**Анализ предметной области**

Театр − это организация, в которой осуществляется художественная деятельность посредством постановки и исполнения спектаклей. В рамках информационной системы театр выступает как центральная среда, объединяющая всех участников процесса: артистов, администраторов и зрителей. Он определяет репертуар, расписания, кадровую политику и обеспечивает функционирование всех связанных с ним процессов, включая распределение ролей и управление персоналом.

Сущности предметной области:

* актёр;
* управляющий;
* спектакль;
* роль;
* назначение;
* заявка.

Актёр − творческий работник театра, исполняющий роли в спектаклях. В системе актёр является пользователем с ограниченными правами: может входить в личный кабинет, просматривать свои назначения и распределение по спектаклям, а также подавать заявки на участие в новых постановках. Доступ к редактированию данных других пользователей и назначений отсутствует.

Управляющий − администратор системы, отвечающий за кадровое распределение и управление учётными записями актёров. Он имеет полный доступ к данным: создаёт актёров, назначает их на роли, просматривает состав всех спектаклей и принимает или отклоняет заявки. Управляющий обеспечивает порядок и контроль в распределении ролей.

Спектакль − постановка, входящая в репертуар театра. Является основной единицей творческой деятельности и служит контекстом для распределения актёров. Каждый спектакль включает несколько ролей, имеет автора, жанр, режиссёра и статус, определяющий его участие в текущем репертуаре.

Роль − персонаж, который исполняется актёром в рамках одного спектакля. Роли различаются по типу (главная, второстепенная), полу, возрасту и другим характеристикам. В системе роли являются объектами, на которые осуществляется назначение или подача заявок.

Назначение − связь между актёром и ролью в конкретном спектакле. Фиксирует факт участия актёра в постановке, определяет его статус (основной исполнитель или дублёр) и используется для формирования афиш, графиков репетиций и внутренней отчётности.

Заявка − инициатива актёра на участие в спектакле на определённую роль. Система позволяет подавать заявки на свободные или конкурируемые позиции, после чего управляющий рассматривает кандидатуру и принимает решение об одобрении или отклонении. Заявка обеспечивает гибкость и вовлечённость актёрского состава.

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**на создание автоматизированной системы**

**«Управление распределением актёров в театре»**

**1. Общие сведения**

Полное наименование системы:

Автоматизированная система управления распределением актёров в театре.

Заказчик системы:

Система выполнена в рамках учебной практики.

Разработчик системы:

Кабунина Ульяна Александровна

ГОСТ 34.602-89 «Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы»

Плановые сроки выполнения работ:

Начало работ: 03.12.2025

Окончание работ: 23.12.2025

Порядок оформления и предъявления результатов:

Результаты работ предъявляются в виде:

* + рабочей версии приложения (мобильная);
  + документации по эксплуатации;
  + отчёта о тестировании;
  + презентации функционала.

Все материалы предоставляются на электронных носителях и в печатном виде.

**2. Назначение и цели создания системы**

**2.1 Назначение системы**

Система предназначена для автоматизации процесса распределения актёров по спектаклям в театральных учреждениях. Объектом автоматизации является управленческая и кадровая деятельность в театре, включающая:

* + создание и ведение учётных записей актёров;
  + назначение актёров на роли в спектаклях;
  + подачу и обработку заявок актёров на участие;
  + просмотр распределения по спектаклям.

**2.2 Цели создания системы**

* + повысить прозрачность и эффективность распределения ролей;
  + сократить время на ручное управление кадрами;
  + обеспечить централизованное хранение данных об актёрах и спектаклях;
  + упростить взаимодействие между управляющим и актёрами;
  + сформировать основу для дальнейшего расширения функционала (графики репетиций, оповещения, аналитика).

**3. Характеристика объектов автоматизации**

Объектом автоматизации является театральное учреждение, осуществляющее деятельность по постановке и исполнению спектаклей.

Ключевые процессы, подлежащие автоматизации:

* + ведение кадрового состава актёров;
  + формирование репертуара;
  + распределение ролей;
  + управление заявками на участие;
  + просмотр текущего распределения.

Система будет использоваться в условиях офиса театра и удалённо актёрами через мобильное приложение.

**4. Требования к системе**

**4.1. Требования к структуре и функционированию**

Система должна обеспечивать:

* + двухуровневую модель доступа: управляющий и актёр;
  + возможность создания, редактирования и просмотра учётных записей;
  + назначение актёров на роли в спектаклях;
  + просмотр распределения по всем спектаклям (для всех пользователей);
  + подачу заявок актёрами на участие в спектаклях;
  + утверждение или отклонение заявок управляющим;
  + ведение журнала изменений (назначения, заявки, статусы).

**4.2. Требования к надёжности**

* + обеспечение сохранности данных при сбоях питания;
  + резервное копирование раз в сутки;
  + восстановление данных из резервной копии за не более чем 30 минут.

**4.3. Требования к безопасности**

* + авторизация по логину и паролю;
  + шифрование паролей (хеширование);
  + разделение прав доступа по ролям;
  + защита от несанкционированного доступа в соответствии с внутренними нормативами заказчика.

**4.4. Требования к эргономике и технической эстетике**

* + интуитивно понятный интерфейс;
  + поддержка русского языка;
  + адаптивный дизайн для мобильных устройств;
  + минимальное количество кликов для выполнения основных операций.

**4.5. Требования к эксплуатации и обслуживанию**

* + система должна работать без технического обслуживания в течение не менее 30 дней;
  + поддержка мобильных платформ: Android 8.0+.

**4.6. Требования к защите информации**

* + доступ к данным только авторизованным пользователям;
  + логирование всех действий управляющего (назначения, удаления, изменения);
  + оповещение о подозрительной активности (по будущему развитию).

**5. Состав и содержание работ по созданию системы**

1. Анализ предметной области и требований заказчика.
2. Проектирование структуры базы данных.
3. Разработка архитектуры системы (мобильное приложение).
4. Реализация функционала:
   * управление пользователями;
   * управление спектаклями и ролями;
   * назначение актёров;
   * подача и обработка заявок;
   * тестирование системы (модульное, интеграционное).
5. Разработка технической документации.

**6. Порядок контроля и приёмки системы**

Приёмка осуществляется преподавателем, ведущим учебную практику.

Критерии приёмки:

* + соответствие функционала требованиям ТЗ;
  + отсутствие критических ошибок;
  + успешное прохождение тестов;
  + подписание акта приёмки.

**7. Требования к подготовке объекта автоматизации**

* + создание учётных записей актёров — осуществляется управляющим вручную.

Подготовка технических условий:

* + доступ в интернет;
  + смартфон с поддержкой современных операционных систем.

**8. Требования к документированию**

Перечень разрабатываемых документов:

* + руководство пользователя (в печатном и электронном виде);
  + руководство администратора;
  + техническое описание системы;
  + акт приёмки;
  + гарантийное обязательство;
  + все документы предоставляются в формате PDF и на бумажном носителе.

Документация соответствует требованиям ГОСТ 34.201-89.

**9. Источники разработки**

ГОСТ 34.602-89 — основной нормативный документ для оформления ТЗ.

**СПЕЦИФИКАЦИЯ ТРЕБОВАНИЙ К АВТОМАТИЗИРОВАННОЙ СИСТЕМЕ «Управление распределением актёров в театре»**

**1. Введение**

**1.1. Назначение документа**

Настоящий документ – спецификация требований – формально описывает функциональные и нефункциональные требования к автоматизированной системе «Управление распределением актёров в театре».

Он определяет, что система должна делать, как она должна себя вести, какие ограничения и условия применимы к её работе.

Документ используется для проектирования, разработки, тестирования и приёмки системы.

**1.2. Объект системы**

Система предназначена для театров и драматических коллективов, где требуется централизованное управление актбёрским составом и распределением ролей по спектаклям.

**1.3. Цель системы**

Обеспечить эффективное, прозрачное и удобное управление кадровыми процессами: назначение актёров, подача заявок, просмотр распределения, ведение учётных записей.

**2. Общее описание системы**

**2.1. Перспектива системы**

Система представляет собой мобильное прилодение, работающее по принципу клиент-сервер. Данные хранятся на сервере, доступ осуществляется через мобильное приложение.

