
 color: Colors.black,  
 ),  
 ),  
 if (location.description != null && location.description!.isNotEmpty)  
 Padding(  
 padding: const EdgeInsets.only(top: 4.0),  
 child: Text(  
 location.description!,  
 style: const TextStyle(  
 fontSize: 15,  
 color: Colors.black54,

2) **Интеграция Firebase в приложение. Библиотеки(зависимости), которые использовались:**firebase\_core: ^3.6.0  
firebase\_auth: ^5.3.1  
cloud\_firestore: ^5.4.4  
firebase\_messaging: ^15.1.3  
firebase\_analytics: ^11.3.3  
firebase\_database: ^11.1.4  
После интеграции Firebase, в виду «устарелости» проекта, требовалось полное обновление его конфигурации(AGP Gradle, Gradle, Kotlin Sdk, Flutter SDK, JDK) для его полноценной работы в как с данной библиотекой, так и в дальнейшем с новыми инструментами.

3) **Создание БД , коллекций и разработка бекэнда.**

**firebase\_service.dart(**реализация запросов**):**

import 'package:cloud\_firestore/cloud\_firestore.dart';  
import '/features/view/view.dart'; // Подкорректируй путь под структуру проекта  
class FirebaseService {  
 final FirebaseFirestore \_firestore = FirebaseFirestore.*instance*;  
 // Получаем ссылку на новый документ  
 Future<DocumentReference> getDocumentReference() async {  
 return \_firestore.collection('locations').doc();  
 }  
  
 // Добавление локации  
 Future<void> addLocation(Location location, DocumentReference docRef) async {  
 try {  
 await docRef.set({  
 'id': location.id,  
 'floor': location.floor,  
 'room': location.room,  
 'type': location.type,  
 'description': location.description,  
 'qrData': 'location/${docRef.id}',  
 'createdAt': FieldValue.*serverTimestamp*(),  
 });  
 } catch (e) {  
 print('Error adding location: $e');  
 rethrow;  
 }  
 }  
 // Добавление предмета  
 Future<void> addItem(Item item, DocumentReference docRef) async {  
 try {  
 await docRef.set({  
 'id': item.id,  
 'name': item.name,  
 'description': item.description,  
 'location': item.location.toJson(),  
 'qrData': 'item/${docRef.id}',  
 'createdAt': FieldValue.*serverTimestamp*(),  
 });  
 } catch (e) {  
 print('Error adding item: $e');  
 rethrow;  
 }  
 }  
 // Получение локаций по этажу и кабинету  
 Future<List<Location>> getFilteredLocations(String floor, String room) async {  
 try {  
 final snapshot = await \_firestore  
 .collection('locations')  
 .where('floor', isEqualTo: floor)  
 .where('room', isEqualTo: room)  
 .get();  
 return snapshot.docs.map((doc) => Location.fromFirestore(doc)).toList();  
 } catch (e) {  
 print('Error fetching locations: $e');  
 rethrow;  
 }  
 }  
 // Получение всех локаций по этажу  
 Future<List<Location>> getLocationsByFloor(String floor) async {  
 try {  
 final snapshot = await \_firestore  
 .collection('locations')  
 .where('floor', isEqualTo: floor)  
 .get();  
 return snapshot.docs.map((doc) => Location.fromFirestore(doc)).toList();  
 } catch (e) {  
 print('Error fetching locations: $e');  
 rethrow;  
 }  
 }  
 // Получение всех предметов, у которых location.floor == floor  
 Future<List<Item>> getItemsByFloor(String floor) async {  
 try {  
 final snapshot = await \_firestore  
 .collection('items')  
 .where('location.floor', isEqualTo: floor)  
 .get();  
  
 return snapshot.docs.map((doc) => Item.fromFirestore(doc)).toList();  
 } catch (e) {  
 print('Error fetching items: $e');  
 rethrow;  
 }  
 }  
 // Получение карты {Location: [Item, ...]} по этажу  
 Future<Map<Location, List<Item>>> getLocationsWithItemsByFloor(String floor) async {  
 try {  
 final locations = await getLocationsByFloor(floor);  
 final items = await getItemsByFloor(floor);  
  
 final Map<Location, List<Item>> result = {};  
  
 for (final location in locations) {  
 final locationItems = items.where((item) => item.location.id == location.id).toList();  
 result[location] = locationItems;  
 }  
  
 return result;  
 } catch (e) {  
 print('Error building location-item map: $e');  
 rethrow;  
 }  
 }  
}

