# **Введение**

В настоящее время автоматизация процессов является ключевым элементом улучшения эффективности и эффективности работы различных предприятий и организаций. Одной из таких областей, где автоматизация может принести значительную выгоду, является сфера безопасности и охраны объектов. В данном отчете рассматривается проблема управления и отчетности охранников на объектах. В современных условиях большинство охранных фирм и предприятий сталкиваются с необходимостью эффективного контроля за деятельностью охранников на объектах, а также своевременного получения информации о текущем статусе работы.

В данном отчете рассматривается проблема управления и отчетности охранников на объектах. В современных условиях большинство охранных фирм и предприятий сталкиваются с необходимостью эффективного контроля за деятельностью охранников на объектах, а также своевременного получения информации о текущем статусе работы. Для решения этой проблемы предлагается разработка информационной системы, которая позволит автоматизировать управление дежурной частью охранного предприятия. Создание мобильного приложения для сотрудников охраны позволит значительно упростить процесс управления и повысить эффективность работы охраны.

Целью данного проекта является разработка мобильного приложения, которое позволит охранникам быстро и удобно сообщать о своем прибытии на объект и окончании смены. Это приложение предоставит возможность улучшить процесс отчетности, сократить время на передачу информации и повысить эффективность контроля со стороны руководства охранной фирмы.

# **Актуальность**

В ходе подготовки к разработке мобильного приложения для охранной фирмы, было выполнено комплексное исследование предметной области. Исследование выявило, что охранная индустрия активно ищет пути автоматизации своих процессов, особенно в контексте управления персоналом и отчетности. Были проанализированы тренды и типичные проблемы охранной деятельности, такие как недостаток оперативности в учете рабочего времени и отслеживании местоположения охранников.

В поисках существующих решений были проанализированы различные программные решения и мобильные приложения, используемые в охранной отрасли. Отмечена ограниченная функциональность в существующих системах, особенно в части реального времени уведомлений и отчетности. Изучены отзывы пользователей, что помогло выявить ключевые недостатки текущих решений, включая сложность использования и низкую надежность. Анализ успешных примеров автоматизации процессов в охранной деятельности выявил, что интеграция мобильных технологий значительно повышает оперативность и точность данных. Выявленные неудачные практики показали важность удобства интерфейса и стабильности работы приложений.

Автоматизация в охранной деятельности является критически важной для повышения эффективности операций и улучшения управления персоналом. Современные технологии, включая мобильные приложения и облачные сервисы, предоставляют значительные возможности для усиления контроля и оптимизации рабочих процессов.

Текущий анализ показывает, что многие охранные фирмы все еще полагаются на устаревшие методы учета рабочего времени и отчетности, что ведет к задержкам в передаче информации и увеличивает вероятность ошибок. Это подтверждается данными из отчетов и исследований, где описаны случаи недостаточной оперативности реагирования на инциденты из-за несвоевременной отчетности.

Сравнительный анализ существующих решений показал, что внедрение мобильных приложений может существенно улучшить процесс управления охраной. Например, системы с функциями мгновенной отметки о прибытии и уходе с объекта значительно повышают точность учета рабочего времени и позволяют оперативно реагировать на изменения в расписании сотрудников.

# **Описание предметной области**

Предметной областью данной работы является система по автоматизации контроля охраняемых объектов, которая позволяет автоматизировать процесс контроля персонала.

Охраннику или оперативнику после выхода на работу и прибытия на объект, необходимо уведомить оператора о заступлении на смену, чтобы руководство понимало, что этот объект находится под охраной, и, если охранник почему-то не вышел на работу, быстро найти другого сотрудника на этот объект. В течении рабочего дня охранник должен докладывать о своем статусе и уведомлять оператора о различных ситуациях на месте. После окончании рабочего дня, он также должен сообщить оператору о конце своей смены, после чего его заменит другой. Все это, на данный момент, проводится обычными звонками в дежурную часть, где оператор вручную записывает информацию в бумажные отчеты. Из-за этого на телефонной линии каждый день возникает «очередь» из звонков от охранников с разных объектов, что сильно замедляет рабочий процесс.

Эта ИС позволяет автоматизировать и упростить процесс по контролю и управлению за сменами охранников и цифровизировать все бумажные отчеты, для удобства отслеживания и анализа эффективности работы каждого сотрудника.

Данная система должна обладать следующими возможностями, такими как:

* Регистрация и управление оперативниками
* Отметка заставления на смену
* Уведомления и рассылки
* Аналитика работы охранников
* Возможность интеграции с системами безопасности
* Интерфейс для руководства

В системе по автоматизации контроля охраняемых объектов могут существовать следующие роли:

1. Сотрудник (Оперативник):

- Отметка о заступлении на смену.

- Отметка об окончании смены.

- Отправка уведомлений о неотложных событиях.

2. Дежурный:

- Просмотр списка сотрудников (кто из сотрудников заступил на смену, а кто нет)

3. Оператор (Администратор):

- Управление данными о персонале (добавление/удаление оперативников).

- Генерация отчетов о работе сотрудников.

База данных располагается на сервере предприятия и хранит информацию о сотрудниках и о статусах их смен.

Объекты в системе автоматизации контроля охраняемых объектов включают в себя различные элементы, которые подлежат мониторингу, управлению и контролю. Это информация о каждом сотруднике, такая как ФИО, должность, квалификация и иные характеристики. Информация об охраняемых объектах, такие как адрес, название, тип и идентификатор объекта. Различные уведомления, как и оперативникам от руководства, так и наоборот, будет содержать информацию о типе уведомления, его содержание и время отправки. Информация о запланированных сменах, даты смен и сотрудники в этой смене.

**Цели и задачи проекта**

Цели проекта:

1. Сбор информации о проблемах и недостатках в текущем управлении дежурной частью.

2. Анализ данных о присутствии охранников на объектах.

3. Разработка и внедрение информационной системы для автоматизации этих процессов.

4. Повышение эффективности работы охранного предприятия и улучшение уровня безопасности на объектах.

5. Упрощение работы дежурной части охранного предприятия.

Задачи проекта:

1. Изучение требований и потребностей охранного предприятия.

2. Разработка технического задания на создание мобильного приложения и веб-приложения для дежурной части.

3. Сбор данных о работе охранников и процессах управления дежурной частью.

4. Анализ собранных данных для выявления ключевых проблем и потребностей.

5. Проектирование информационной системы, включая логическую и физическую модели, а также дизайн интерфейса.

6. Разработка и тестирование мобильного приложения и декстоп-приложения.

7. Внедрение информационной системы на охранном предприятии и обучение персонала.

8. Поддержка и обновление информационной системы, включая исправление багов и добавление новых функций.

# **Диаграммы**

**Диаграмма прецедентов**

Диаграмма прецедентов (вариантов использования) помогает визуализировать взаимодействие между актерами и системой, а также описывает различные сценарии использования системы.

В контексте моей системы по автоматизации контроля охраняемых объектов, возможны следующие прецеденты: представлены 3 актера оперативник(охранник), сотрудник дежурной части, оператор. Оператор управляет данными сотрудников, оперативник отмечает свой статус и может сообщить о ЧП, дежурный обрабатывает данные от оперативника и вносит в систему при необходимости.