**2.2. Функции системы**

* + управление пользователями (актёры, управляющий) ;
  + управление спектаклями и ролями;
  + назначение актёров на роли;
  + подача и обработка заявок;
  + просмотр распределения (для всех пользователей);
  + авторизация и контроль доступа.

**2.3. Пользователи системы**

|  |  |
| --- | --- |
| Роль | Описание |
| Управляющий | Администратор, имеет полный доступ к управлению данными |
| Актёр | Пользователь с ограниченными правами: просмотр и подача заявок |

**2.4. Предположения и зависимости**

* + пользователи имеют доступ к интернету;
  + управляющий обладает базовыми навыками работы с ПК и смартфонами;
  + система не интегрируется с внешними кадровыми или бухгалтерскими системами (на текущем этапе);
  + все данные вводятся вручную или через импорт из Excel.

**3. Функциональные требования**

FR-01. Управление актёрами

Управляющий может:

* + создавать учётную запись актёра, указывая ФИО, контактные данные логин и пароль. Система автоматически регистрирует актёра;
  + удалять актёра из системы.  
    Пароли всех пользователей хранятся в зашифрованном виде.

FR-02. Авторизация

Все пользователи (управляющий и актёры) входят в систему по логину и паролю. Доступ предоставляется только после успешной аутентификации.

FR-03. Управление спектаклями

Управляющий может:

* + создавать спектакль, указывая: название, автора, жанр, режиссёра, дату премьеры, статус;
  + удалять спектакль, если на него нет назначений актёров и заявок на роли.

FR-04. Управление ролями

Управляющий может:

* + добавлять роль в спектакль, указывая: название, тип (главная/второстепенная), пол, возрастные рамки, описание;
  + удалять роль, если она не назначена ни на одного актёра и нет активных заявок на неё.

FR-05. Назначение и снятие с роли

Управляющий может:

* + назначить актёра на роль в спектакле.

FR-06. Просмотр назначений

* + актёр видит все свои назначения в личном кабинете;
  + управляющий имеет доступ ко всем назначениям по всем спектаклям.

FR-07. Просмотр репертуара и составов

* + все пользователи могут просматривать список спектаклей с их статусами (например, «в постановке», «на паузе», «в репертуаре»).

**4. Нефункциональные требования**

**4.1. Требования к производительности**

|  |  |
| --- | --- |
| Требование | Значение |
| Время отклика системы | Не более 2 секунд при выполнении стандартных операций (просмотр, вход, подача заявки) |
| Количество одновременных пользователей | До 100 (актёров) и 5 управляющих |
| Время доступности системы | Не менее 99% в месяц |

**4.2. Требования к надёжности**

* + система должна обеспечивать целостность данных при сбоях;
  + ежедневное резервное копирование;
  + восстановление данных из резервной копии — не более 30 минут;
  + отказоустойчивость: при сбое сервера — автоматический перезапуск службы.

**4.3. Требования к безопасности**

* + авторизация по логину и паролю;
  + хранение паролей — с использованием хеширования (bcrypt);
  + разделение прав доступа:
  + управляющий — полный доступ;
  + актёр — только просмотр и подача заявок;
  + логирование всех действий управляющего (кто, что, когда изменил);
  + защита от SQL-инъекций, XSS и CSRF-атак.

**4.4. Требования к совместимости**

* мобильные платформы: Android 8.0+;
* разрешения экрана: от 360×640 (мобильные).

**4.5. Требования к эргономике**

* + интерфейс на русском языке;
  + адаптивный дизайн: поддержка мобильных устройств;
  + минимальное количество действий для выполнения задач:
  + вход — 1 клик;
  + подача заявки — не более 3 кликов;
  + подсказки при вводе данных (например, формат даты);
  + отсутствие технических терминов в интерфейсе.

**4.6. Требования к сопровождению**

* + гарантийное сопровождение — 30 дней после сдачи;
  + исправление ошибок — в течение 3 рабочих дней;
  + обновления функционала — по отдельному соглашению;
  + документация: руководство пользователя, техническое описание.

**5. Ограничения системы**

|  |  |
| --- | --- |
| Ограничение | Описание |
| Масштаб | Система рассчитана на 1 театр. Не поддерживает сеть театров. |
| Интеграции | Нет интеграции с внешними системами (бухгалтерия, кассы, почта). |
| Уведомления | Уведомления о заявках и назначениях — только внутри системы. Push-уведомления не реализованы. |
| Языки | Только русский. Другие языки не поддерживаются. |
| Данные | Нет автоматического импорта биографий, фото, аудио/видео материалов актёров. |

**6. Состояния и переходы системы**

**6.1. Состояния пользователя**

|  |  |  |
| --- | --- | --- |
| Состояние | Описание | Переход |
| Неавторизован | Пользователь не вошёл в систему | → Авторизация |
| Авторизован (актёр) | Актёр вошёл, видит свои назначения и может подавать заявки | → Просмотр, подача заявки |
| Авторизован (управляющий) | Управляющий имеет доступ ко всем функциям | → Управление, назначение, обработка заявок |

**6.2. Состояния заявки**

|  |  |
| --- | --- |
| Состояние | Описание |
| Подана | Актёр подал заявку, ожидает рассмотрения |
| Рассматривается | Управляющий открыл заявку |
| Одобрена | Заявка утверждена, создано назначение |
| Отклонена | Заявка отклонена управляющим |

**7. Требования к интерфейсу**

**7.1. Мобильный интерфейс**

* + адаптированная версия веб-интерфейса;
  + упрощённое меню;
  + крупные кнопки, удобные для сенсорного ввода.

**8. Требования к данным**

**8.1. Форматы данных**

|  |  |  |
| --- | --- | --- |
| Поле | Формат | Обязательность |
| Логин | Текст, 3–30 символов | Да |
| Пароль | Текст, 6–20 символов, с проверкой сложности | Да |
| ФИО | Текст, до 100 символов | Да |
| Дата рождения | ДД.ММ.ГГГГ | Нет |
| Дата премьеры | ДД.ММ.ГГГГ | Да |
| Email | example@domain.ru | Нет |

**8.2. Хранение данных**

* + база данных: Firebase Firestore;
  + резервное копирование: автоматическое, ежедневно в 02:00.

**9. Заключение**

Спецификация требований определяет полный набор функций, ограничений и поведения системы.

Система должна быть реализована в соответствии с настоящим документом.

Изменения в спецификацию вносятся только по согласованию сторон в письменной форме.

**Описание прецендентов**

**1. Для роли «Управляющий»:**

* + Назначение актёров на роли — управляющий распределяет актёров по ролям в спектаклях. Это основной процесс формирования составов.
  + Просмотр всех назначений — управляющий просматривает список всех текущих назначений актёров на роли для контроля и корректировки.
  + Создание назначения — после одобрения заявки формируется официальное назначение актёра на роль.
  + Управление актёрами (создать/удалить) — управляющий ведёт базу данных актёров: добавляет новых, редактирует информацию о существующих, удаляет неактуальные записи. Включает прецедент «Ввод данных актёра» — заполнение/обновление информации об актёре (ФИО, контакты, навыки и т. д.).
  + Управление спектаклями — управляющий работает с информацией о спектаклях: создаёт, редактирует, удаляет записи о спектаклях. Включает прецедент «Ввод данных спектакля» — заполнение информации о спектакле (название, дата, место проведения и т. д.).
  + Управление ролями — управляющий создаёт, редактирует и удаляет роли в спектаклях. Включает прецедент «Ввод данных роли» — заполнение информации о роли (название, описание, требования к актёру).
  + Просмотр репертуаров и составов — управляющий просматривает текущий репертуар и составы спектаклей для анализа и планирования.

**2. Для роли «Актёр»:**

* + Просмотр назначений — актёр просматривает список своих назначений на роли в спектаклях.
  + Подача заявки на роль — актёр подаёт заявку на участие в определённой роли. Включает прецедент «Добавление комментария» — актёр может добавить комментарий к своей заявке (например, указать свои преимущества для этой роли).
  + Авторизация — актёр входит в систему, используя логин и пароль. Включает прецедент «Проверка логина и пароля» — система проверяет корректность введённых учётных данных для доступа к функционалу.

