2 Проектирование системы

2.1 Анализ основных требований

Конечное мобильное приложение направлено на управление мероприятиями и регистрацией студентов на них. Для реализации проекта необходимо разработать программную логику, которая будет обрабатывать запросы пользователей и ответственных за мероприятия и осуществить настройку интеграции с разрабатываемой информационной системой «Рейтинг студентов».

Функциональность системы должна позволять пользователям просматривать информацию о доступных мероприятиях, включая дату, время, место проведения и описание мероприятия. Регистрация студентов на мероприятия должна быть простой и удобной, а каждый зарегистрированный студент должен получить уникальный QR-код, который будет использоваться для подтверждения его фактического присутствия на мероприятии.

Для подтверждения фактического присутствия студента на мероприятии, необходимо, чтобы студент предъявил свой QR-код на входе на мероприятие, и система автоматически подтвердила его присутствие после сканирования кода ответственным за мероприятие.

Ответственные за мероприятия должны иметь возможность просмотра статистической информации о присутствующих на мероприятии студентах, включая количество зарегистрированных студентов и количество студентов, которые действительно присутствовали на мероприятии.

Для успешной реализации проекта необходимо также разработать удобный интерфейс для пользователей системы, который будет удобен и легок в использовании.

Для решения данного спектра задач необходимо определиться с необходимыми технологиями и средствами реализации проектирования.

2.2 Обоснование использования программных средств разработки

Выбор оптимальных инструментов и технологий может существенно снизить время разработки, повысить производительность и обеспечить высокое качество конечного продукта. Правильное решение на этапе выбора технологий – ключевой фактор успешной разработки мобильного приложения для Android.

**Kotlin** – это статически типизированный язык программирования, разработанный компанией JetBrains, работающий на платформе Java Virtual Machine (JVM) и являющийся официальным языком программирования для разработки Android-приложений, как и Java. Данный язык программирования является полностью совместимым с Java и может быть без проблем интегрирован в существующие проекты на Java. Это позволяет разработчикам постепенно переходить на Kotlin, сохраняя совместимость с существующим Java-кодом. Kotlin поддерживает функциональное программирование, что позволяет использовать лямбда-выражения, высокоуровневые функции, расширения функций и другие функциональные конструкции, тем самым упрощая разработку сложных функций и обработку данных. Kotlin поддерживает функциональное программирование, что позволяет использовать лямбда-выражения, высокоуровневые функции, расширения функций и другие функциональные конструкции. Это упрощает разработку сложных функций и обработку данных [6].

**Material Design** – это дизайн-язык, разработанный компанией Google, который используется для создания современных и стильных пользовательских интерфейсов мобильных приложений, включая Android-приложения. Material Design предлагает набор готовых компонентов и рекомендаций по оформлению интерфейса, которые обеспечивают единый и согласованный внешний вид и поведение приложений на платформе Android. Material Design предлагает набор компонентов, таких как кнопки, поля ввода, списки, диалоговые окна и другие, выполненных в виде материальных объектов, которые имитируют физические объекты в трехмерном пространстве. Для разработки мобильных приложений на платформе Android с использованием Material Design, Google предоставляет библиотеку под названием «Material Components for Android», которая включает набор готовых компонентов и стилей, соответствующих принципам Material Design. Эта библиотека интегрируется в Android Studio и позволяет разработчикам легко создавать современные и стильные интерфейсы для своих приложений, соответствующие дизайн-языку Material Design [7].

**Volley** – это библиотека для обработки сетевых запросов в мобильных приложениях на платформе Android, разработанная компанией Google. Она предоставляет удобные и эффективные средства для работы с сетью, такие как отправка HTTP-запросов, обработка ответов, кэширование, управление очередью запросов и другие функции, что делает ее популярным инструментом для разработки мобильных приложений.

Volley также предлагает ряд дополнительных возможностей, таких как поддержка отмены запросов, обработка ошибок, автоматическая обработка cookie, поддержка множественных соединений и другие, что делает ее мощным инструментом для работы с сетью в мобильных приложениях на платформе Android [8].