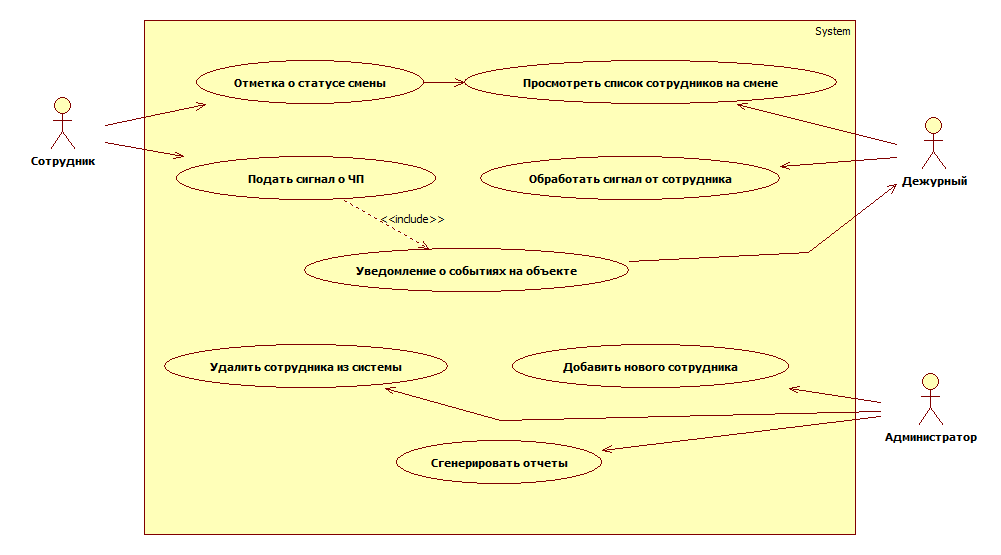


Рис.1 Диаграмма прецедентов

**Диаграмма классов**

Диаграмма классов представляет собой визуальное представление структуры классов в системе и их взаимосвязей, демонстрирует общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними.

Определим основные классы для данной системы по автоматизации контроля охраняемых объектов. На диаграмме изображен классы оперативник, оператор, дежурный, БД, система учета смен, журнал событий. Все классы имеют различные атрибуты такие как ФИО, должность, дата и другие. Также они имеют различные методы, оперативник содержит методы по заступлению и окончанию смен, подачи сигнала о ЧП. Дежурный может обработать сигнал ЧП от оперативника, добавить записи в журнал событий. БД, к примеру, позволяет добавить, удалить, изменить и получить различную информацию.

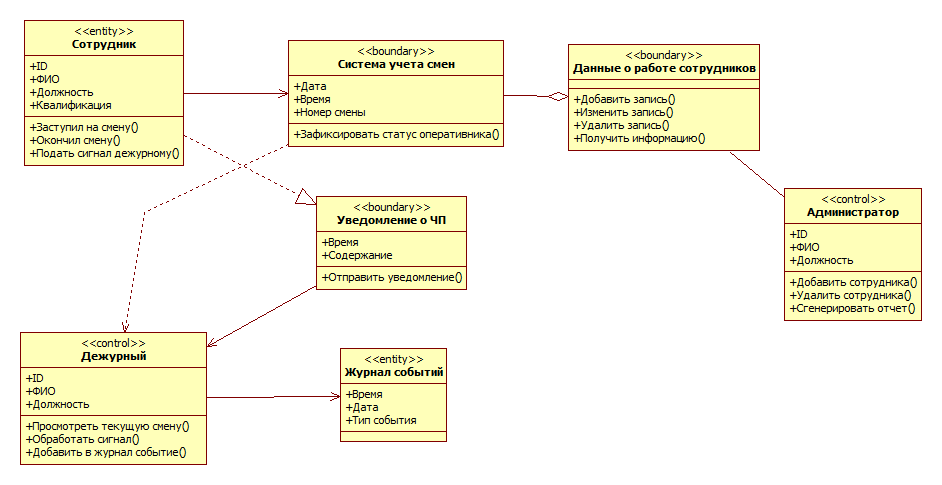


Рис.2 Диаграмма классов

**Диаграмма состояний**

Диаграмма состояний показывает, как объект в системе переходит между различными состояниями в ответ на события. Она показывает, как объект переходит из одного состояния в другое. В моем случае мы можем рассмотреть диаграмму состояний для объекта "Оперативник/Охранник" в системе контроля охраняемых объектов. На диаграмме представлено состояния охранника ЧОП. В начале он находится в ожидании смены, после заступления на смену, начинается активная фаза работы, в случае возникновения каких-либо проблем статус становится на паузе, т.е. сотрудник ожидает прибытия группы быстрого реагирования и решения проблемы. После, сотрудник продолжает свою работу и под конец рабочего дня завершает свою смену.

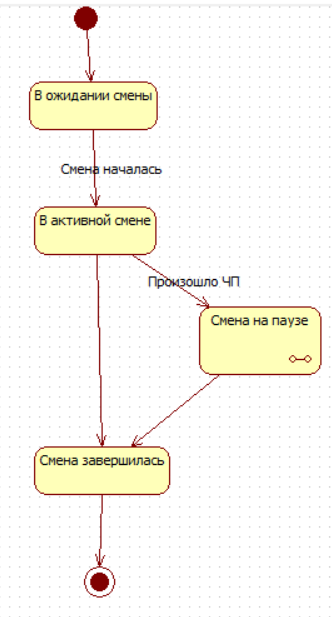


Рис.3 Диаграмма состояний

**Диаграмма последовательностей**

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями. Впрочем, часто возвращаемые результаты обозначают лишь в том случае, если это не очевидно из контекста.

На моей диаграмме представлены 3 объекта: Оперативник, Система и Дежурный. Поток управления начинается с того что оперативник заступает на смену и отмечается в системе, система оповещает дежурного об этом. В течении смены у оперативника может возникнуть ЧП, об этом он сообщает дежурному через систему, дежурный обрабатывает эту информацию, добавляет запись в журнал событий, находящийся в системе. В конце смены оперативник снова отмечается в системе, которая оповещает дежурного.

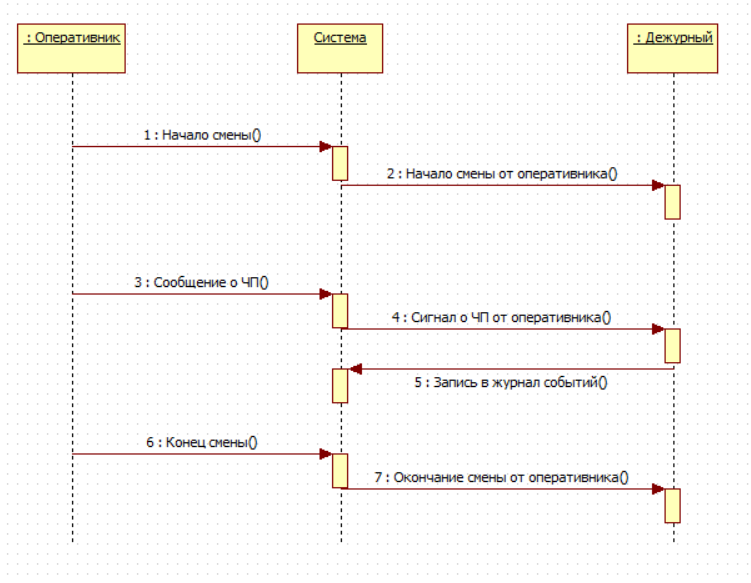
****

Рис.4 Диаграмма последовательностей

**Диаграмма активности**

Диаграмма активности позволяет более детально визуализировать конкретный случай использования. Это поведенческая диаграмма, которая иллюстрирует поток деятельности через систему.

Давайте рассмотрим пример диаграммы активности для сценария "Оперативник отмечает начало и конец смены, а также сообщает о ЧП". На диаграмме представлены основные шаги, совершаемые оперативником, системой и дежурным в контексте данного сценария. Шаги представлены стрелками, указывающими направление выполнения действий. Поток управления начинается с того что оперативник начинает смену, отмечается через систему, в течении дня происходит какое-то событие, и, если его ему не удается решить, он сообщает о ЧП через систему дежурному. Дежурный собирает информацию вызывает помощь на этот объект и вносит итоги в систему. По завершению смены оперативник снова отмечается в системе и активность завершается.

