

Техническая документация: Логика приложения-календаря «Кот на часах».

*Москва 2024*

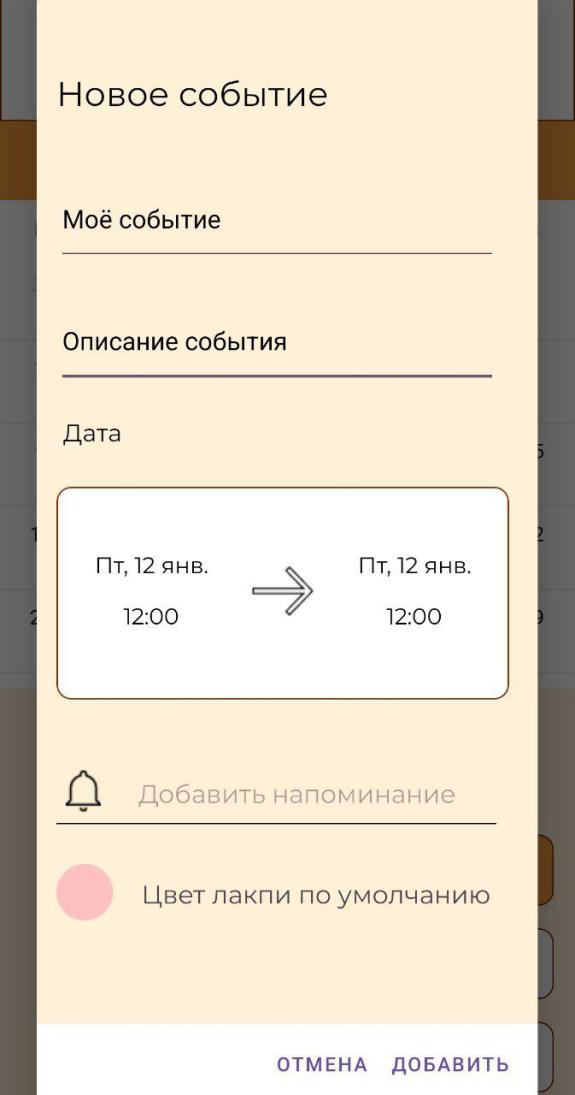
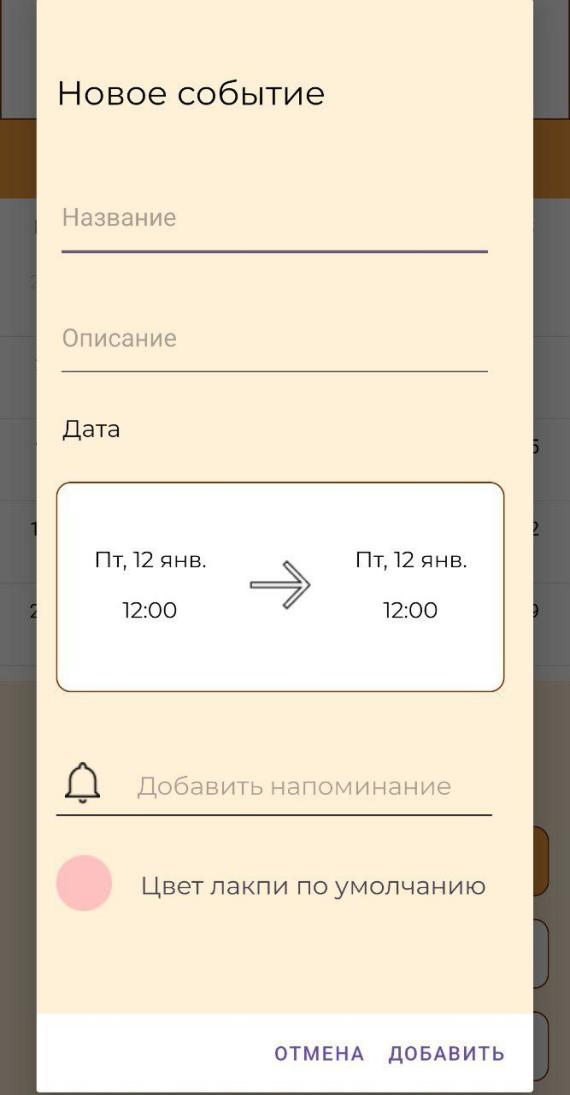
### Введение

Приложение предназначено для управления мероприятиями. Пользователи могут создавать, редактировать и удалять события, просматривать их на календаре и получать уведомления. Архитектура приложения построена на использовании Android Studio для интерфейса и Firebase для хранения данных.