**Draw.io** — это веб-приложение, предоставляющее возможность создавать диаграммы и схемы различных типов. Он широко используется для создания блок-схем, потоковых диаграмм, организационных диаграмм, сетевых диаграмм, диаграмм классов, ER-диаграмм и многих других типов диаграмм [9].

Draw.io предлагает простой в использовании интерфейс с интуитивно понятными инструментами рисования и множеством готовых элементов, таких как блоки, стрелки, символы и формы, которые можно использовать для создания диаграммы. Пользователи могут создавать диаграммы "перетаскиванием и редактированием", добавлять текст, изменять цвета, размеры и стили элементов, а также настраивать соединения между элементами.

**Ramus** — это программное обеспечение для создания диаграмм требований и моделирования бизнес-процессов [10].

Ramus предлагает графический интерфейс, который позволяет создавать различные виды диаграмм, такие как диаграммы потоков данных (DFD), диаграммы IDEF0, диаграммы требований и другие. Он также предоставляет возможность создания таблиц, описывающих требования, атрибуты и другую дополнительную информацию о моделируемых системах.

Одна из основных особенностей Ramus – это поддержка структурного подхода к моделированию, такого как функциональное моделирование и моделирование потоков данных. Он также предлагает возможность анализировать и описывать требования системы, управлять версиями моделей, создавать отчеты и экспортировать модели в различные форматы файлов.

**GitHub** — это веб-платформа, предназначенная для хостинга и управления репозиториями (хранилищами) кода на базе системы контроля версий Git. GitHub предлагает удобные инструменты для разработчиков, которые позволяют им работать с кодом, вносить изменения, отслеживать историю изменений, сотрудничать с другими разработчиками, проводить рецензирование кода, управлять проектами и многое другое [11].

Одна из основных особенностей GitHub — это возможность совместной работы над проектами. Разработчики могут вносить изменения в репозиторий, создавать ветки и отправлять запросы на слияние (Pull Requests), что позволяет обсуждать изменения, проводить рецензирование кода и принимать решение о включении изменений в основную ветку проекта. Это способствует коллаборации и обмену знаниями между разработчиками.

В целом, GitHub является мощным инструментом для хостинга и управления репозиториями кода, с широким набором функций для совместной разработки, автоматизации процессов разработки и интеграции с другими инструментами разработки. Он широко используется разработчиками по всему миру и является важным инструментом в современной разработке программного обеспечения.

**Android Studio** – интегрированная среда разработки, специально разработанная для создания мобильных приложений на платформе Android. Она предлагает широкий набор функций, таких как автодополнение кода, рефакторинг, отладка, профилирование, визуальный макет редактор и другие инструменты. Android Studio поддерживает языки программирования Java, Kotlin и C++, а также различные фреймворки и библиотеки. Она также обладает богатыми возможностями тестирования и настраиваемым интерфейсом, что делает ее мощным инструментом для разработки высококачественных мобильных приложений на Android [12].

2.3 Архитектура подсистемы

2.3.1 Общая архитектура системы

Общая архитектура системы представляет собой набор из двух компонентов: клиентским мобильным приложением и сервером. Обмен данными осуществляется в двустороннем формате. Серверное приложение обрабатывает запросы, взаимодействует с базой данным и отправляет результат в виде JSON-файла. Клиент обращается к серверу и получает данные, которые использует для заполнения соответствующих полей графического интерфейса.

Архитектура системы представлена на рисунке 2.1.

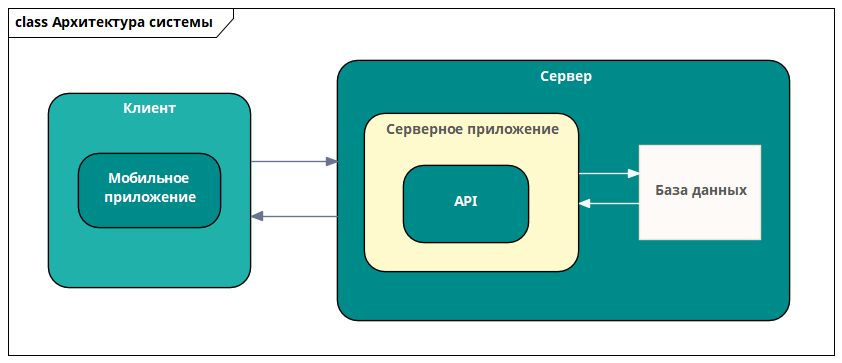


Рисунок 2.1 – Архитектура системы

В качестве данных выступают: список проводимых мероприятий с открытой регистрацией, информация о необходимом мероприятии, перечень участников мероприятия, информация об авторизованном пользователе.