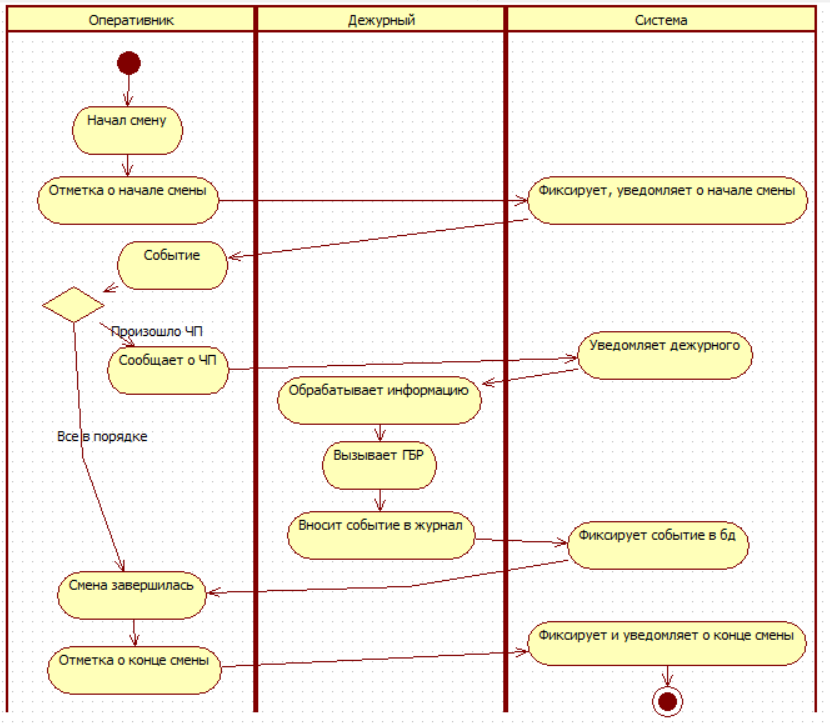


Рис.5 Диаграмма активности

**Диаграмма развертывания**

Диаграмма развертывания в UML используется для отображения физической структуры системы, показывая, как программное обеспечение, аппаратное обеспечение и сети взаимодействуют друг с другом. Она отображает размещение компонентов системы на физических устройствах и их взаимодействие через сети связи.

На диаграмме представлены: Сервер БД, где находятся СУБД, БД всех аттестаций, а также БД всех сотрудников и их данных. Управление осуществляется с ПК оператора, который связан с БД посредством сети Интернет. Сотрудники взаимодействуют с системой путем мобильного приложения, которое также сообщается с БД посредством сети Интернет.

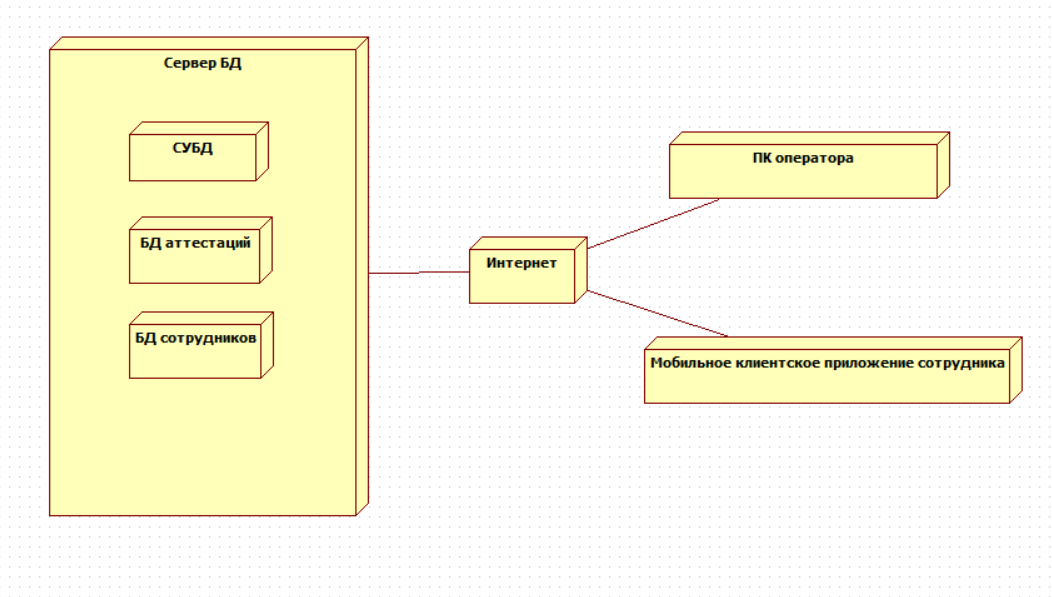
****

Рис.6 Диаграмма развертывания

**Диаграмма компонентов**

Диаграммы компонентов используются для визуализации организации компонентов системы и зависимостей между ними. Они позволяют получить высокоуровневое представление о компонентах системы.

Компонентами могут быть программные компоненты, такие как база данных или пользовательский интерфейс; или аппаратные компоненты, такие как схема, микросхема или устройство; или бизнес-подразделение, такое как поставщик, платежная ведомость или доставка.

На моей диаграмме представлены: Клиентское приложение, хранящее в себе GUI- графический интерфейс сотрудника, а также уведомления об аттестациях, веб сервер, занимающийся обработкой запросов, сервер приложения осуществляющий логику управления сотрудниками и аттестациями, а также несущий интерфейс оператора, а также база данных, что аккумулирует себе средства управления базой данных и хранит все данные.

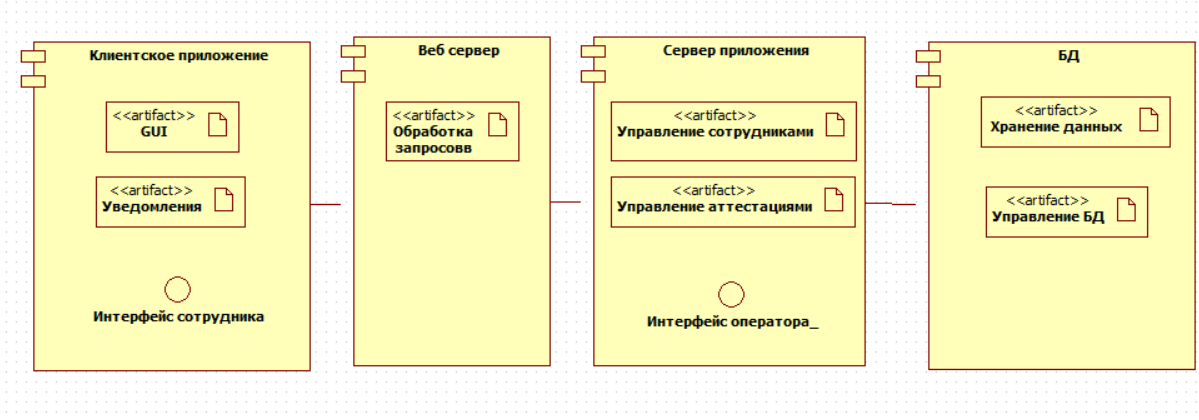
****

Рис.7 Диаграмма компонентов

## **Описание разрабатываемого продукта**

Презентационный слой мобильного приложения включает в себя все элементы пользовательского интерфейса, которые видит и с которыми взаимодействует пользователь. Это включает в себя различные экраны приложения, элементы управления, такие как кнопки и поля ввода, графические компоненты, такие как изображения и иконки, стилизацию текста и другие визуальные атрибуты, а также анимации и переходы между экранами.

В нашем мобильном приложении имеется 4 экрана:

1. Экран авторизации (LoginScreen) – он содержит изображение и текстовое приветствие, поля для ввода логина и пароля, а также кнопку «Войти» при нажатии на которую происходит попытка авторизации пользователя. При вводе правильного логина и пароля пользователь перенаправляется на главный экран (MainScreen). При неправильном вводе отображается диалоговое окно с сообщением об ошибке.