**Особенности связей:**

* + Связь «include» (например, «Управление актёрами» → «Ввод данных актёра») означает, что один прецедент всегда включает другой как обязательную часть.
  + Связь «extend» (например, «Обработка заявок» → «Одобрение заявки») означает, что один прецедент может быть расширен другим в определённых условиях (например, заявка может быть либо одобрена, либо отклонена).

На основе функциональных требований была разработана диаграмма вариантов использования (рисунок 1).

Изображение выглядит как текст, снимок экрана, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 1 – Диаграмма вариантов использования

**Макетирование**

Файл: activity\_main.xml

Назначение экрана: Главная активность, служащая контейнером для смены фрагментов. Не содержит собственных UI-элементов, кроме FrameLayout, в который динамически загружаются все фрагменты приложения в зависимости от роли пользователя и текущего состояния.

Структура макета:

Контейнер для фрагментов (FrameLayout с ID fragment\_container): Единственный элемент на экране, занимающий всю доступную область. В него динамически подгружаются и заменяются все экраны приложения.

Особенности компоновки:

* + Корневой макет — ConstraintLayout, заполняющий весь экран.
  + FrameLayout растянут на весь экран с использованием констрейнтов (привязан ко всем четырем сторонам родителя).
  + Весь пользовательский интерфейс реализован во фрагментах, которые загружаются в этот контейнер.

Сценарии использования контейнера:

Контейнер fragment\_container используется для отображения всех экранов приложения:

* + AuthenticationFragment — экран входа
  + AdminHostFragment — главный экран администратора
  + ActorHostFragment — главный экран актера
  + ManageActorsFragment — управление актерами
  + ManageAssignmentsFragment — управление назначениями
  + RepertoireFragment — просмотр репертуара
  + AddPlayFragment — добавление спектакля
  + ViewAssignmentsFragment — просмотр и удаление назначений



Рисунок 2 – Макет MainActivity.java

Файл: fragment\_actor\_host.xml

Назначение экрана: Главный экран актера. Отображает приветствие, список текущих ролей (назначений) актера и предоставляет навигацию к репертуару театра.

Структура макета:

1. Заголовок и кнопка выхода:

* + Текст приветствия (text\_view\_actor\_welcome): Отображает имя актера.
  + Кнопка выхода (btn\_logout\_actor): Кнопка с иконкой дверного проема для завершения сеанса.

2. Карточка с приветствием:

* + Декоративная карточка с заголовком "Театральная панель" и пояснительным текстом о назначении экрана.

3. Блок с назначениями:

* + Заголовок "Мои роли".
  + RecyclerView (recycler\_view\_my\_assignments): Список текущих ролей актера в различных спектаклях. Высота фиксирована (200dp).

4. Кнопки действий:

* + Кнопка "Посмотреть репертуар" (btn\_view\_repertoire): Переход к фрагменту с полным списком спектаклей театра.

5. Футер:

* + Информация о версии приложения.

Особенности компоновки:

* + Корневой макет — вертикальный LinearLayout с отступами 24dp и градиентным фоном.
  + Все блоки размещены последовательно сверху вниз.
  + Карточки имеют скругленные углы (используется drawable card\_rounded) и тень (elevation).
  + Кнопка выхода компактная (56x56dp), расположена справа в заголовке.

Функциональные элементы:

1. text\_view\_actor\_welcome — Динамически меняется, подставляя имя актера из сессии.
2. btn\_logout\_actor — При нажатии очищает сессию и возвращает на экран аутентификации.
3. recycler\_view\_my\_assignments — Отображает список назначений (ролей) через AssignmentDetailAdapter.
4. btn\_view\_repertoire — Загружает RepertoireFragment для просмотра всех спектаклей.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 3 – Макет ActorHostFragment.java

Файл: fragment\_add\_play.xml

Назначение экрана: Форма добавления нового спектакля в репертуар театра. Доступна только администратору.

Структура макета:

1. Заголовок:

* + Крупная иконка "➕".
  + Основной заголовок "Добавление спектакля".
  + Подзаголовок "Создание новой театральной постановки".

2. Карточка формы:

* + Карточка (CardView) с белым фоном, скругленными углами и тенью.
  + Внутренний заголовок "Информация о спектакле".

3. Поля ввода:

* + Название спектакля (edit\_text\_title): Обязательное поле для ввода названия.
  + Автор (драматург) (edit\_text\_author): Поле для указания автора пьесы.
  + Режиссер-постановщик (edit\_text\_director): Поле для указания режиссера.
  + Жанр (edit\_text\_genre): Поле для указания жанра (комедия, драма и т.д.).
  + Дата премьеры (edit\_text\_premiere): Поле для ввода даты в формате ДД.ММ.ГГГГ.
  + Подсказка под полем даты с примером формата.

4. Кнопка сохранения:

* + Кнопка "Сохранить спектакль" (button\_save\_play): Выполняет валидацию и отправку данных в Firestore.

5. Футер:

* + Информационная строка "Новая постановка • v1.0".
  + Особенности компоновки:
  + Корневой макет — ScrollView, что позволяет прокручивать форму на устройствах с маленьким экраном.
  + Основной контейнер — вертикальный LinearLayout с отступами 24dp и градиентным фоном.
  + Поля ввода оформлены в виде отдельных горизонтальных LinearLayout с иконками-эмодзи слева.
  + Все поля ввода используют общий фон input\_background для визуального единства.

Функциональные элементы:

1. edit\_text\_title — Основное обязательное поле. При сохранении проверяется на пустоту.
2. edit\_text\_author — Опциональное поле для автора.
3. edit\_text\_director — Опциональное поле для режиссера.
4. edit\_text\_genre — Опциональное поле для жанра.
5. edit\_text\_premiere — Опциональное поле для даты премьеры.
6. button\_save\_play — При нажатии вызывает метод savePlayToFirestore(), который создает объект Play и сохраняет его в коллекцию "plays".

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 4 – Макет AddPlayFragment.java

Файл: fragment\_admin\_host.xml

Назначение экрана: Главная панель администратора. Предоставляет доступ ко всем функциям управления театральной системой через кнопки-действия.

Структура макета:

1. Заголовок:

* + Крупная иконка "👑" (корона).
  + Основной заголовок "Панель администратора".
  + Подзаголовок "Управление театральной системой".

2. Карточка с функциями:

* + Белая карточка со скругленными углами (card\_rounded) и тенью.
  + Внутренний заголовок "Основные функции".

3. Кнопки функций:

* + Список актеров (btn\_manage\_actors): Переход к управлению актерами (ManageActorsFragment).
  + Добавить спектакль (btn\_add\_play): Переход к форме добавления спектакля (AddPlayFragment).
  + Назначить на роль (btn\_assign\_role): Переход к управлению назначениями (ManageAssignmentsFragment).
  + Просмотр репертуара (btn\_view\_repertoire): Переход к просмотру репертуара (RepertoireFragment).
  + Управление назначениями (btn\_manage\_assignments): Переход к просмотру и удалению назначений (ViewAssignmentsFragment).

4. Кнопка выхода:

* + Выйти из системы (btn\_logout\_admin): Красная кнопка для завершения административной сессии.

5. Футер:

* + Информационная строка "Административная панель v1.0".
  + Особенности компоновки:
  + Корневой макет — ScrollView для обеспечения прокрутки при необходимости.
  + Основной контейнер — вертикальный LinearLayout с отступами 24dp и градиентным фоном.
  + Все элементы центрированы по горизонтали (gravity="center\_horizontal").
  + Кнопки функций размещены в белой карточке с равными отступами между ними.

Функциональные элементы:

1. btn\_manage\_actors — Загружает фрагмент для создания, просмотра и удаления актеров.
2. btn\_add\_play — Открывает форму добавления нового спектакля.
3. btn\_assign\_role — Открывает интерфейс для создания новых назначений (актер → роль → спектакль).
4. btn\_view\_repertoire — Показывает список всех спектаклей в театре.
5. btn\_manage\_assignments — Показывает все существующие назначения с возможностью удаления.
6. btn\_logout\_admin — Очищает сессию и возвращает на экран аутентификации.