3) **Реализация скана и генераторов Qr-кодов.**

Были использованы библиотеки:

* qr\_flutter: ^4.0.0
* mobile\_scanner: ^6.0.10

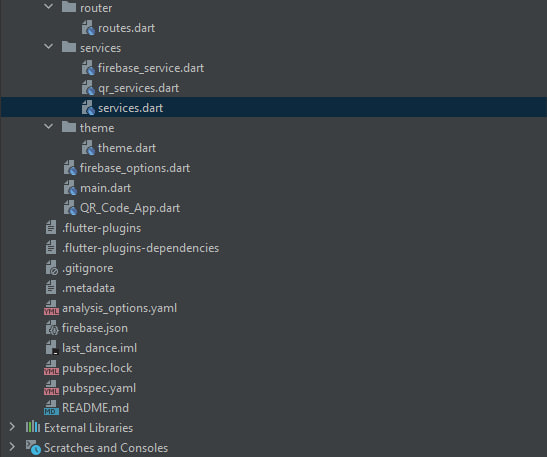
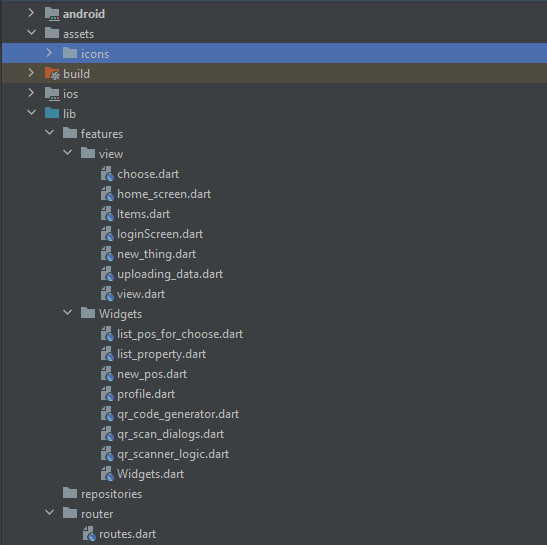
4)Текущая версия проекта(директория):  
 

Рис. 8 – текущая директория проекта

# **ПЛАНИРОВАНИЕ И РАСПРЕДЕЛЕНИЕ ЗАДАЧ ПО ПРОЕКТУ**

В процессе планирования и распределения задач для проекта было выделено несколько ключевых аспектов, которые помогут организовать эффективное выполнение всех этапов работы.

Конкретной целью проекта является разработка мобильного приложения, которое будет использовать QR-коды для инвентаризации объектов, таких как вещи, помещения и шкафы, с учётом их местоположения. Приложение должно предоставлять пользователю возможность сканировать QR-коды, получать информацию об объекте, а также редактировать её, если права доступа позволяют. Для этого необходимо создать функциональную базу данных, которая будет содержать информацию о местоположении объектов и их описание.

Измеримая цель заключается в создании рабочего прототипа с основным функционалом, включая генерацию и сканирование QR-кодов, а также интеграцию базы данных с объектами и их местоположениями. При этом требуется обеспечить успешное считывание QR-кодов с точностью 95% за 1 секунду.

Для достижения этой цели проект будет использовать популярные технологии, такие как PostgreSQL для базы данных и Flutter&Dart для кроссплатформенной разработки мобильных приложений. Команда из 2-3 разработчиков сможет реализовать проект с использованием доступных инструментов в установленный срок.