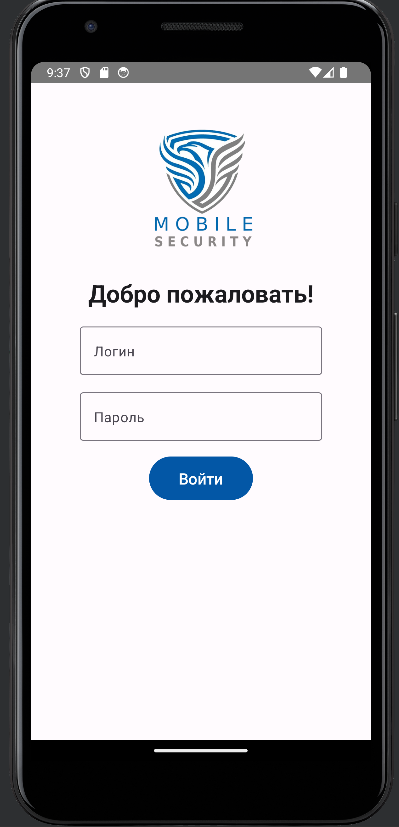


Рис.8 «LoginScreen»

1. Главный экран (MainScreen) – он содержит две кнопки – «Аттестация» и «Смена». При нажатии на кнопку «Аттестация» совершается переход на экран аттестации(AttestationScreen), при нажатии на кнопку «Смена» осуществляется переход на экран смена(ShiftScreen).

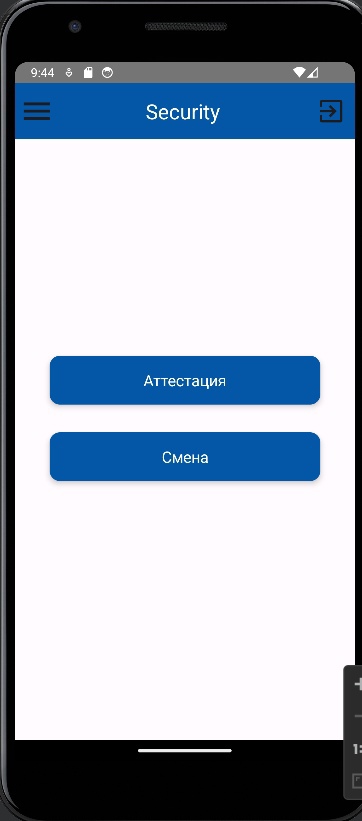


Рис.9 «MainScreen»

1. Экран аттестации (AttestationScreen) – он отображает список аттестаций пользователя в виде карточек с названием, датой прохождения и статусом. Пользователь может просматривать список аттестаций. По завершении просмотра можно вернуться на предыдущий экран с помощью кнопки "Назад".

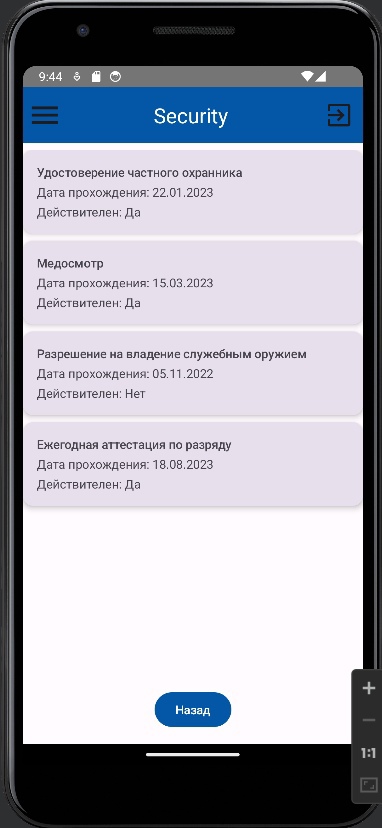


Рис.10 «AttestationScreen»

1. Экран смена (ShiftScreen) - Экран показывает информацию о текущей смене: объект, адрес, дата, время начала и окончания, а также предоставляет возможность начать или завершить смену. Также есть кнопка для возврата на главный экран.

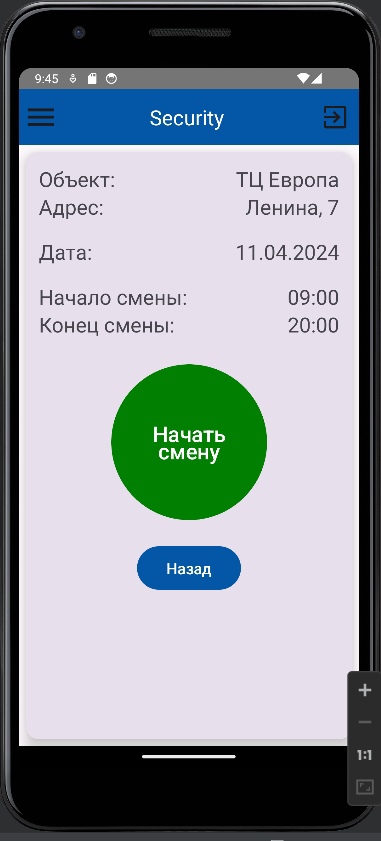


Рис.11 «ShiftScreen»

Кроме этого используются дополнительные компоненты в виде рисунков, надписей, иконок, элементов Material Design, а также элемент Header – он содержит кнопку перехода на главный экран, название приложения, а также кнопку выхода из системы.

Переход между экранами осуществляется с помощью компонента NavController. NavController - это компонент архитектуры навигации, предоставляемый Android Jetpack, который облегчает управление фрагментами и активностями в приложении. Он представляет собой централизованный объект, который обеспечивает связь между различными точками назначения в приложении и позволяет осуществлять переходы между ними.

Взаимодействие с NavController осуществляется через методы, такие как navigate() для перехода на новый фрагмент и popBackStack() для возврата к предыдущему состоянию приложения. В данной работе этот компонент является основным методом для перехода между экранами. Осуществляется это с помощью нажатий на кнопки и метода navigate. К примеру Button() {onClick = {navController.navigate(main screen)}}, после чего происходит переключение на главный экран. Пример использования popBackStack - Button() {onClick = {navController. popBackStack ()}}

Кроме этого используются дополнительные компоненты в виде рисунков, надписей, иконок, элементов Material Design, а также элемент Header – он содержит кнопку перехода на главный экран, название приложения, а также кнопку выхода из системы.

Для разработки приложения была выбрана платформа Android, как популярная и самая распространенная платформа и с учетом целевой аудитории. Язык программирования использовался Kotlin, как современный, безопасный и поддерживаемый Google язык.

Kotlin – высокоуровневый язык программирования общего применения, который является надстройкой над Java и разработан JetBrains – русской компанией разработчиков. Язык поддерживается компанией Google и используется в качестве основного языка для разработки приложений на Android в официальной SDK – Android Studio. Подобно Java, Kotlin – язык с сильной статической типизацией. Kotlin – объектно-ориентированный язык, при этом язык поддерживает широкий выбор возможностей для функционального программирования.

Язык обратно совместим с Java и выполняется в виртуальной среде Java – JVM. Также Kotlin поддерживает пакеты расширений Java. При этом, Kotlin обладает своими пакетами расширений, некоторая часть которых – доработанные пакеты Java. Kotlin и Java интероперабельны – их код может сосуществовать в одном приложении. При этом, Kotlin обладает более компактным и читаемым синтаксисом, а также рядом дополнительных функций, который нет в Java. Kotlin позволяет избежать некоторых ошибок, которые возникают в Java – например, NullPointerException.

Для повышения эффективности и удобства программирования на Kotlin, была использована технология Jetpack Compose. Фреймворк для UI, для интерактивного и реактивного интерфейса.

Jetpack Compose - это современный инструментарий для создания пользовательских интерфейсов в приложениях Android, разработанный компанией Google. Этот декларативный фреймворк позволяет разработчикам создавать динамические и интерактивные пользовательские интерфейсы, используя набор компонентов и функций Kotlin.

Основная идея Jetpack Compose заключается в том, чтобы позволить разработчикам описывать интерфейс приложения как функцию состояния, которая реагирует на изменения данных и автоматически обновляет пользовательский интерфейс при необходимости. Это обеспечивает простоту и четкость кода, а также повышает производительность приложения за счет улучшенного управления рендерингом и ресурсами.

Благодаря своей декларативной природе, Jetpack Compose сокращает объем кода, улучшает читаемость и поддерживает более эффективное тестирование пользовательского интерфейса. Это делает его мощным инструментом для разработки современных и привлекательных приложений Android.

Android Studio — интегрированная среда разработки производства Google, представляет собой мощный инструмент, специально адаптированный для работы с мобильными приложениями. Android Studio можно установить на Windows, Mac и Linux. Android Studio создавалась на базе IntelliJ IDEA. IDE можно загрузить и пользоваться бесплатно. В ней присутствуют макеты для создания UI, с чего обычно начинается работа над приложением. В Studio содержатся инструменты для разработки решений для смартфонов и планшетов.