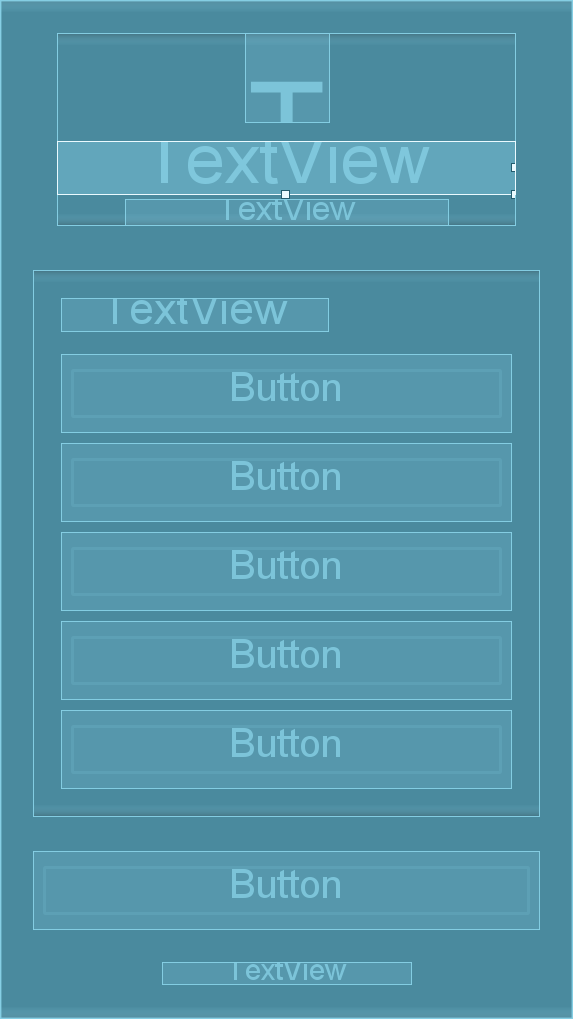


Рисунок 5 – Макет AdminHostFragment.java

Файл: fragment\_authentication.xml

Назначение экрана: Экран аутентификации (входа в систему). Пользователь вводит свой ID для входа в качестве актера или администратора.

Структура макета:

1. Логотип:

* + Изображение театра (image\_theatre) размером 120x120dp.
  + Расположено в верхней части экрана с отступом сверху 80dp.

2. Заголовок:

* + Текст "Театральная сцена" с эмодзи театральной маски.
  + Крупный шрифт 28sp, белый цвет с тенью.

3. Подзаголовок:

* + Текст "Войдите в свой аккаунт" под заголовком.
  + Серый цвет (#E0E0E0), шрифт 16sp.

4. Карточка для входа:

* + Белая карточка (CardView) со скругленными углами (20dp) и тенью.
  + Содержит поле ввода и информационный текст.

5. Поле ввода ID:

* + EditText (edit\_text\_id) с иконкой ключа слева.
  + Подсказка "Введите ваш ID".
  + Использует фон input\_background для стилизации.

6. Информационный текст:

* + Подпись "Используйте ID, полученный от администратора".
  + Мелкий серый текст (12sp) под полем ввода.

7. Кнопка входа:

* + Кнопка "Войти в систему" (button\_login) фиолетового цвета (#B330D5).
  + Содержит иконку дверного проема.
  + Растянута на всю ширину с боковыми отступами 32dp.

8. Футер:

* + Декоративный текст "Добро пожаловать за кулисы" в нижней части экрана.
  + Особенности компоновки:
  + Корневой макет — ConstraintLayout для точного позиционирования элементов.
  + Все элементы центрированы по горизонтали.
  + Градиентный фон (gradient\_background) в едином стиле с другими экранами.
  + Карточка входа ограничена по ширине (0dp с констрейнтами start и end).

Функциональные элементы:

1. edit\_text\_id — Поле для ввода уникального идентификатора пользователя.
2. button\_login — При нажатии вызывает метод authenticateUser(), который ищет пользователя в коллекции Firestore "users" по полю "id".

Процесс аутентификации:

* + Пользователь вводит ID, полученный от администратора.
  + При нажатии кнопки выполняется поиск в Firestore по полю "id".
  + При успешном нахождении:
  + Сохраняются данные сессии через SessionManager.
  + В зависимости от роли ("admin" или "actor") загружается соответствующий фрагмент.
  + При ошибке отображается Toast-сообщение.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 6 – Макет AuthenticationFragment.java

Файл: fragment\_manage\_actors.xml

Назначение экрана: Панель управления актерами. Позволяет администратору создавать новых актеров, просматривать существующий актерский состав и удалять актеров.

Структура макета:

1. Заголовок:

* + Крупная иконка "👥" (два человека).
  + Основной заголовок "Управление актерами".
  + Подзаголовок "Создание и управление актерским составом".

2. Карточка создания актера:

* + Белая карточка (CardView) с тенью и скругленными углами.
  + Внутренний заголовок "Создание актера".

3. Форма создания:

* + ФИО актера (edit\_text\_fio): Поле для ввода полного имени актера.
  + Уникальный ID для входа (edit\_text\_actor\_id): Поле для ввода уникального идентификатора, который актер будет использовать для входа.
  + Кнопка "Создать актера" (button\_create\_actor): Фиолетовая кнопка для сохранения нового актера.

4. Карточка со списком актеров:

* + Вторая белая карточка с тенью.
  + Заголовок "Актерский состав".

5. Счетчик актеров:

* + Текст (tv\_actor\_count): Отображает количество загруженных актеров (например, "Актеров в базе: 5").

6. Список актеров:

* + RecyclerView (recycler\_view\_actors): Отображает список всех актеров через ActorAdapter.
  + Фиксированная высота 400dp для ограничения области прокрутки.

7. Футер:

* + Информационная строка "Актерский состав • v1.0".
  + Особенности компоновки:
  + Корневой макет — ScrollView для прокрутки всего содержимого.
  + Основной контейнер — вертикальный LinearLayout с отступами 24dp.
  + Две основные карточки расположены последовательно сверху вниз.
  + Поля ввода оформлены в горизонтальных LinearLayout с иконками слева и общим фоном input\_background.

Функциональные элементы:

1. edit\_text\_fio — Поле для ввода имени актера (обязательное).
2. edit\_text\_actor\_id — Поле для ввода уникального ID (обязательное, проверяется на уникальность в Firestore).
3. button\_create\_actor — При нажатии вызывает метод createActor(), который:
   * Проверяет заполненность полей.
   * Проверяет уникальность ID в Firestore.
   * Создает документ в коллекции "users" с ролью "actor".
   * Очищает поля и обновляет список.
4. tv\_actor\_count — Динамически обновляется при загрузке актеров.
5. recycler\_view\_actors — Отображает список актеров с кнопкой удаления для каждого.

Процесс работы:

* + При загрузке фрагмента вызывается loadActors(), который загружает всех пользователей с ролью "actor" из Firestore.
  + Каждый элемент списка содержит:
  + Имя актера.
  + ID актера.
  + Роль актера.
  + Кнопку удаления.
  + При нажатии на кнопку удаления вызывается deleteActor(), который удаляет документ из Firestore.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 7 – Макет ManageActorsFragment.java (Часть1)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 8 – Макет ManageActorsFragment.java (Часть2)

Файл: fragment\_manage\_assignments.xml

Назначение экрана: Форма создания нового назначения (распределения актера на роль в конкретном спектакле). Доступна только администратору.

Структура макета:

1. Заголовок:

* + Крупная иконка "🎯" (мишень).
  + Основной заголовок "Назначение на роли".
  + Подзаголовок "Распределение актеров по спектаклям".

2. Карточка формы:

* + Белая карточка (CardView) с тенью и скругленными углами (20dp).
  + Внутренний заголовок "Новое назначение".

3. Поля формы:

* + Спектакль (spinner\_plays): Выпадающий список для выбора спектакля из загруженных в систему.
  + Актер (spinner\_actors): Выпадающий список для выбора актера из актерского состава.
  + Название роли (edit\_text\_role\_name): Поле для ввода названия роли, на которую назначается актер.

4. Кнопка назначения:

* + Назначить на роль (button\_assign\_actor): Фиолетовая кнопка для создания назначения.