Актуальность проекта обусловлена необходимостью ускорения процесса инвентаризации и повышения точности учёта для организаций. Приложение решает проблему автоматизации учёта активов и товаров, что способствует снижению ошибок в учёте на 30%. Это решение важно для бизнеса и учреждений, которые стремятся повысить эффективность работы.

Время выполнения проекта также имеет чёткие рамки: минимально жизнеспособный продукт (MVP) должен быть готов через 5 месяцев, а полная версия приложения с тестированием и исправлением ошибок — через 7 месяцев.

Распределение задач по проекту:

1. Анализ и моделирование процесса: на этом этапе команда будет моделировать процесс инвентаризации с использованием различных инструментов, таких как Miro и Drawio. Задача будет выполнена с 1 октября 2024 года по 15 ноября 2024 года.
2. Исправление ошибок после демонстрации: после завершения основного этапа разработки приложение будет протестировано, и на основе обратной связи будут внесены исправления. Задача будет выполняться с 1 мая 2025 года по 16 мая 2025 года.
3. Подготовка отчёта и презентации за 1-й семестр: команда подготовит отчёт с результатами 1-го семестра разработки. Задача будет выполнена с 25 декабря 2024 года по 8 января 2025 года.
4. Подготовка отчёта и презентации за 2-й семестр: по завершении второго семестра будет подготовлен отчёт с итогами работы. Срок выполнения задачи — с 16 мая 2025 года по 26 мая 2025 года.
5. Разработка базы данных и интеграция её в приложение: этот этап включает создание и настройку базы данных, которая будет хранить информацию об объектах и их местоположении. Срок выполнения — с 1 октября 2024 года по 1 мая 2025 года.
6. Разработка приложения: на этом этапе будет разрабатываться мобильное приложение с использованием Flutter&Dart. Прототип приложения будет готов к 1 января 2025 года. Основная работа будет завершена к 1 мая 2025 года.
7. Разработка прототипа дизайна: задача по созданию прототипа дизайна будет завершена в начале 2025 года. Используемый инструмент — Figma.
8. Тестирование и демонстрация заказчику: в апреле 2025 года приложение будет протестировано, и заказчику будет предоставлена демонстрация работы продукта для сбора обратной связи.

Внутренние факторы проекта включают несколько сильных и слабых сторон. К сильным сторонам можно отнести возможность проведения инвентаризации без постоянного подключения к интернету, что позволяет синхронизировать данные позже, а также то, что приложение значительно упрощает процессы инвентаризации, экономит время сотрудников и минимизирует ошибки при ручной обработке данных. Однако существует слабость, связанная с тем, что в первой версии приложения могут быть недочёты, а также ограниченная функциональность для специфических потребностей некоторых компаний или складов.

Среди внешних факторов проект имеет как возможности, так и угрозы. Одной из возможностей является рост интереса к цифровым решениям в рамках цифровизации бизнеса, а также переход компаний на мобильные приложения для повышения точности учёта и снижения времени, затрачиваемого на инвентаризацию. Однако существует угроза, что пользователи будут ожидать максимально простых и удобных интерфейсов, что потребует дополнительных усилий для дизайна. Также есть риск утечки данных, что может повредить репутации компании и потребовать значительных инвестиций в защиту данных.

Таким образом, проект имеет четко определённый план и задачи, которые будут выполнены в рамках установленных сроков с использованием проверенных технологий.

# **ЗАКЛЮЧЕНИЕ**

В рамках данного проекта была достигнута его основная цель было создано веб-приложение, которое значительно улучшает процесс инвентаризации с использованием QR-кодов. Интеграция современных технологий автоматизированного учёта с интуитивно понятным интерфейсом позволяет упростить и ускорить процедуру инвентаризации для пользователей на любом устройстве с интернет-браузером.