2.3.2 Архитектура мобильного приложения

В мобильном приложении используется многослойная архитектура, которая состоит из четырех основных частей: UI Layer, Adapter Layer, Data Layer и Helper Layer.

UI Layer - первый слой, который отвечает за обработку пользовательского взаимодействия с графическим интерфейсом. В этом слое содержится набор Activity, включая EventActivity, SettingsActivity, EventInfoActivity, ScannerActivity, ResultScannerActivity, QrActivity, MyEventActivity, EventStatsActivity, NetworkErrorActivity и MainActivity.

1. EventActivity – Activity для демонстрации пользователю информации о выбранном им мероприятии;
2. SettingsActivity – Activity настроек мобильного приложения;
3. EventInfoActivity – вывод полной информации о мероприятии для ответственного;
4. ScannerActivity – Activity сканера QR-кодов мероприятия;
5. ResultScannerActivity – вывод результата сканирования QR-кода;
6. QrActivity – демонстрация увеличенного варианта QR-кода участника мероприятия;
7. MyEventActivity – вывод информации о мероприятии с пройденном регистрацией;
8. MainActivity – основное окно приложения;
9. EventStatsActivity – Activity для отображения статистики мероприятия;
10. NetworkErrorActivity – Activity отображения информации об ошибке приложения в случае отсутствия интернет-соединения.

В классе MainActivity находится меню управления фрагментами, а также объект для отображения фрагментов. Фрагменты — это маленькие кусочки пользовательского интерфейса, которые могут быть заменены друг на друга в рамках одной активности.

Список фрагментов в приложении включает в себя:

1. EventsFragment – фрагмент, содержащий карточки мероприятий, отображаемые в виде списка;
2. PresenceFragment – фрагмент, отображающий список мероприятий, доступных для подтверждения присутствия участников;
3. MyEventsFragment – фрагмент, отображающий список мероприятий, на которые пользователь зарегистрировался для участия;
4. EventsResponsibleFragment – фрагмент, содержащий карточки мероприятий для ответственного за мероприятия;
5. ProfileCleanFragment – фрагмент профиля без авторизации. Он позволяет пользователям авторизоваться в системе;
6. ProfileFragment – фрагмент профиля с авторизацией, отображает информацию об авторизованном пользователе;
7. ModerationFragment – фрагмент, отображающий список мероприятий, доступных для модерации.

Слой адаптеров (Adapter Layer) представлен тремя классами-адаптерами, необходимые для правильного взаимодействия с RecyclerView:

1. EventAdapter – контроллер карточек мероприятий;
2. MyEventAdapter – контроллер списка мероприятий, на которые зарегистрировался пользователь;
3. EventResponsibleAdapter – контроллер списка карточек мероприятий для ответственных за мероприятия.

Слой данных (Data Layer) состоит из четырех классов, каждый из которых отвечает за хранение определенных данных:

1. Participant – содержит данные для таблицы статистики мероприятий;
2. MyEvent – содержит данные о мероприятии, с пройденной регистрацией;
3. Event – содержит данные о мероприятиях;
4. User – содержит данные об авторизованном пользователе.

Слой Helper Layer включает в себя классы, необходимые для правильной работы мобильного приложения:

1. LanguageConfig – позволяет менять язык мобильного приложения;
2. MyContextWrapper – наследует и переопределяет стандартный класс ContextWrapper, чтобы делегировать его вызовы другому контексту;
3. ServerHelper – отвечает за связь приложения с API информационной системы «Рейтинг студентов»;
4. SharedPrefs – позволяет работать с чтением и записью необходимой информации в SharedPreferences.