### Основные функции

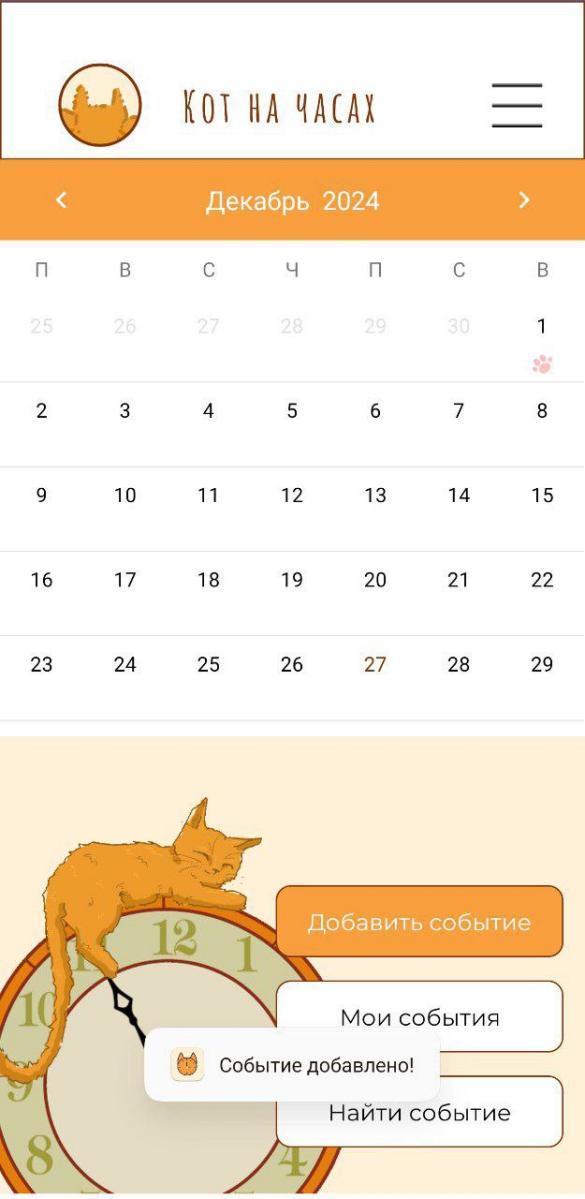
#### Управление мероприятиями

* **Создание события:** Пользователь вводит данные (название, описание, дату, время). Сохранение происходит в базе данных Firebase.
* **Редактирование события:** Возможность изменения данных существующего события.
* **Удаление события:** Удаление события из базы данных.



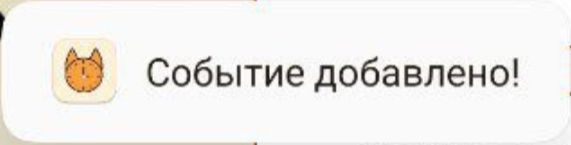
#### Календарь

* Просмотр всех мероприятий в удобном формате календаря.
* Возможность фильтрации событий по дате.
* Интуитивно понятный интерфейс для выбора и навигации по датам.



#### Уведомления

* Настраиваемые напоминания о событиях.
* Уведомления основаны на данных из Firebase (дата и время).



### Структура приложения

#### Архитектура

Приложение построено на основе архитектуры MVC (Model-View-Controller):

* **Model:** Отвечает за данные и логику их обработки (классы Event и Notification).
* **View:** Обеспечивает отображение пользовательского интерфейса на Android Studio.
* **Controller:** Координирует действия пользователя, взаимодействует с моделью и представлением.

#### Структура данных

База данных Firebase содержит:

* Коллекция events:
  + Id\_event (INTEGER, уникальный идентификатор)
  + name (TEXT)
  + description (TEXT)
  + date\_start (DATE)
  + date\_end (DATE)
  + time\_start (TIME)
  + time\_end (TIME)
  + Id\_client (INTEGER)

#### Основные классы

* **Event:**
  + Атрибуты: Id\_event, name, description, date\_end, time\_start, time\_end, Id\_client.
  + Методы: добавление нового события, редактирование существующего события, удаление события, извлечение списка событий из базы данных.
* **Client:**
  + Атрибуты: id\_client, name, email, phone, password.
  + Методы: регистрация клиента, обновление информации клиента, удаление аккаунта клиента, получение списка уведомлений.