**База данных**

База данных написана на программном обеспечении IntelliJ IDEA. IntelliJ IDEA (также известная как IntelliJ) — это интегрированная среда разработки (IDE) для программирования на различных языках, таких как Java, Kotlin, Groovy, Scala и других. Это продукт компании JetBrains, предназначенный для повышения производительности разработчиков и обеспечения удобной среды для создания программного обеспечения. IntelliJ IDEA предоставляет множество функций и инструментов для разработки приложений, включая подсветку синтаксиса, автодополнение кода, интегрированные системы сборки, отладчик, рефакторинг кода, управление зависимостями, поддержку контроля версий и многое другое. Одной из ключевых особенностей IntelliJ IDEA является его интеллектуальная поддержка, которая предлагает различные средства анализа кода, помогающие разработчикам создавать качественное программное обеспечение с меньшими ошибками. IDEA поддерживает различные плагины и расширения, что позволяет разработчикам интегрировать инструменты и языковые фреймворки, которые им необходимы для своего проекта. Это делает IntelliJ IDEA популярным выбором среди разработчиков Java и других языков, и она широко используется в индустрии разработки программного обеспечения.

Также для разработки используется фреймворк SpringBoot. Spring Boot - это фреймворк для создания приложений на языке Java. Он предоставляет удобные средства для быстрой разработки и запуска самодостаточных, упакованных по умолчанию приложений Spring. Spring Boot упрощает настройку и развертывание приложений, обеспечивая автоматическую конфигурацию большинства компонентов, что позволяет разработчикам фокусироваться на бизнес-логике приложения. Spring Boot также интегрируется с другими проектами Spring, такими как Spring Framework, Spring Data и Spring Security, обеспечивая широкие возможности для создания различных типов приложений, включая веб-приложения, микросервисы, приложения для обработки данных и другие.

Для решения проблемы миграции базы данных используется библиотека Flyway. Flyway - это инструмент для управления миграциями базы данных. Он позволяет автоматизировать и контролировать процесс изменения структуры базы данных в рамках разработки приложений. Flyway позволяет создавать и применять миграции базы данных, что обеспечивает управление версиями базы данных и автоматизацию процесса миграции при запуске приложения. Он также совместим с различными системами управления базами данных и может быть интегрирован с процессом непрерывной интеграции и поставки (CI/CD), что делает его полезным инструментом для разработчиков, работающих с базами данных в контексте разработки приложений.

**Сущности БД**

Сущность в базе данных является основным понятием, используемым для описания и хранения данных. Она представляет собой отдельный объект или предмет, о котором содержится информация и который имеет свои характеристики и атрибуты. В нашей базе данных присутствуют 2 сущности – UserEntity и AttestationEntity.

Сущность UserEntity представляет собой отображение объекта "пользователь" в базе данных в контексте Java Persistence API (JPA). Аннотация @Entity указывает, что класс является сущностью, которая будет отображаться в таблицу базы данных. Аннотация @Table (name = "user") указывает имя таблицы, к которой будет относиться данная сущность. Аннотация @Id указывает, что поле id является первичным ключом, а @GeneratedValue(strategy = GenerationType.IDENTITY) определяет способ последовательной генерации значений для этого ключа. Аннотация @OneToMany указывает на отношение "один ко многим" между сущностью UserEntity и сущностью AttestationEntity. Это означает, что одному пользователю может соответствовать несколько аттестаций. Геттеры и сеттеры используются для доступа к полям сущности из других частей приложения, а конструктор по умолчанию позволяет создавать объекты данной сущности без аргументов.

Поля класса:

* id: Первичный ключ сущности пользователя.
* login: Строковое поле, предназначенное для хранения логина пользователя.
* password: Строковое поле, используемое для хранения пароля пользователя.
* FIO: Строковое поле, представляющее ФИО (Фамилия, Имя, Отчество) пользователя.
* attestations: Список аттестаций, связанных с данным пользователем. Отношение "один ко многим" (One-to-Many) между пользователем и его аттестациями.

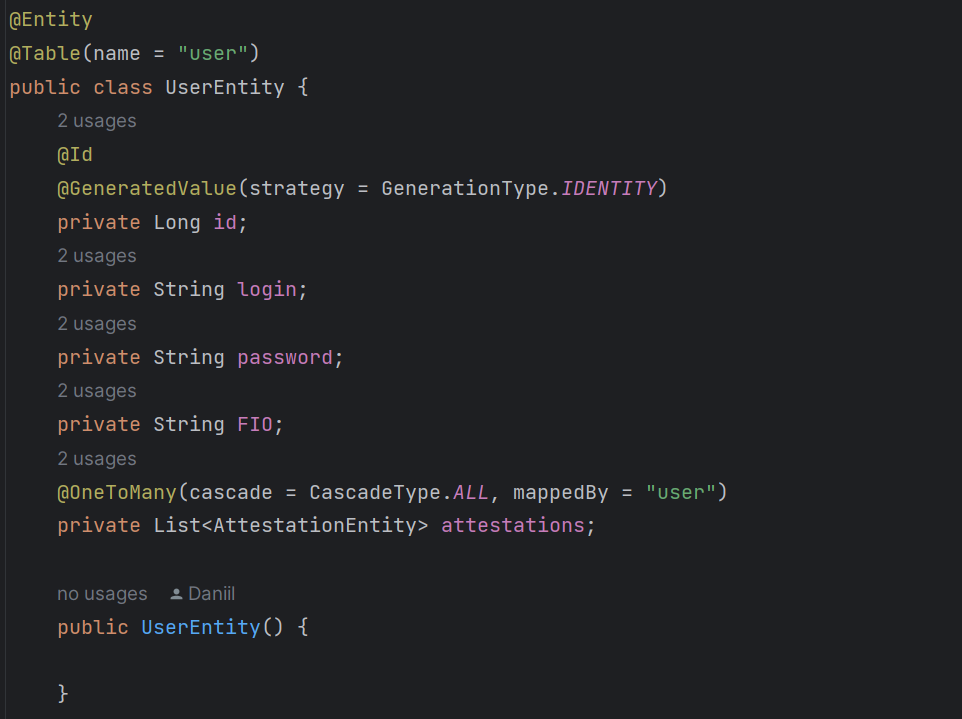


Рис.12 «Сущность UserEntity»

Сущность AttestationEntity представляет объект "аттестация" в базе данных в контексте Java Persistence API (JPA). Аннотация @Entity указывает, что класс является сущностью, которая будет отображаться в таблицу базы данных. Аннотация @Table (name = "attestation") указывает имя таблицы, к которой будет относиться данная сущность. Аннотация @Id указывает, что поле id является первичным ключом, а @GeneratedValue (strategy = GenerationType.IDENTITY) определяет способ генерации значений для этого ключа. Поле user является связью с сущностью UserEntity, которая представляет отношение "многие к одному" (Many-to-One) между AttestationEntity и UserEntity. Это означает, что нескольким аттестациям может соответствовать один пользователь. Геттеры и сеттеры используются для доступа к полям сущности из других частей приложения, а конструктор по умолчанию позволяет создавать объекты данной сущности без аргументов.

Поля класса:

* id: Поле id представляет собой первичный ключ сущности аттестации. Аннотация @Id указывает, что это поле является первичным ключом, а @GeneratedValue(strategy = GenerationType.IDENTITY) определяет способ генерации значений для этого ключа. В данном случае, используется генерация значений с помощью автоинкремента.
* title: Поле title представляет название аттестации. Оно отображается в столбце "title" в таблице базы данных.
* isValid: Поле isValid представляет состояние аттестации (действительна или нет). Тип данных Boolean указывает на возможные значения true или false.
* dataPassed: Поле dataPassed представляет дату прохождения аттестации. Оно отображается в соответствующем столбце в таблице базы данных.
* user: Поле user представляет связь с сущностью UserEntity и определяет отношение "многие к одному" (Many-to-One) между AttestationEntity и UserEntity. Оно связывается с полем "id" в сущности UserEntity посредством аннотации @JoinColumn(name = "user\_id").