Многослойная архитектура мобильного приложения позволяет более гибко управлять его компонентами и облегчает поддержку приложения в будущем. Каждый слой выполняет свою функцию и обеспечивает правильное взаимодействие компонентов приложения.

Архитектура мобильного приложения представлена на рисунке 2.2.

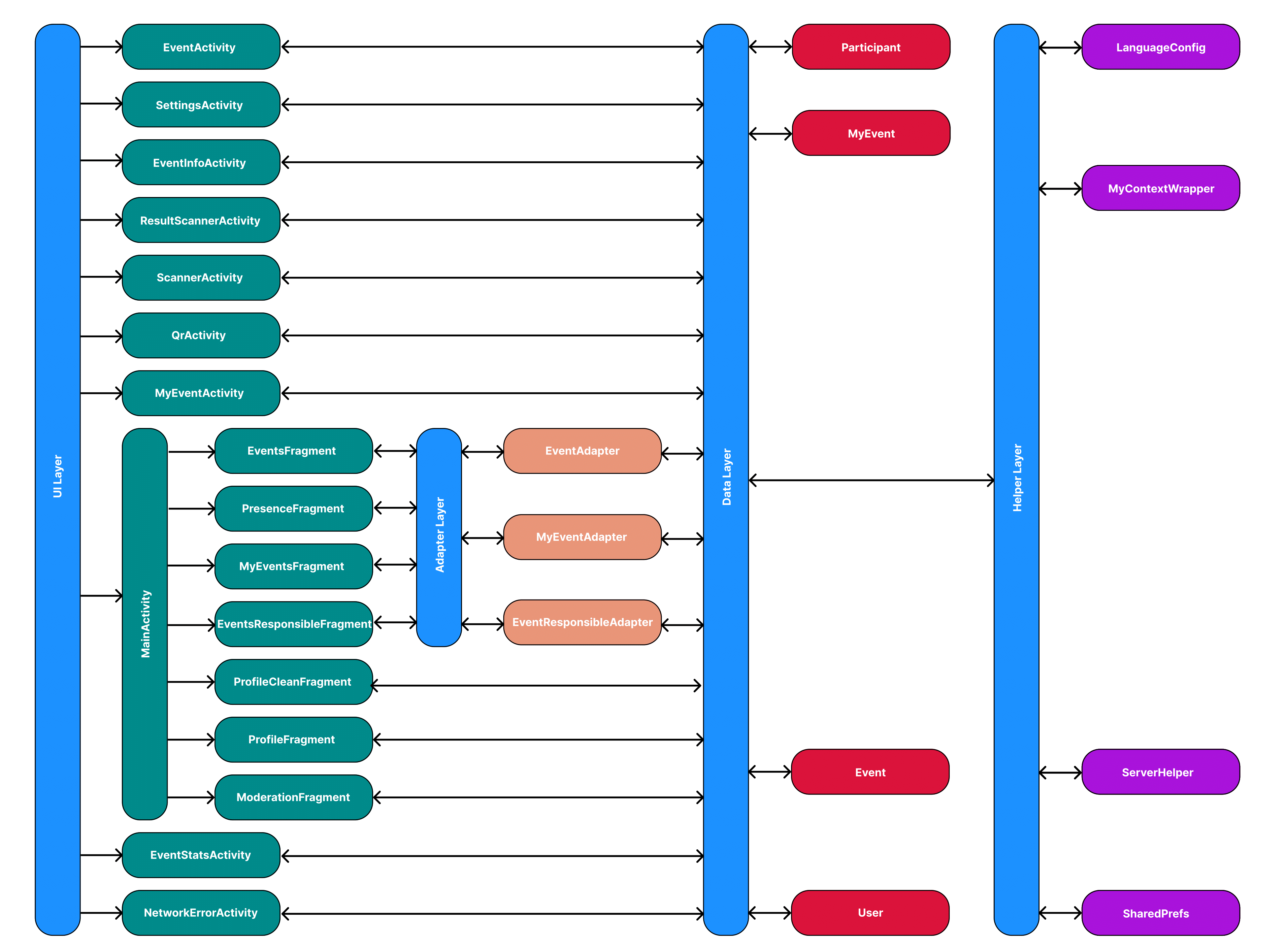


Рисунок 2.2 – Архитектура мобильного приложения

2.4 Проектирование классов данных

Дата классы описывает систему управления мероприятиями и участниками в мобильном приложении.