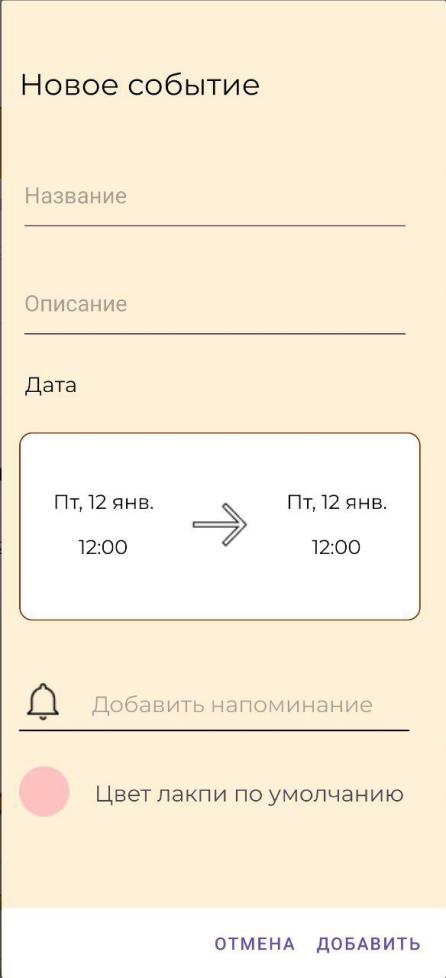
### Логика взаимодействий

#### Создание события

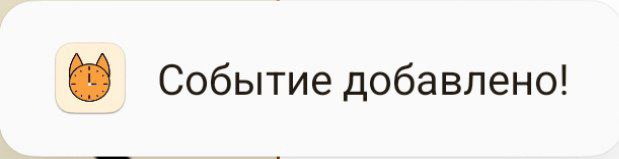
1. Пользователь нажимает кнопку “Добавить событие”.



1. Открывается форма для ввода данных.



1. Данные передаются в контроллер и сохраняются в таблицу events.



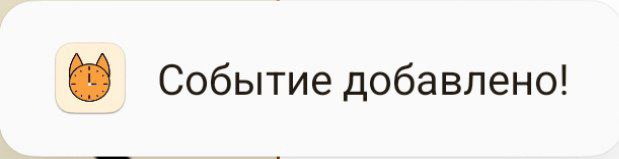
#### Отображение событий на календаре

1. При запуске приложения контроллер извлекает данные из базы.
2. Данные передаются в интерфейс календаря.
3. Пользователь видит отмеченные даты с событиями.



#### Уведомления

1. В момент создания события рассчитывается время для уведомления.
2. Контроллер регистрирует уведомление через библиотеку flutter\_local\_notifications.
3. Приложение отправляет уведомление в установленное время.

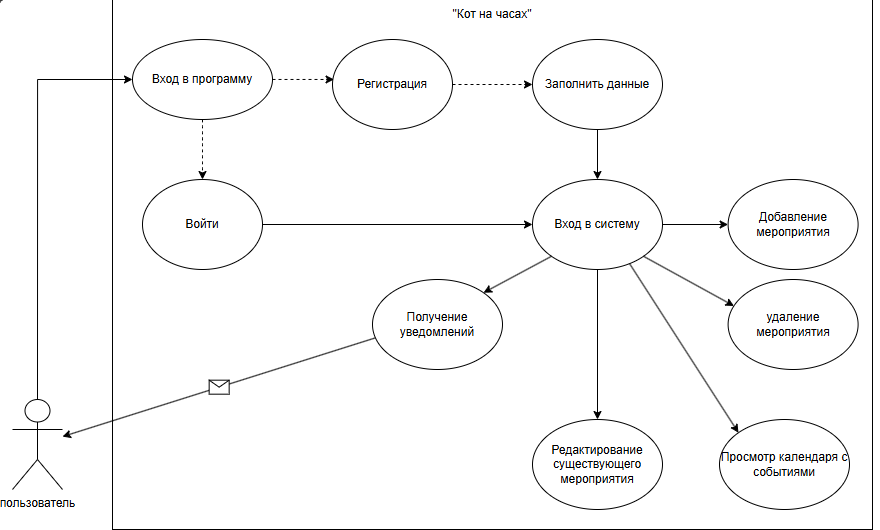


### UML-диаграммы

#### Диаграмма вариантов использования (Use Case Diagram)

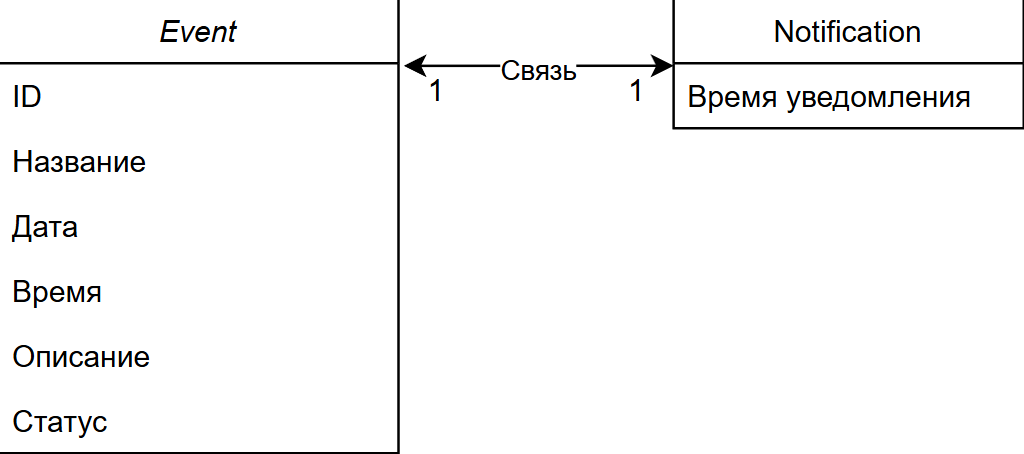
Отображает ключевые действия пользователя:

* Создание, редактирование и удаление мероприятий.
* Просмотр событий в календаре.
* Получение уведомлений.