5. Подсказка:

* + Текст "Выберите спектакль, актера и укажите роль" с иконкой лампочки.
  + Особенности компоновки:
  + Корневой макет — ScrollView для обеспечения прокрутки при необходимости.
  + Основной контейнер — вертикальный LinearLayout с отступами 24dp.
  + Все элементы центрированы по горизонтали.
  + Поля формы оформлены в горизонтальных LinearLayout с иконками слева и общим фоном input\_background.
  + Выпадающие списки имеют прозрачный фон для интеграции с фоном контейнера.

Функциональные элементы:

1. spinner\_plays — Загружает список спектаклей из коллекции Firestore "plays". Первым элементом всегда идет "Выберите Спектакль".
2. spinner\_actors — Загружает список актеров (пользователей с ролью "actor") из коллекции "users". Формат отображения: "ФИО (ID)".
3. edit\_text\_role\_name — Поле для ввода произвольного названия роли.
4. button\_assign\_actor — При нажатии вызывает метод createAssignment(), который:
   * Проверяет, что выбраны спектакль и актер, и введено название роли.
   * Проверяет на дублирование (чтобы один актер не был назначен на одну и ту же роль в одном спектакле дважды).
   * Создает документ в коллекции "assignments" в Firestore.

Процесс работы:

* + При загрузке фрагмента вызываются методы loadPlays() и loadActors(), которые заполняют соответствующие списки.
  + Данные хранятся в параллельных списках (playList, actorList) для сопоставления выбранной позиции с объектами.
  + При создании назначения генерируется уникальный assignmentId и сохраняется документ с полями: assignmentId, playId, actorId, roleName.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 9 – Макет ManageAssignmentsFragment.java

Файл: fragment\_repertoire.xml

Назначение экрана: Отображение полного репертуара театра. Показывает все спектакли, доступные в системе, с детальной информацией по каждому. Доступен как администраторам, так и актерам.

Структура макета:

1. Шапка с заголовком:

* + Крупная иконка "📜" (свиток).
  + Основной заголовок "Репертуар театра" темно-синего цвета (#1A237E).
  + Подзаголовок "Актуальные постановки текущего сезона" серого цвета.

2. Панель со счетчиком:

* + Светло-серый фон (#F5F5F5).
  + Иконка театральной маски слева.
  + Счетчик спектаклей (tv\_play\_count): Динамически обновляемый текст, отображающий количество загруженных спектаклей.
  + Декоративная иконка "✨" справа.

3. Список спектаклей:

* + RecyclerView (recycler\_view\_repertoire): Основная область для отображения списка спектаклей через PlayAdapter.
  + Занимает все доступное пространство между панелью счетчика и футером (использует layout\_weight="1").
  + Имеет внутренние отступы 8dp и отключенное обрезание (clipToPadding="false") для лучшего визуального восприятия.

4. Футер:

* + Информационная строка "Театральная программа • v1.0" серого цвета.
  + Особенности компоновки:
  + Корневой макет — вертикальный LinearLayout.
  + Шапка и панель счетчика имеют белый и светло-серый фоны соответственно, что контрастирует с градиентным фоном основного экрана.
  + RecyclerView использует layout\_weight="1" для заполнения оставшегося пространства.
  + Все текстовые элементы центрированы по горизонтали.

Функциональные элементы:

1. tv\_play\_count — Динамически обновляется при загрузке данных. Отображает количество спектаклей в репертуаре с учетом русского склонения (например, "1 спектакль", "2 спектакля", "5 спектаклей").
2. recycler\_view\_repertoire — Отображает список спектаклей через PlayAdapter. Каждый элемент содержит:
   * Название спектакля.
   * Жанр (если указан).
   * Режиссера и автора (объединенные через " • ").
   * Дату премьеры с иконкой календаря.
   * Статус спектакля с соответствующей эмодзи (получается через метод getStatusEmoji()).

Процесс работы:

* + При загрузке фрагмента вызывается метод loadRepertoire(), который загружает все документы из коллекции "plays" в Firestore.
  + Для каждого документа создается объект Play, устанавливается playId (равный ID документа), и объект добавляется в список.
  + Список передается в адаптер, который обновляет RecyclerView.
  + Одновременно обновляется счетчик спектаклей с правильным склонением.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 10 – Макет RepertoireFragment.java

Файл: fragment\_view\_assignments.xml

Назначение экрана: Просмотр всех существующих назначений (распределений актеров на роли в спектаклях) с возможностью удаления. Доступен только администратору.

Структура макета:

1. Заголовок:

* + Крупная иконка "⚙️" (шестеренка).
  + Основной заголовок "Управление назначениями" темно-синего цвета (#1A237E).
  + Подзаголовок "Все текущие роли и распределения" серого цвета.

2. Панель со счетчиком:

* + Светло-серый фон (#F5F5F5).
  + Иконка клипборда "📋" слева.
  + Счетчик назначений (tv\_assignment\_count): Динамически обновляемый текст, отображающий количество загруженных назначений.
  + Декоративная иконка "✨" справа.

3. Список назначений:

* + RecyclerView (recycler\_view\_assignments): Основная область для отображения списка всех назначений через ViewAssignmentAdapter.
  + Занимает все доступное пространство между панелью счетчика и кнопкой назад.
  + Имеет внутренние отступы 8dp и отключенное обрезание (clipToPadding="false").

4. Кнопка назад:

* + Назад к панели (btn\_back): Бирюзовая кнопка (#52CCCA) для возврата на главную панель администратора.
  + Расположена в нижней части экрана с отступами 16dp.

5. Футер:

* + Информационная строка "Распределение ролей • v1.0" серого цвета.
  + Особенности компоновки:
  + Корневой макет — вертикальный LinearLayout.
  + Шапка и панель счетчика имеют белый и светло-серый фоны соответственно, что контрастирует с градиентным фоном.
  + RecyclerView использует layout\_weight="1" для заполнения оставшегося пространства.
  + Кнопка назад размещена над футером, что обеспечивает легкий доступ для навигации.

Функциональные элементы:

1. tv\_assignment\_count — Планируется для отображения количества назначений (в текущем коде счетчик реализован в других фрагментах, но здесь пока не подключен).
2. recycler\_view\_assignments — Отображает список назначений через ViewAssignmentAdapter. Каждый элемент содержит:
   * Название роли.
   * Информацию об актере (имя или ID).
   * Информацию о спектакле (название или ID).
   * Кнопку удаления (красная).
3. btn\_back — При нажатии вызывает getActivity().onBackPressed(), возвращая на предыдущий экран (главную панель администратора).

Процесс работы:

* + При загрузке фрагмента вызывается метод loadAllAssignments(), который загружает все документы из коллекции "assignments" в Firestore.
  + Для каждого документа создается объект Assignment, устанавливаются documentId и assignmentId.
  + Затем вызывается loadAdditionalDetails(), который для каждого назначения:
  + Загружает имя актера из коллекции "users" по actorId.
  + Загружает название спектакля из коллекции "plays" по playId.
  + Обновляет объекты и уведомляет адаптер.
  + Каждое назначение отображается с кнопкой удаления, при нажатии на которую вызывается deleteAssignment().

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 11 – Макет ViewAssignmentsFragment.java

**Тестирование**

**1 Модульное тестирование**

Модульное тестирование было проведено с помощью тест-кейсов.