Сущность Event представляет собой конкретное мероприятие, описывая его id, название, описание, изображения, теги, дату и место проведения, количество зарегистрированных участников, даты начала и окончания регистрации, а также организаторов.

Сущность MyEvent хранит данные о мероприятии, на которое зарегистрировался пользователь, включая id регистрации и id мероприятия, название мероприятия, теги, дату и место проведения, uid пользователя для подтверждения присутствия.

Сущность User описывает авторизованного пользователя, включая его id, ФИО, роль в системе, учебную группу и изображение.

Сущность Participant представляет собой таблицу статистики мероприятия, описывая присутствие зарегистрированных на мероприятии пользователей, их ФИО и учебную группу, фактическое присутствие, роль на мероприятии и подтверждение присутствия ответственным за мероприятие.

На рисунке 2.3 представлена схема классов данных, в таблице 2.1 представлено описание полей.

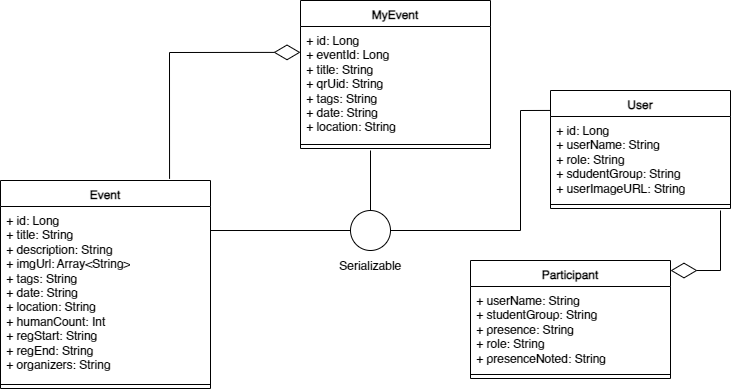


Рисунок 2.3 – Классы данных приложения

Таблица 2.1 – Описание полей Data-классов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Уровень доступа | Тип данных | Свойство | Назначение |
| Event | | | | |
| id | public | Long | val | id мероприятия в системе |
| title | public | String | val | Название мероприятия |

Продолжение таблицы 2.1 – Описание полей Data-классов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| description | public | String | val | Описание мероприятия |
| imgUrl | public | Array<String> | val | Набор ссылок на изображения |
| tags | public | String | val | Набор тегов мероприятия |
| date | public | String | val | Дата и время проведения мероприятия |
| location | public | String | val | Место проведения мероприятия |
| humanCount | public | Int | val | Количество зарегистрированных участников |
| regStart | public | String | val | Дата и время начала регистрации |
| regEnd | public | String | val | Дата и время конца регистрации |
| organizers | public | String | val | Список организаторов мероприятия |
| MyEvent | | | | |
| id | public | Long | val | id зарегистрированного мероприятия |
| eventId | public | Long | val | id мероприятия в системе |
| title | public | String | val | Название мероприятия |
| qrUid | public | String | val | Uid регистрации |
| tags | public | String | val | Набор тегов мероприятия |
| date | public | String | val | Дата и время проведения мероприятия |
| location | public | String | val | Место проведения мероприятия |
| User | | | | |
| id | public | Long | val | id пользователя в системе |
| userName | public | String | val | ФИО пользователя |
| userImageURL | public | String | val | Ссылка на изображение пользователя |
| studentGroup | public | String | val | Учебная группа |

Продолжение таблицы 2.1 – Описание полей Data-классов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| role | public | String | val | Роль в системе |
| Participant | | | | |
| eventId | public | Long | val | id мероприятия в системе |
| userName | public | String | val | ФИО пользователя |
| studentGroup | public | String | val | Учебная группа |
| presence | public | String | val | Фактическое присутствие |
| role | public | String | val | Роль в системе |
| presenceNoted | public | String | val | Подтвердивший ответственный |

2.5 Проектирование графического интерфейса

Для реализации графического интерфейса спроектирован набор экранов.