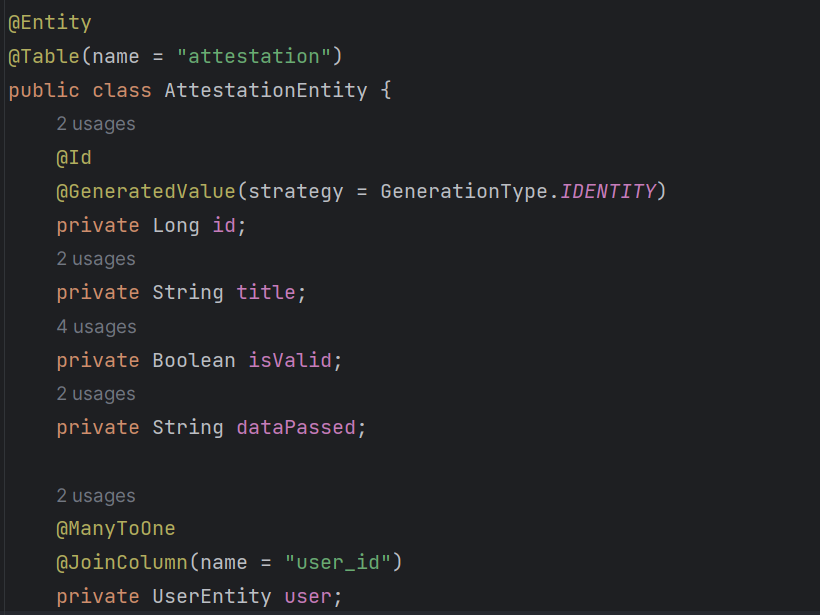


Рис.13 «Сущность AttestationEntity»

Для управления базой данных написаны методы REST API. В классе UserController описаны методы registration для добавления пользователя, getOneUser для получения пользователя по его id номеру, а также deleteUser для удаления пользователя по его id номеру.



Рис.14 «Методы класса UserController»

Для разделения бизнес логики и реализации все необходимые операции реализованы в классе UserService



Рис.15 «Методы класса UserService»

В классе AttestationController описаны методы createAttestation для создания аттестации и привязки к пользователю по его id, completeAttestation для обновления статуса аттестации, а также deleteAttestation для удаления аттестации.

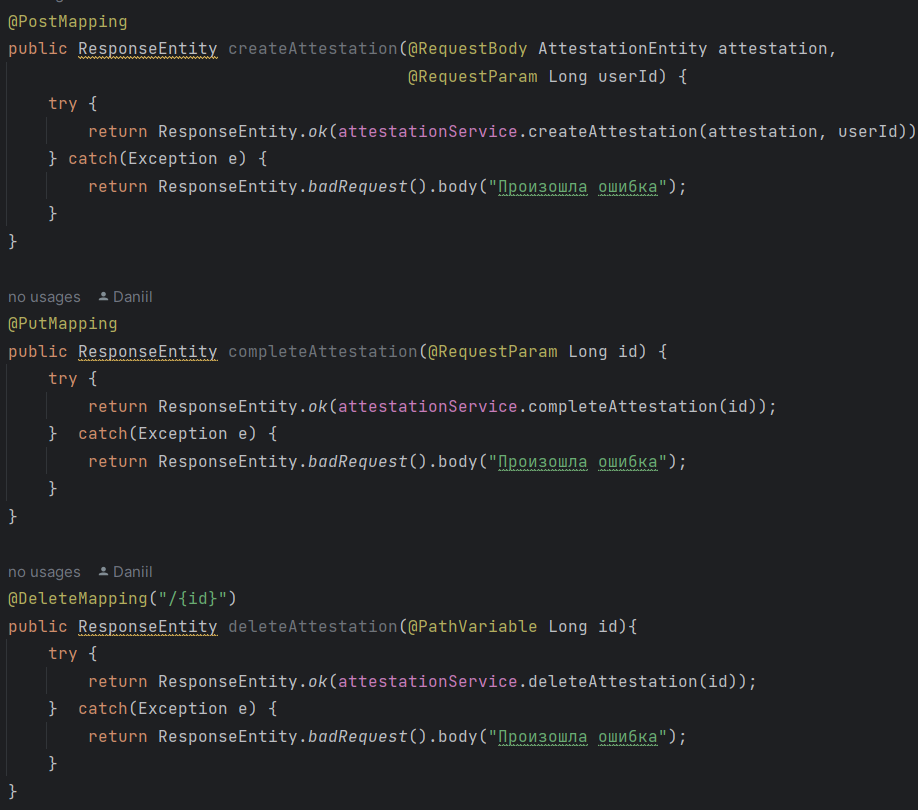


Рис.16 «Методы класса AttestationController»

Для разделения бизнес логики и реализации все необходимые операции реализованы в классе AttestationService

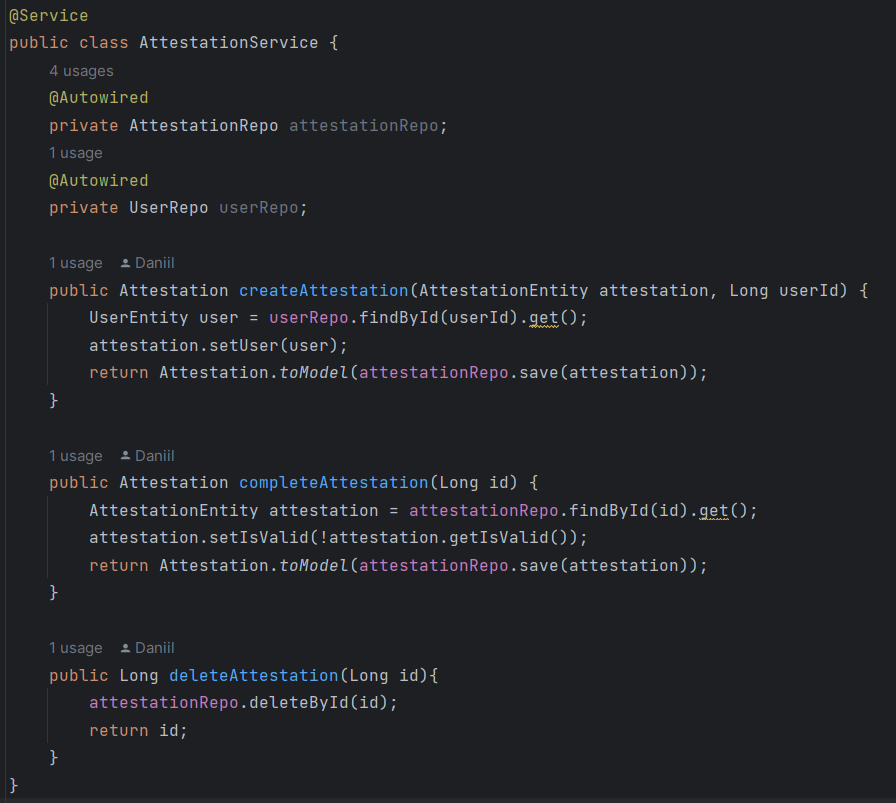


Рис.17 «Методы класса AttestationService»

Для решения проблемы миграции баз данных написан файл V1\_\_CREATE\_TABLES.sql

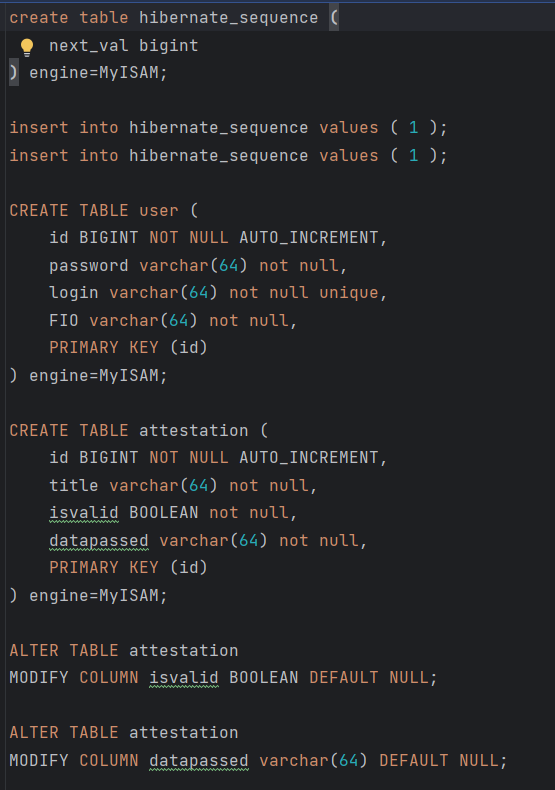


Рис.18 «Файл V1\_\_CREATE\_TABLES.sql»