1 Модуль: AuthenticationFragment

Вид тестирования: модульное.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID тест-кейса** | **Название тест-кейса** | **Цель тестирования** | **Предусловие** | **Шаги выполнения** | **Ожидаемый результат** |
| TC-AUTH-01 | Вход с пустым ID | Проверка валидации обязательного поля ID | Открыт экран аутентификации | 1. Оставить поле ID пустым2. Нажать «Войти» | Появляется сообщение «Введите ID», запрос в Firestore не выполняется |
| TC-AUTH-02 | Вход с несуществующим ID | Проверка обработки отсутствующего пользователя | В БД нет пользователя с таким ID | 1. Ввести несуществующий ID2. Нажать «Войти» | Появляется сообщение «ID не найден. Попробуйте снова», вход не выполняется |
| TC-AUTH-03 | Вход администратора | Проверка входа пользователя с ролью admin | В БД есть пользователь с ролью admin | 1. Ввести ID администратора2. Нажать «Войти» | Сессия сохранена, открыт AdminHostFragment, сообщение «Вход как Администратор» |
| TC-AUTH-04 | Вход актёра | Проверка входа пользователя с ролью actor | В БД есть пользователь с ролью actor | 1. Ввести ID актёра2. Нажать «Войти» | Сессия сохранена, открыт ActorHostFragment, отображается приветствие |
| TC-AUTH-05 | Пользователь с неверной ролью | Проверка обработки неизвестной роли | В БД роль ≠ admin/actor | 1. Ввести ID2. Нажать «Войти» | Сообщение «Неверная роль», переход не осуществляется |
| TC-AUTH-06 | Некорректный объект Actor | Проверка устойчивости к ошибке данных | Документ существует, но объект Actor некорректен | 1. Ввести ID2. Нажать «Войти» | Сообщение «Ошибка: неверный формат профиля», приложение не падает |
| TC-AUTH-07 | Отсутствие Context | Проверка защиты от NullPointerException | Fragment отсоединён от Activity | 1. Вызвать authenticateUser() | Сообщение «Ошибка контекста», аварийного завершения нет |
| TC-AUTH-08 | SessionManager не инициализирован | Проверка резервной инициализации сессии | SessionManager = null | 1. Выполнить успешный вход | Сессия инициализируется альтернативным способом, вход успешен |
| TC-AUTH-09 | Проверка сохранения данных сессии | Проверка корректности данных в SessionManager | Успешная авторизация | 1. Выполнить вход2. Проверить данные сессии | ID, имя и роль сохранены, isLoggedIn = true |
| TC-AUTH-10 | Повторный вход пользователя | Проверка повторной авторизации | Пользователь ранее был в системе | 1. Выполнить выход2. Повторно войти | Старая сессия очищена, новая сессия сохранена корректно |

2 Модуль: SessionManager

Вид тестирования: модульное

Назначение модуля: управление пользовательской сессией (ID, имя, роль, статус входа).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID тест-кейса** | **Название тест-кейса** | **Цель тестирования** | **Предусловие** | **Шаги выполнения** | **Ожидаемый результат** |
| TC-SES-01 | Инициализация SessionManager с контекстом | Проверка корректной инициализации SharedPreferences | Приложение запущено | 1. Вызвать SessionManager.initialize(context) | Экземпляр SessionManager создан, SharedPreferences инициализированы |
| TC-SES-02 | Получение экземпляра без инициализации | Проверка защиты от некорректного вызова | initialize() не вызывался | 1. Вызвать getInstance() | Возвращается null, в логах ошибка инициализации |
| TC-SES-03 | Сохранение ID пользователя | Проверка сохранения ID актёра | SessionManager инициализирован | 1. Вызвать setCurrentActorId("123") | ID сохранён в SharedPreferences, isLoggedIn = true |
| TC-SES-04 | Сохранение имени пользователя | Проверка сохранения имени актёра | SessionManager инициализирован | 1. Вызвать setCurrentActorName("Иванов И.И.") | Имя пользователя сохранено |
| TC-SES-05 | Сохранение роли пользователя | Проверка сохранения роли | SessionManager инициализирован | 1. Вызвать setCurrentActorRole("actor") | Роль пользователя сохранена |
| TC-SES-06 | Получение сохранённых данных | Проверка корректности геттеров | Данные ранее сохранены | 1. Вызвать getCurrentActorId()2. getCurrentActorName()3. getCurrentActorRole() | Возвращаются корректные значения |
| TC-SES-07 | Проверка статуса входа | Проверка метода isLoggedIn() | ID пользователя сохранён | 1. Вызвать isLoggedIn() | Возвращается true |
| TC-SES-08 | Очистка сессии | Проверка выхода пользователя | Пользователь авторизован | 1. Вызвать clearSession() | Все данные удалены, isLoggedIn = false |
| TC-SES-09 | Повторная инициализация | Проверка Singleton-поведения | SessionManager уже создан | 1. Повторно вызвать initialize(context) | Создаётся один и тот же экземпляр |
| TC-SES-10 | Работа с null-контекстом | Проверка устойчивости к ошибке | Context = null | 1. Вызвать getInstance(null) | Экземпляр не создаётся, приложение не падает |

3 Модуль: ActorAdapter

Тип компонента: RecyclerView.Adapter

Вид тестирования: модульное

Назначение модуля: отображение списка актёров и обработка удаления

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID тест-кейса** | **Название тест-кейса** | **Цель тестирования** | **Предусловие** | **Шаги выполнения** | **Ожидаемый результат** |
| TC-ACTAD-01 | Создание адаптера с корректным списком | Проверка инициализации адаптера | Список актёров не пуст | 1. Создать ActorAdapter(actorList) | Адаптер создан без ошибок |
| TC-ACTAD-02 | Проверка количества элементов | Проверка метода getItemCount() | Список содержит N актёров | 1. Вызвать getItemCount() | Возвращается значение N |
| TC-ACTAD-03 | Отображение имени актёра | Проверка привязки имени | Actor.name ≠ null | 1. Вызвать onBindViewHolder() | В TextView отображается имя актёра |
| TC-ACTAD-04 | Отображение значения «Без имени» | Проверка обработки null имени | Actor.name = null | 1. Вызвать onBindViewHolder() | Отображается текст «Без имени» |
| TC-ACTAD-05 | Отображение ID актёра | Проверка отображения ID | Actor.id ≠ null | 1. Вызвать onBindViewHolder() | Отображается «ID: <значение>» |
| TC-ACTAD-06 | Отображение отсутствующего ID | Проверка обработки null ID | Actor.id = null | 1. Вызвать onBindViewHolder() | Отображается «ID: Не указан» |
| TC-ACTAD-07 | Отображение роли актёра | Проверка привязки роли | Actor.role ≠ null | 1. Вызвать onBindViewHolder() | Отображается «Роль: <роль>» |
| TC-ACTAD-08 | Обработка нажатия кнопки «Удалить» | Проверка вызова listener | Listener установлен | 1. Нажать кнопку «Удалить» | Вызывается onDeleteClick() с актёром и позицией |
| TC-ACTAD-09 | Отсутствие listener | Проверка устойчивости без listener | Listener = null | 1. Нажать кнопку «Удалить» | Ошибок и падений нет |
| TC-ACTAD-10 | Обновление списка актёров | Проверка метода updateList() | Новый список передан | 1. Вызвать updateList(newList) | Данные обновлены, notifyDataSetChanged() вызван |

4 Модуль: AdminHostFragment

Вид тестирования: модульное

Назначение модуля: навигация администратора по функциям системы и выход из системы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID тест-кейса** | **Название тест-кейса** | **Цель тестирования** | **Предусловие** | **Шаги выполнения** | **Ожидаемый результат** |
| TC-ADM-01 | Загрузка фрагмента администратора | Проверка корректного создания интерфейса | Пользователь с ролью admin авторизован | 1. Открыть AdminHostFragment | Фрагмент загружается без ошибок, кнопки отображаются |
| TC-ADM-02 | Переход к управлению актёрами | Проверка кнопки «Управление актёрами» | AdminHostFragment открыт | 1. Нажать кнопку «Управление актёрами» | Загружается ManageActorsFragment |
| TC-ADM-03 | Переход к управлению назначениями | Проверка кнопки «Управление назначениями» | AdminHostFragment открыт | 1. Нажать кнопку «Управление назначениями» | Загружается ViewAssignmentsFragment |
| TC-ADM-04 | Переход к добавлению спектакля | Проверка кнопки «Добавить спектакль» | AdminHostFragment открыт | 1. Нажать кнопку «Добавить спектакль» | Загружается AddPlayFragment |
| TC-ADM-05 | Переход к назначению ролей | Проверка кнопки «Назначить роль» | AdminHostFragment открыт | 1. Нажать кнопку «Назначить роль» | Загружается ManageAssignmentsFragment |
| TC-ADM-06 | Переход к просмотру репертуара | Проверка кнопки «Просмотр репертуара» | AdminHostFragment открыт | 1. Нажать кнопку «Просмотр репертуара» | Загружается RepertoireFragment |
| TC-ADM-07 | Выход администратора | Проверка завершения сессии | Администратор авторизован | 1. Нажать кнопку «Выход» | Сессия очищена, открыт AuthenticationFragment |
| TC-ADM-08 | Очистка SessionManager при выходе | Проверка очистки данных сессии | Пользователь вошёл как admin | 1. Нажать «Выход»2. Проверить SessionManager | Все данные удалены, isLoggedIn = false |
| TC-ADM-09 | Нажатие кнопки при null Activity | Проверка устойчивости к ошибке | getActivity() = null | 1. Нажать любую кнопку | Приложение не падает |
| TC-ADM-10 | Повторный вход администратора | Проверка корректного повторного входа | Администратор ранее вышел | 1. Повторно войти как admin | Фрагмент администратора загружается корректно |