Новизна проекта заключается в внедрении функционала динамических QR-кодов, что дает возможность обновлять данные инвентаризации в реальном времени, не требуя создания новых кодов. Кроме того, приложение разработано с учётом масштабируемости, что позволяет его адаптировать как для малых предприятий, так и для крупных корпораций с различными требованиями и настройками.

Предложенное решение не только способствует снижению затрат, но и повышает точность учета, что делает его привлекательным для широкого круга организаций. Внедрение данного приложения открывает новые возможности для автоматизации бизнес-процессов, улучшая эффективность работы и снижая вероятность ошибок в учете.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ**

1. Сайт hexlet.io [Электронный ресурс] URL: [Базы данных в Python: как подключить PostgreSQL и что это такое (hexlet.io)](https://ru.hexlet.io/blog/posts/python-postgresql)
2. Сайт zetcode.com [Электронный ресурс] URL: [Python openpyxl - read, write Excel xlsx files in Python (zetcode.com)](https://zetcode.com/python/openpyxl/)
3. Сайт docs.flutter.dev [Электронный ресурс] URL: [Flutter documentation | Flutter](https://docs.flutter.dev/)
4. Сайт docs.flutter.dev [Электронный ресурс] URL: [Codelabs | Flutter](https://docs.flutter.dev/codelabs)
5. Сайт dart.dev [Электронный ресурс] URL: [Dart documentation | Dart](https://dart.dev/guides)
6. Сайт github.com [Электронный ресурс] URL: [Projects · flutter · GitHub](https://github.com/flutter/flutter/projects?query=is%3Aopen)
7. Сайт habr.com/ru [Электронный ресурс] URL: [habr.com/ru/articles/172525/](https://habr.com/ru/articles/172525/)
8. Сайт habr.com/ru [Электронный ресурс] URL: habr.com/ru/articles/528320/

**ПРИЛОЖЕНИЕ 1. Сведения о проекте BusinessChain**

|  |  |
| --- | --- |
| **Номер проекта:** | *2024-1935* |
| **Название трека** | *«От реализации к извлечению уроков»* |
| **Результат выполнения:** | *Полностью* |
| **Ссылка на страницу проекта в БЧ** | *https://businesschain.io/bch/view/my-projects/edit?id=18739* |

|  |
| --- |
|  |
| *(подпись куратора с расшифровкой)* |

|  |
| --- |
| **Снимок экрана (скриншот), подтверждающий выполнение этапов трека:** |
|  |
| **Снимок экрана (скриншот), отзыва куратора проектной команды на Businesschain (последний этап трека):** |
|  |

**ПРИЛОЖЕНИЕ 2. Сведения о посещённых Мероприятиях проектного дня**

2.1. Прокопенко А.С.

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Наименование мероприятия** | **Ссылка на мероприятие в Makeevents** |
| **1** | *Разговор о мега-проектах и будущем* | *https://guu.make.events/eventsNew/r8spsl* |
| **2** | *Принятие решений на основе математического ожидания* | *https://guu.make.events/eventsNew/CPjgtr* |
| **3** | *Онтологизация целей, ценностей и смыслов* | *https://guu.make.events/eventsNew/O8Ogtl* |
| **4** | *Базы данных: SQL vs NoSQL. Какую выбрать для проекта?* | *https://guu.make.events/eventsNew/Oz9pct* |
| **5** | *Гостевая лекция ООО "Нестле Россия"* | *https://guu.make.events/eventsNew/rOxmcW* |
| **6** | *Научная публикация глазами автора и издателя* | *https://guu.make.events/eventsNew/OIhrUW?code=,c7441fff52ead6d70a4a94c9fa429100* |
| **7** | *Современные тренды эффективной продуктовой политики* | *https://guu.make.events/events/a1jg9v/?code=,c7441fff52ead6d70a4a94c9fa429100* |
| **8** | *Актуализация маркетинга услуг в современных условиях* | *https://guu.make.events/eventsNew/ruPw9t?code=,c7441fff52ead6d70a4a94c9fa429100* |
| **9** | *Имитационное моделирование бизнес-процессов* | *https://guu.make.events/eventsNew/YdLw9W* |
| **10** | *Мастер-класс "Как писать для англоязычных медиа"* | *https://guu.make.events/eventsNew/cuxAUl* |