**Тест-Кейсы**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Название | Требование | Валидные данные | Результат ожидаемый | Не валидные данные | Результат не ожидаемый |
| 1 | логин | От 4 символов, латинские символы, без цифр | Ivanov  Smirnov | Пройдено | Иванов  Смирнов228  «» | Выбрасывается ошибка: неверный логин или пароль |
| 2 | пароль | От 4 символов, латинские символы | kmuinmd | Пройдено | \_\_\_\_\_\_\_\_  ------------  «» | Выбрасывается ошибка: неверный логин или пароль |
| 3 | авторизация | Введен верный логин и пароль |  | переход на экран Mainscreen |  | Переход на другое окно / ничего не произошло |
| 4 | Запрос на вывод 1 пользователя | Осуществлен GETзапрос на localhost:8080/users?id=1 |  | status code: 200 и в теле запроса  {      "id": 1,      "login": "test",      "fio": "Иванов Иван Иванович"  } |  | Вернулась ошибка 400 Bad request или любая другая ошибка |
| 5 | Запрос на добавление пользователя | Осуществлен POST запрос на localhost:8080/users | {      "login": "test2",      "password": "test2",      "fio": "Петров Иван Иванович"  } | status code: 200 и в теле запроса «Пользователь сохранен» |  | Вернулась ошибка 400 Bad request или любая другая ошибка |
| 6 | Запрос на удаление пользователя по id | Осуществлен запрос DELETE localhost:8080/users/2 для удаления пользователя с id = 2 |  | status code: 200 и id удаленного пользователя (2) |  | Вернулась ошибка 400 Bad request или любая другая ошибка |
| 7 | Запрос на создание аттестации | Осуществлен запрос POST localhost:8080/attestations?userId=3 | {      "title": "Удостоверение частного охранника",      "dataPassed": "22.01.2023",      "isValid": **true**  } | status code: 200 и в теле запроса  {      "id": 7,      "title": "Удостоверение частного охранника",      "isValid": **true**,      "dataPassed": "22.01.2023"  } |  | Вернулась ошибка 400 Bad request или любая другая ошибка |
| 8 | Запрос на обновление аттестации | Осуществлен запрос PUT localhost:8080/attestations?id=8 |  | status code: 200 и в теле запроса  {      "id": 8,      "title": "Удостоверение частного охранника",      "isValid": **false**,      "dataPassed": "22.01.2023"  } |  | Вернулась ошибка 400 Bad request или любая другая ошибка |
| 9 | Запрос на удаление аттестации | Осуществлен запрос DELETE localhost:8080/attestations/7 |  | status code: 200 и в теле запроса  7 – id удаленной аттестации |  | Вернулась ошибка 400 Bad request или любая другая ошибка |

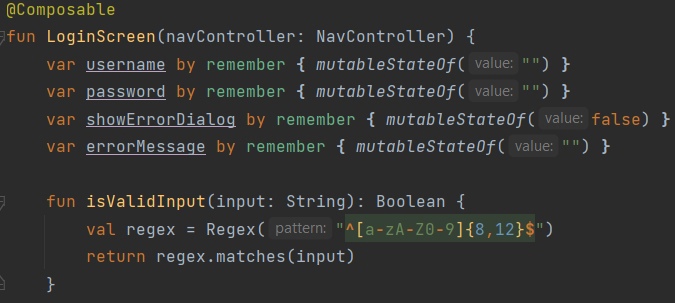


Рис.19 «Тест-кейс 1»

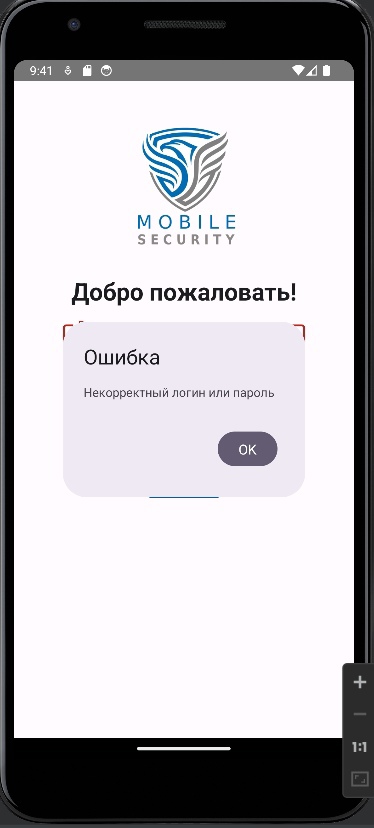


Рис.20 «Тест-кейс 2»

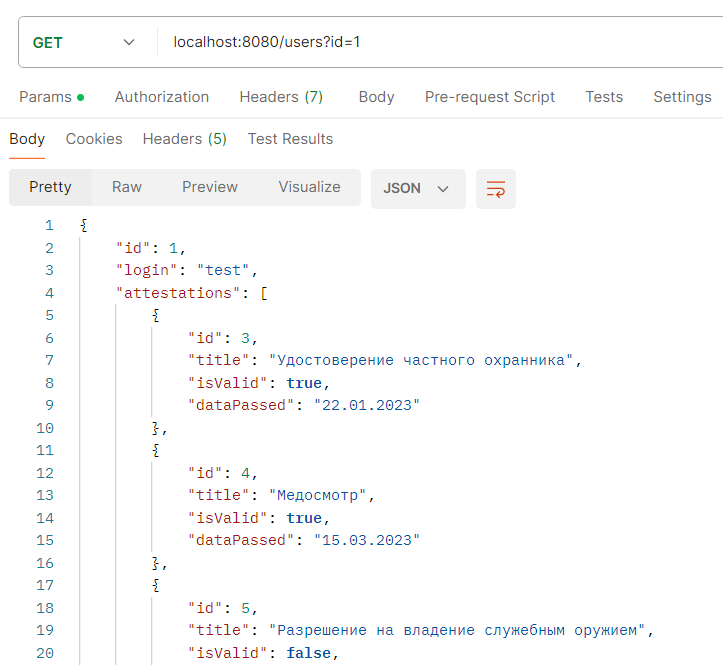


Рис.21 «Тест-кейс 4»

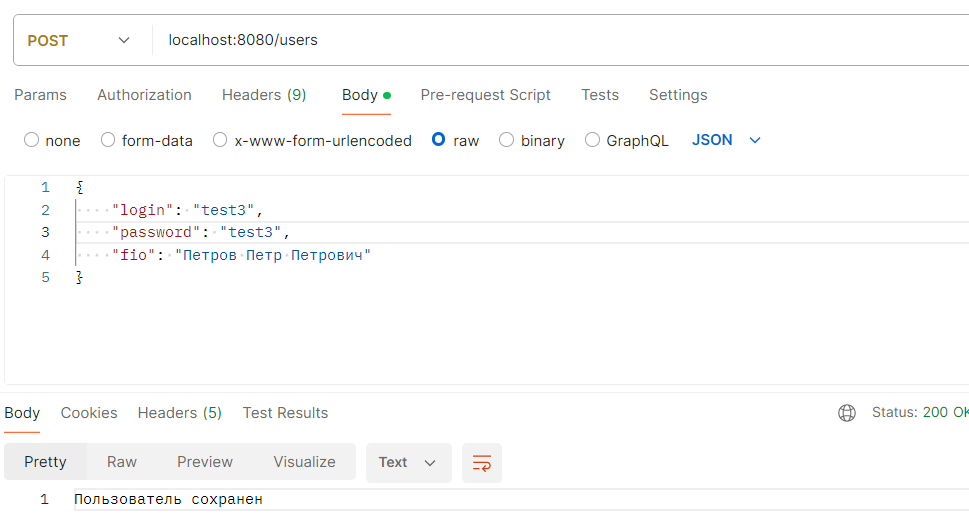


Рис.22 «Тест-кейс 5»