На рисунке 2.4. представлена диаграмма интерфейсных классов для роли «Студент». Диаграмма состоит из восьми экранов и шести диалоговых окон.

На рисунке 2.5 представлена диаграмма интерфейсных классов для роли «Ответственный за мероприятие». Диаграмма состоит из одиннадцати экранов и пяти диалоговых окон.

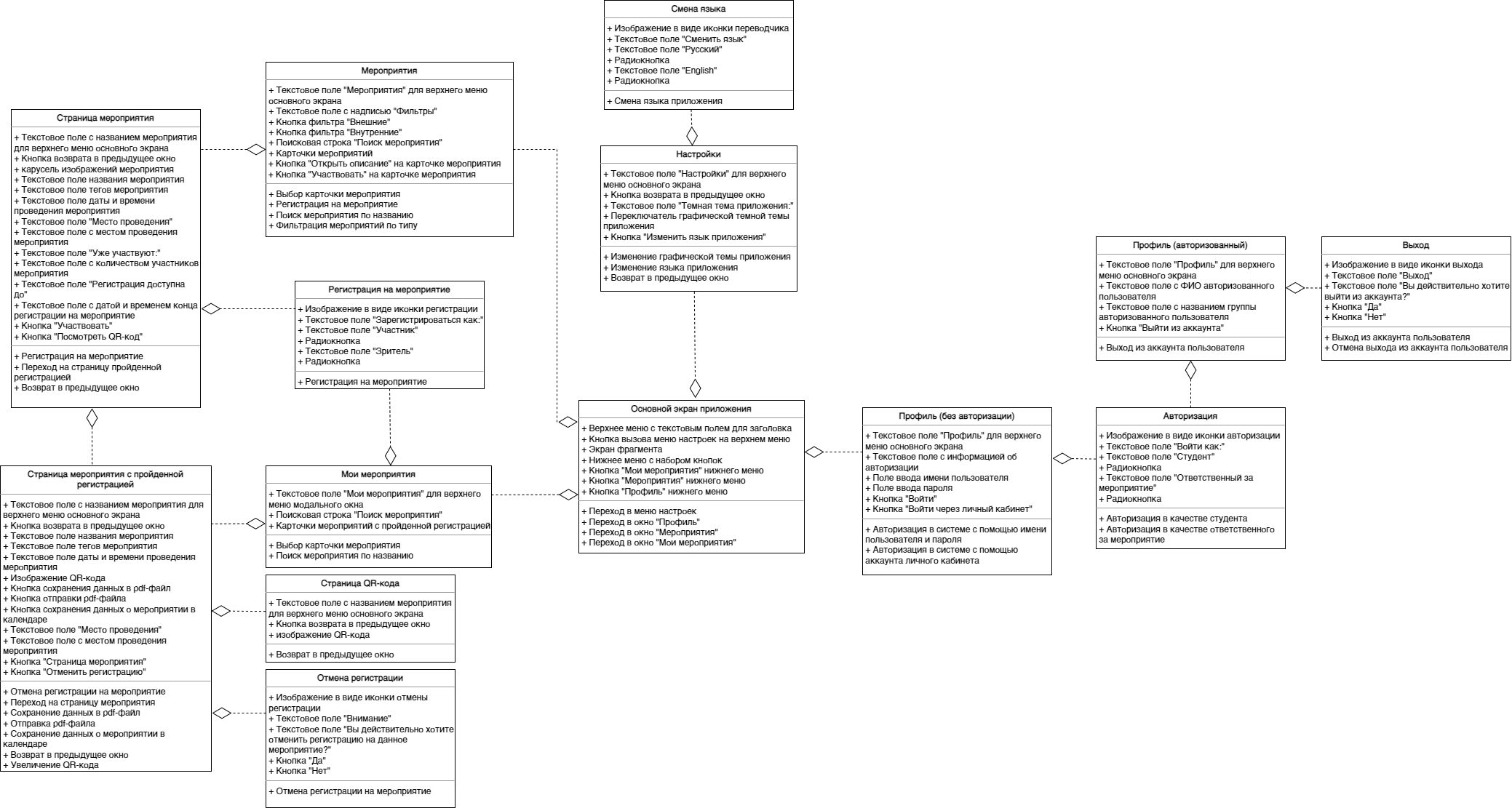


Рисунок 2.4 – Диаграмма интерфейсных классов для роли «Студент»