|  |  |
| --- | --- |
| **Ссылка на профиль студента Makeevents** | *https://make.events/public/user/cOLAKW* |

|  |
| --- |
| Снимок экрана (скриншот), подтверждающий посещение мероприятия |
|  |

2.2. Воротников И.А.

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Наименование мероприятия** | **Ссылка на мероприятие в Makeevents** |
| **1** | *БРИКС: инвестиционная привлекательность пространства* | *https://guu.make.events/eventsNew/r1L7UD* |
| **2** | *Как построить успешную карьеру в крупной компании* | *https://guu.make.events/eventsNew/vOv8hp* |
| **3** | *DevOps как призвание и как профессия* | *https://guu.make.events/eventsNew/O0smhv* |
| **4** | *Семинар поддержки проектного обучения ИТ-проектов 1* | *https://guu.make.events/eventsNew/v9vwwr* |
| **5** | *Знания и умения руководителя аналитич. подразделения* | *https://guu.make.events/eventsNew/au9gtW* |
| **6** | *Будущее с нейросетями и метавселенными: что нас ждёт?* | *https://guu.make.events/eventsNew/vzjFhW* |
| **7** | *Вероятностно-статистические методы в управлении* | *https://guu.make.events/eventsNew/vPOpw3* |
| **8** | *ПО для решения задач эконометрики* | *https://guu.make.events/eventsNew/O0O8ct* |
| **9** | *Особенности реализации проектов в социальной сфере* | *https://guu.make.events/eventsNew/v0smsr* |
| **10** | *Соврем.технологии в проектах безопасной городской среды* | *https://guu.make.events/eventsNew/qdjgwt* |

|  |  |
| --- | --- |
| **Ссылка на профиль студента Makeevents** | *https://make.events/public/user/JdsgLk* |

|  |
| --- |
| Снимок экрана (скриншот), подтверждающий посещение мероприятия |
|  |

2.3 Нестеров Ф.О.

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Наименование мероприятия** | **Ссылка на мероприятие в Makeevents** |
| **1** | *Роботизация и искусственный интеллект в HR* | https://guu.make.events/eventsNew/qds7zk |
| **2** | *НЕФТЕГАЗ-2025* | *https://guu.make.events/eventsNew/C8Oghv* |
| **3** | *Эволюция денежного инструмента.Цифровой рубль.* | *https://guu.make.events/eventsNew/YIOt9r* |
| **4** | *"Предзащита" проектов проектного обучения для 3 курса* | *https://guu.make.events/eventsNew/J0sgw3* |
| **5** | *Тестирование универсальных компетенций* | *https://guu.make.events/eventsNew/KP9wtk* |
| **6** | *Клуб развития международного бизнеса и туризма* | *https://guu.make.events/eventsNew/C9LFzl* |
| **7** | *Клуб "Школа молодого психолога"* | *https://guu.make.events/eventsNew/YPh8cD* |
| **8** | *Клуб "Школа молодого психолога"* | *https://guu.make.events/eventsNew/qPjgzr* |
| **9** | *Маркетинг впечатлений: как очARовать потребителя* | *https://guu.make.events/eventsNew/JIvwcWё* |
| **10** | *Мастер-класс «Идеи для создания устойчивых проектов»* | *https://guu.make.events/eventsNew/vO9mUp* |

|  |  |
| --- | --- |
| **Ссылка на профиль студента Makeevents** | *https://make.events/public/user/cdj8tr* |

|  |
| --- |
| Снимок экрана (скриншот), подтверждающий посещение мероприятия |
|  |