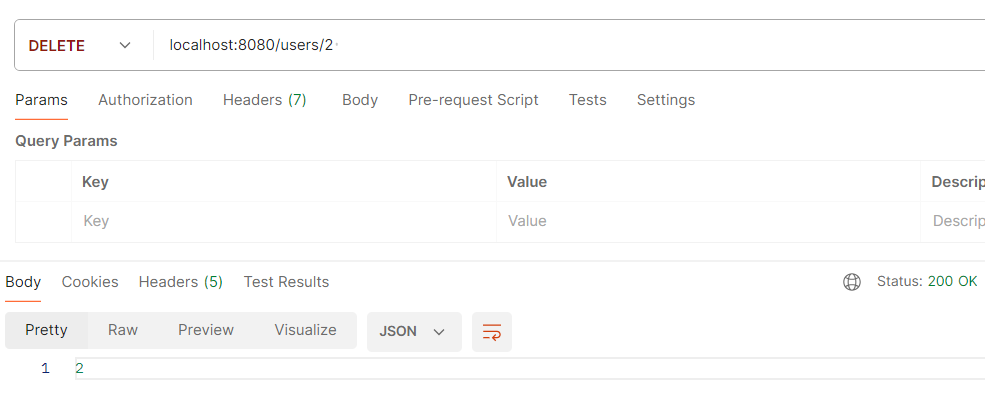


Рис.23 «Тест-кейс 6»

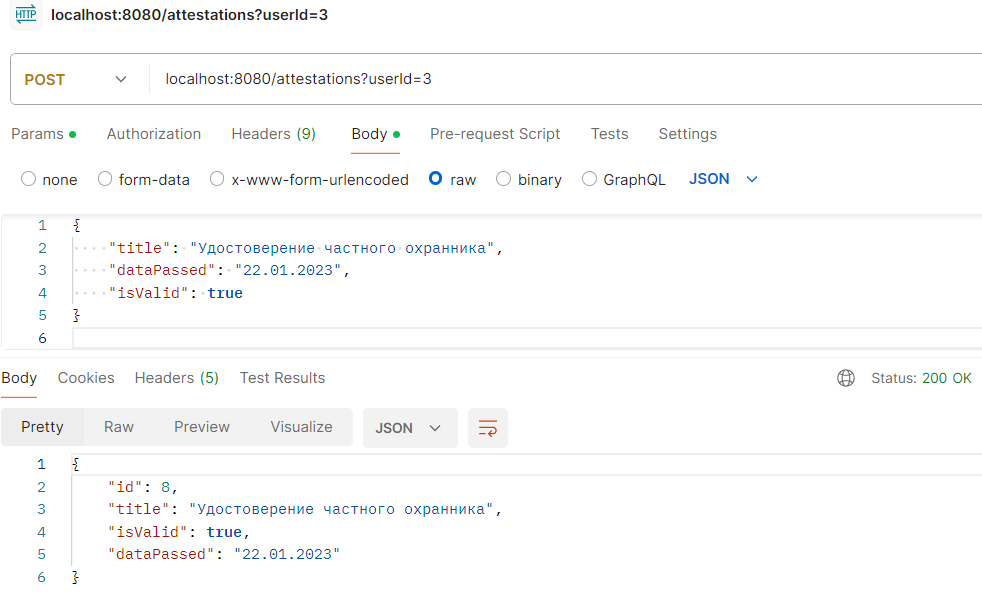


Рис.24 «Тест-кейс 7»

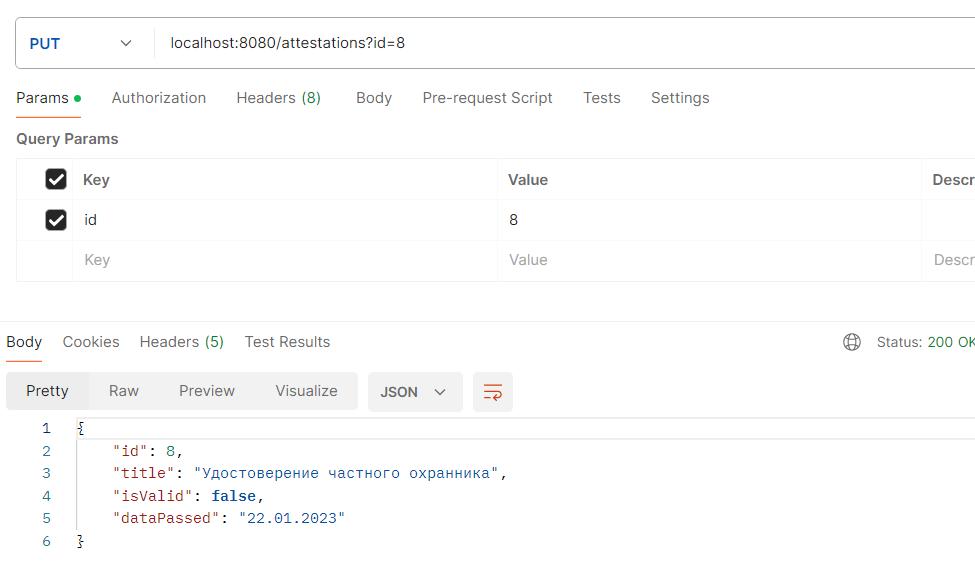


Рис.25 «Тест-кейс 8»

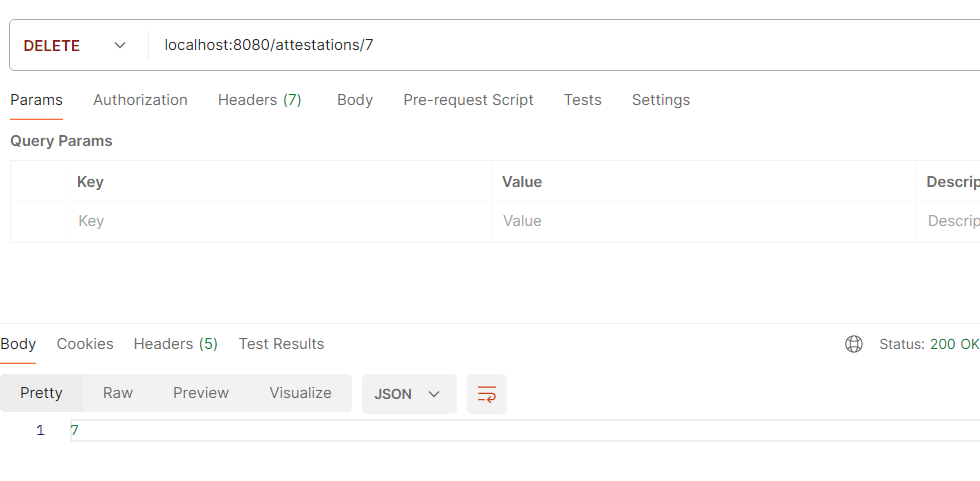


Рис.26 «Тест-кейс 9»

# **ЗАКЛЮЧЕНИЕ**

В ходе данной работы были изучены и проанализированы проблематика и предметная область. Подготовительный этап исследования и анализа требований позволил выявить ключевые потребности и ожидания от мобильного приложения, что послужило основой для его успешной разработки. Была исследована текущая ситуация, были описаны требования к разрабатываемому продукту и была разработана информационная система автоматизации дежурной части ЧОП. Также была написана техническая документация по разрабатываемой системе. Полученные данные и обратная связь от охранников и руководства охранной фирмы были использованы для создания функционального и удобного инструмента, значительно упрощающего процесс отчетности и контроля в рамках охранной деятельности.