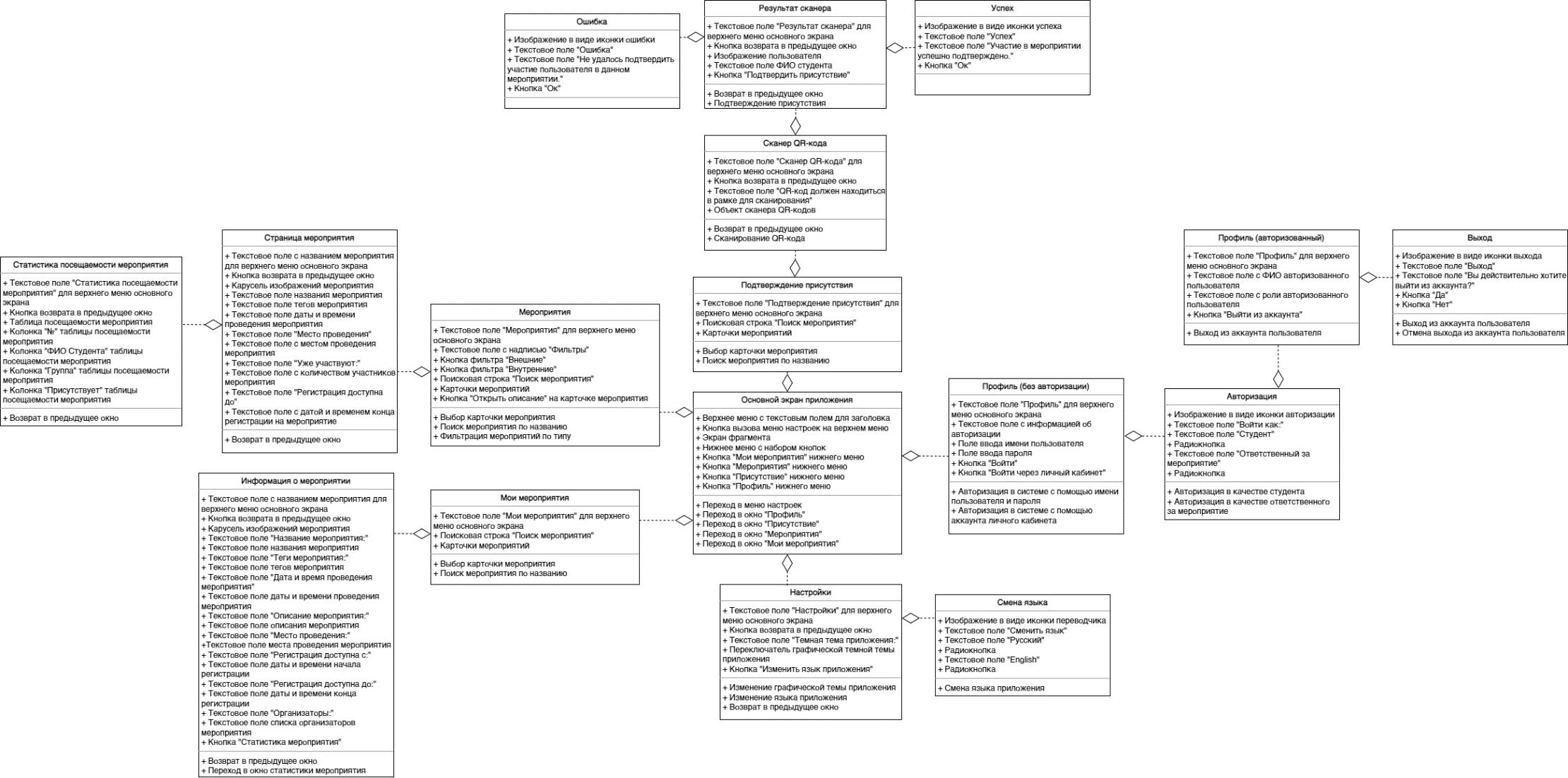


Рисунок 2.5 – Диаграмма интерфейсных классов для роли «Ответственный за мероприятие»