5 Модуль: ManageActorsFragment

Вид тестирования: модульное

Назначение модуля: создание, просмотр и удаление актёров в системе.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID тест-кейса** | **Название тест-кейса** | **Цель тестирования** | **Предусловие** | **Шаги выполнения** | **Ожидаемый результат** |
| TC-MA-01 | Загрузка экрана управления актёрами | Проверка инициализации интерфейса | Пользователь — администратор | 1. Открыть ManageActorsFragment | Экран загружается, поля ввода и список отображаются |
| TC-MA-02 | Просмотр списка актёров | Проверка загрузки актёров из Firestore | В БД есть актёры с ролью actor | 1. Открыть экран | Отображается список актёров |
| TC-MA-03 | Отсутствие актёров в базе | Проверка обработки пустого списка | В БД нет актёров | 1. Открыть экран | Сообщение «Нет зарегистрированных актёров» |
| TC-MA-04 | Создание актёра с пустыми полями | Проверка валидации формы | Экран открыт | 1. Оставить поля пустыми2. Нажать «Создать» | Сообщение об обязательности заполнения полей |
| TC-MA-05 | Создание актёра с корректными данными | Проверка успешного создания | В БД нет актёра с таким ID | 1. Ввести ФИО и ID2. Нажать «Создать» | Актёр сохранён, список обновлён |
| TC-MA-06 | Создание актёра с существующим ID | Проверка уникальности ID | В БД есть актёр с таким ID | 1. Ввести существующий ID2. Нажать «Создать» | Сообщение «ID уже используется» |
| TC-MA-07 | Очистка полей после создания | Проверка UI после успешного создания | Актёр успешно создан | 1. Создать актёра | Поля ввода очищены |
| TC-MA-08 | Удаление актёра | Проверка функции удаления | В списке есть актёр | 1. Нажать «Удалить» | Актёр удалён из БД и списка |
| TC-MA-09 | Удаление актёра без ID | Проверка обработки ошибки | ID актёра отсутствует | 1. Нажать «Удалить» | Сообщение об ошибке, приложение не падает |
| TC-MA-10 | Обновление счётчика актёров | Проверка отображения количества | Список обновлён | 1. Создать/удалить актёра | Счётчик отображает корректное количество |

6 Модуль: ManageAssignmentsFragment

Вид тестирования: модульное

Назначение модуля: назначение актёров на роли в спектаклях

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID тест-кейса** | **Название тест-кейса** | **Цель тестирования** | **Предусловие** | **Шаги выполнения** | **Ожидаемый результат** |
| TC-MAS-01 | Загрузка экрана назначения ролей | Проверка инициализации интерфейса | Пользователь — администратор | 1. Открыть ManageAssignmentsFragment | Экран загружается, элементы UI отображаются |
| TC-MAS-02 | Загрузка списка спектаклей | Проверка получения данных из Firestore | В БД есть спектакли | 1. Открыть экран | Spinner спектаклей заполнен |
| TC-MAS-03 | Загрузка списка актёров | Проверка фильтрации по роли actor | В БД есть актёры | 1. Открыть экран | Spinner актёров заполнен |
| TC-MAS-04 | Назначение роли без выбора данных | Проверка валидации формы | Элементы не выбраны | 1. Нажать «Назначить» | Сообщение о необходимости выбора |
| TC-MAS-05 | Назначение роли с пустым названием | Проверка валидации поля роли | Спектакль и актёр выбраны | 1. Не вводить роль2. Нажать «Назначить» | Сообщение об ошибке |
| TC-MAS-06 | Успешное назначение роли | Проверка сохранения назначения | Данные корректны | 1. Выбрать спектакль2. Выбрать актёра3. Ввести роль4. Нажать «Назначить» | Назначение сохранено, UI очищен |
| TC-MAS-07 | Повторное назначение той же роли | Проверка защиты от дублирования | Назначение уже существует | 1. Повторить назначение | Сообщение о дублировании |
| TC-MAS-08 | Очистка UI после назначения | Проверка состояния формы | Назначение создано | 1. Завершить назначение | Поле роли очищено, Spinner сброшен |
| TC-MAS-09 | Ошибка Firestore при сохранении | Проверка обработки ошибки | Firestore недоступен | 1. Выполнить назначение | Сообщение об ошибке |
| TC-MAS-10 | Назначение с некорректными данными | Проверка устойчивости | Данные повреждены | 1. Назначить роль | Приложение не падает |

7 Модуль: ViewAssignmentsFragment

Вид тестирования: модульное

Назначение модуля: просмотр всех назначений и удаление назначений администратором

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID тест-кейса** | **Название тест-кейса** | **Цель тестирования** | **Предусловие** | **Шаги выполнения** | **Ожидаемый результат** |
| TC-VA-01 | Загрузка экрана просмотра назначений | Проверка инициализации интерфейса | Пользователь — администратор | 1. Открыть ViewAssignmentsFragment | Экран загружается, список отображается |
| TC-VA-02 | Загрузка списка назначений | Проверка получения данных из Firestore | В БД есть назначения | 1. Открыть экран | Назначения отображаются в списке |
| TC-VA-03 | Отсутствие назначений | Проверка обработки пустого списка | В БД нет назначений | 1. Открыть экран | Отображается пустой список без ошибок |
| TC-VA-04 | Загрузка имени актёра | Проверка получения доп. данных | В назначении есть actorId | 1. Загрузить экран | Отображается имя актёра |
| TC-VA-05 | Загрузка названия спектакля | Проверка получения доп. данных | В назначении есть playId | 1. Загрузить экран | Отображается название спектакля |
| TC-VA-06 | Удаление назначения | Проверка функции удаления | Назначение существует | 1. Нажать «Удалить» | Назначение удалено из БД и списка |
| TC-VA-07 | Удаление без ID назначения | Проверка обработки ошибки | ID назначения отсутствует | 1. Нажать «Удалить» | Сообщение об ошибке, приложение не падает |
| TC-VA-08 | Ошибка Firestore при удалении | Проверка обработки ошибки | Firestore недоступен | 1. Нажать «Удалить» | Сообщение об ошибке удаления |
| TC-VA-09 | Обновление списка после удаления | Проверка обновления RecyclerView | Назначение удалено | 1. Удалить назначение | Элемент исчезает из списка |
| TC-VA-10 | Кнопка «Назад» | Проверка навигации | Экран открыт | 1. Нажать «Назад» | Возврат на предыдущий экран |