2.5.1 Проектирование графического интерфейса для роли «Студент»

**2.5.1.1 Описание главного экрана приложения**

При запуске приложения пользователь должен попадать на главный экран приложения с автоматически загруженным фрагментом «Мероприятия». Экран состоит из следующего набора элементов:

1. **Элемент 1** – верхнее меню приложения. Данный элемент состоит из поля заголовка названия фрагмента и кнопки вызова контекстного меню настроек (**элемент 4**).
2. **Элемент 2** –экран фрагмента. Фрагмент находится внутри основного контейнера экранной формы и расширяет его другими фреймами. При необходимости данный элемент может быть пролистан жестом перемещения пальца снизу вверх. Данная возможность позволяет размещать внутри контейнера фрагмента неограниченное количество карточек мероприятий (**элемент 7**);
3. **Элемент 3** – нижнее меню навигации. Данный компонент отвечает за динамическую загрузку необходимого фрагмента. Выбор необходимого осуществляется за счет трех кнопок, содержащих соответствующую иконку и подпись вызываемого окна;
4. **Элемент 5** – панель фильтрации мероприятий. Данная панель включает три кнопки-варианта применяемых фильтров. Одновременно применен может быть только один фильтр. Для отображения всех возможных мероприятий необходимо активировать кнопку «Любые»;
5. **Элемент 6** – строка поиска мероприятия. Данный компонент отвечает за поиск определенного мероприятия по критерию «Название мероприятия».
6. **Элемент 7** – карточка мероприятия. Данный элемент состоит из изображения мероприятия (устанавливается нулевой объект из набора графических изображений мероприятия), текстового поля для названия мероприятия, тегов мероприятия, даты и времени проведения мероприятия в формате «dd.mm.yyyy hh:mm», текстового поля для отображения фрагмента описания мероприятия и кнопок «Открыть описание» и «Участвовать» (**элементы 8 и 9 соответственно**);
7. **Элемент 8** – кнопка «Открыть описание». При нажатии на данную кнопку пользователь будет перенаправлен на экран описания мероприятия.
8. **Элемент 9** – кнопка «Участвовать». При нажатии на данную кнопку мобильное приложение отобразит диалоговое окно выбора варианта регистрации на мероприятие в качестве участника или зрителя. Данная кнопка находится в неактивном состоянии в двух случаях: при не пройдённой авторизации в систему и при пройденной регистрации на мероприятие.

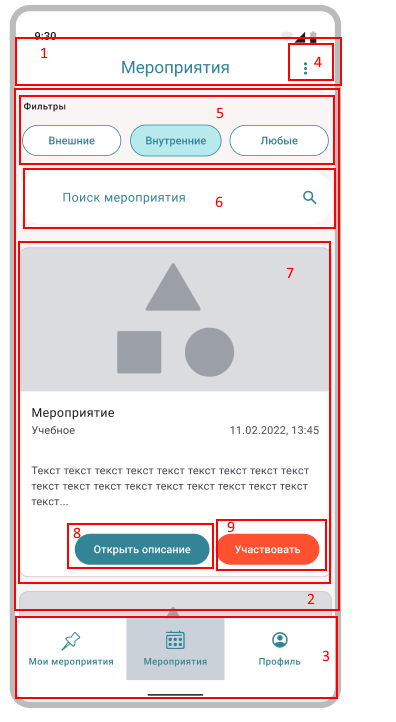


Рисунок 2.6 – Макет основного экрана приложения

**2.5.1.2 Описание экрана «Мои мероприятия»**