8 Модуль: AddPlayFragment

Вид тестирования: модульное

Назначение модуля: добавление нового спектакля в репертуар

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID тест-кейса** | **Название тест-кейса** | **Цель тестирования** | **Предусловие** | **Шаги выполнения** | **Ожидаемый результат** |
| TC-AP-01 | Загрузка экрана добавления спектакля | Проверка инициализации интерфейса | Пользователь — администратор | 1. Открыть AddPlayFragment | Экран загружается, поля ввода отображаются |
| TC-AP-02 | Добавление спектакля с пустым названием | Проверка валидации обязательного поля | Экран открыт | 1. Оставить поле «Название» пустым2. Нажать «Сохранить» | Сообщение «Введите название спектакля» |
| TC-AP-03 | Добавление спектакля с минимальными данными | Проверка успешного сохранения | Название введено | 1. Ввести название2. Нажать «Сохранить» | Спектакль сохранён со статусом «В репертуаре» |
| TC-AP-04 | Добавление спектакля с заполненными полями | Проверка передачи всех данных | Все поля заполнены | 1. Заполнить форму2. Нажать «Сохранить» | Спектакль сохранён с корректными данными |
| TC-AP-05 | Генерация ID спектакля | Проверка установки playId | Сохранение выполнено | 1. Добавить спектакль | playId равен ID документа Firestore |
| TC-AP-06 | Очистка экрана после сохранения | Проверка возврата к предыдущему экрану | Сохранение успешно | 1. Добавить спектакль | Происходит возврат назад |
| TC-AP-07 | Ошибка Firestore при сохранении | Проверка обработки ошибки | Firestore недоступен | 1. Нажать «Сохранить» | Сообщение об ошибке |
| TC-AP-08 | Статус спектакля по умолчанию | Проверка бизнес-логики | Статус не задан вручную | 1. Добавить спектакль | Статус = «В репертуаре» |
| TC-AP-09 | Повторное добавление спектакля | Проверка устойчивости | Экран открыт | 1. Добавить несколько спектаклей | Каждый спектакль сохраняется |
| TC-AP-10 | Некорректные данные в полях | Проверка устойчивости | Введены спецсимволы | 1. Добавить спектакль | Приложение не падает |

9 Модуль: RepertoireFragment

Вид тестирования: модульное

Назначение модуля: просмотр репертуара спектаклей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID тест-кейса** | **Название тест-кейса** | **Цель тестирования** | **Предусловие** | **Шаги выполнения** | **Ожидаемый результат** |
| TC-REP-01 | Загрузка экрана репертуара | Проверка инициализации интерфейса | Пользователь авторизован | 1. Открыть RepertoireFragment | Экран загружается, RecyclerView отображается |
| TC-REP-02 | Отображение индикатора загрузки | Проверка состояния загрузки | Экран открыт | 1. Открыть экран | Отображается текст «Загрузка спектаклей…» |
| TC-REP-03 | Загрузка списка спектаклей | Проверка получения данных из Firestore | В БД есть спектакли | 1. Открыть экран | Спектакли отображаются в списке |
| TC-REP-04 | Отсутствие спектаклей | Проверка обработки пустого списка | В БД нет спектаклей | 1. Открыть экран | Сообщение «Спектакли не найдены» |
| TC-REP-05 | Отображение названия спектакля | Проверка отображения данных | У спектакля есть название | 1. Загрузить экран | Название корректно отображается |
| TC-REP-06 | Подсчёт количества спектаклей | Проверка счётчика | В БД несколько спектаклей | 1. Загрузить экран | Счётчик отображает корректное количество |
| TC-REP-07 | Корректное склонение в счётчике | Проверка бизнес-логики | Количество = 1 / 2–4 / ≥5 | 1. Загрузить экран | Используется корректная форма слова |
| TC-REP-08 | Обновление списка при возврате | Проверка onResume() | Экран открыт повторно | 1. Вернуться на экран | Репертуар обновляется |
| TC-REP-09 | Ошибка Firestore при загрузке | Проверка обработки ошибки | Firestore недоступен | 1. Открыть экран | Отображается сообщение об ошибке |
| TC-REP-10 | Некорректные данные спектакля | Проверка устойчивости | Документ повреждён | 1. Загрузить экран | Приложение не падает, лог фиксирует ошибку |

**2 Интеграционное тестирование**

Для проведения интеграционного тестирования был составлен граф программы (рисунок 12).

Изображение выглядит как луна, круг, снимок экрана, черно-белый

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 12 – Управляющий шраф программы.

Далее были составлены тестовые пути на основе управляющего графа программы:

|  |  |  |
| --- | --- | --- |
| **ID тестового пути** | **Последовательность узлов** | **Описание тестового пути** |
| **TP-01** | 1 → 2 → 4 → 2 → 5 → 2 → 6 → 2 → 7 → 2 → 1 | Полный цикл работы администратора |
| **TP-02** | 1 → 3 → 7 → 3 → 1 | Цикл работы актёра |
| **TP-03** | 1 → 2 → 5 → 2 → 7 → 2 → 1 | Создание назначения и выход |
| **TP-04** | 1 → 2 → 4 → 2 → 6 → 2 → 1 | Управление актёрами и спектаклями |

Тест-кейсы для интеграционного тестирования:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID тест-кейса** | **ID тестового пути** | **Название тест-кейса** | **Описание** | **Предусловие** | **Шаги выполнения** | **Ожидаемый результат** |
| TC-INT-001 | TP-01 | Полный рабочий цикл администратора | Проверка всех возможностей администратора | Пользователь с ролью администратора существует | 1. Запустить приложение → AuthenticationFragment  2. Ввести ID администратора → AdminHostFragment  3. «Управление актерами» → ManageActorsFragment  4. Создать актера  5. Назад → AdminHostFragment  6. «Добавить спектакль» → AddPlayFragment  7. Создать спектакль  8. Назад → AdminHostFragment  9. «Назначить роль» → ManageAssignmentsFragment  10. Создать назначение  11. Назад → AdminHostFragment  12. «Просмотр репертуара» → RepertoireFragment  13. Проверить спектакль  14. Назад → AdminHostFragment  15. «Выход» → AuthenticationFragment | Все переходы выполняются корректно, данные сохраняются, интерфейс работает без ошибок |
| TC-INT-002 | TP-02 | Цикл работы актера | Проверка функционала актера | Пользователь с ролью актера существует | 1. Запустить приложение → AuthenticationFragment  2. Ввести ID актера → ActorHostFragment  3. Проверить приветствие  4. Проверить назначения  5. «Просмотр репертуара» → RepertoireFragment  6. Проверить список спектаклей  7. Назад → ActorHostFragment  8. «Выход» → AuthenticationFragment | Актер видит свои назначения, репертуар, корректно выходит из системы |
| TC-INT-003 | TP-04 | Создание и удаление актера | Проверка жизненного цикла актера | Администратор авторизован | 1. Войти как администратор → AdminHostFragment  2.ManageActorsFragment  3. Создать актера  4. Проверить отображение  5. Удалить актера  6. Проверить отсутствие в списке  7. Назад | Актер успешно создается, отображается и удаляется |
| TC-INT-004 | TP-03 | Создание назначения и проверка у актера | Проверка сквозного функционала назначений | Администратор и актер существуют | 1. Войти как администратор  2. Создать актера (если нет)  3. Создать спектакль (если нет)  4. Назначить роль  5. Выйти  6. Войти как актер  7. Проверить назначение в ActorHostFragment  8. Проверить отображение в репертуаре | Назначение корректно создается и отображается у актера |
| TC-INT-005 | TP-01 | Управление сессией | Проверка сохранения и очистки сессии | Пользователь авторизован | 1. Войти как администратор  2. Свернуть приложение  3. Развернуть приложение  4. Проверить автоматический вход  5. Выйти из системы6. Перезапустить приложение | Сессия сохраняется, при выходе очищается, повторный вход обязателен |
| TC-INT-006 | TP-01 / TP-02 | Навигация между фрагментами | Проверка корректности навигации | Приложение запущено | 1. Проверить переход Authentication → AdminHost  2. Authentication → ActorHost  3. Проверить кнопки AdminHostFragment  4. Проверить кнопки ActorHostFragment  5. Проверить кнопки «Назад»  6. Проверить системную кнопку «Назад» | Все переходы работают корректно, стек фрагментов не нарушается |
| TC-INT-007 | TP-01 / TP-03 | Работа с Firestore | Проверка интеграции с базой данных | Доступ к Firestore | 1. Создать актеров, спектакли, назначения  2. Проверить  Отображение  3. Удалить данные  4. Проверить обновление UI  5. Проверить обработку сетевых ошибок | Данные синхронизируются, интерфейс обновляется корректно |
| TC-INT-008 | TP-02 | Ролевой доступ | Проверка разграничения прав | В системе есть роли admin и actor | 1. Войти как актер  2. Проверить отсутствие админ-функций  3. Войти как администратор  4. Проверить наличие всех функций  5. Проверить недоступность админ-разделов актеру | Права доступа соблюдаются корректно |