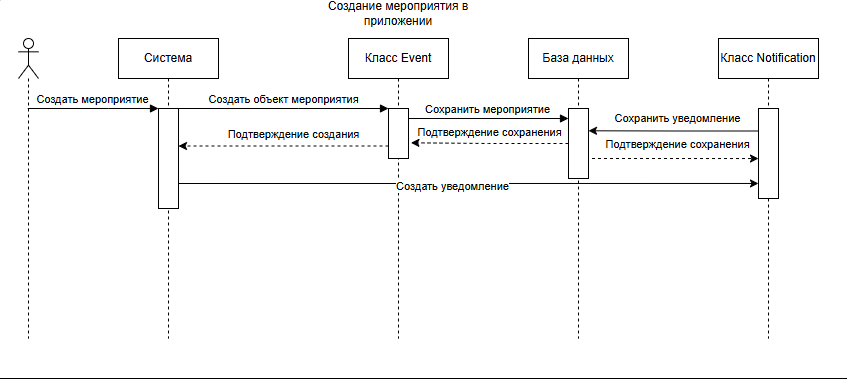
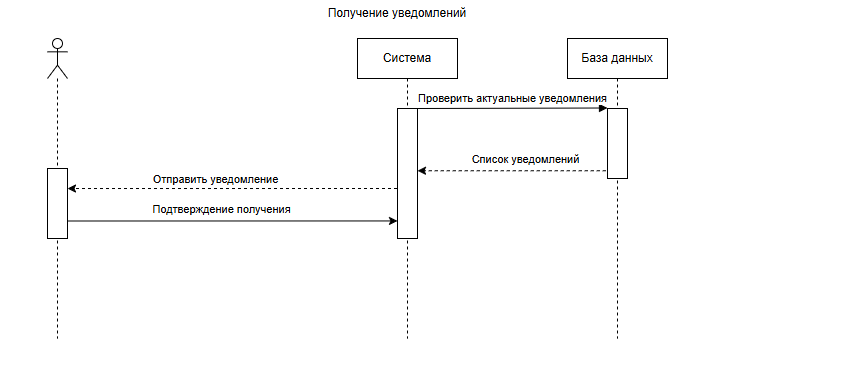


#### Диаграмма классов (Class Diagram)

* **Event:** Содержит данные о мероприятии.
* **Notification:** Управляет уведомлениями.
* **DatabaseHelper:** Отвечает за взаимодействие с SQLite.

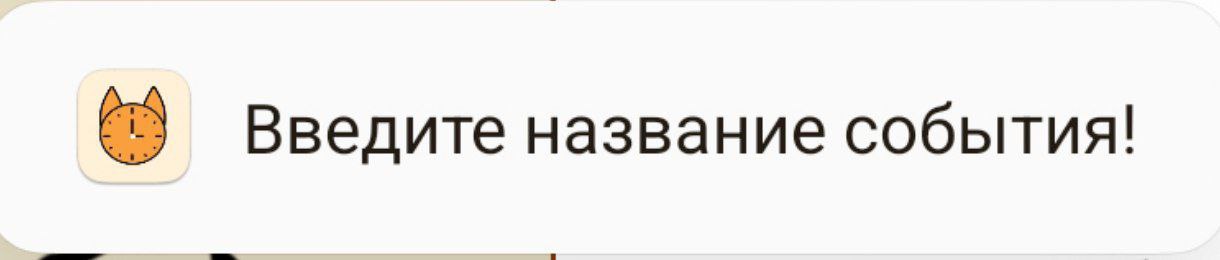


#### Диаграмма последовательностей (Sequence Diagram)

* **Создание мероприятия:** Пользователь → Контроллер → База данных.
* **Получение уведомлений:** База данных → Уведомление → Пользователь.

### Дополнительные аспекты

#### Обработка ошибок

* Некорректный ввод данных: отображение сообщения с пояснением. (При создании события, если мы не вводим название события, то событие не сохраняется в базе данных и выходит ошибка.) 
* Ошибки базы данных: логирование и уведомление пользователя.

### Заключение

Приложение обеспечивает удобное планирование мероприятий, предоставляя пользователям гибкие возможности управления событиями и получения напоминаний. Основываясь на современной архитектуре и технологиях, оно готово к дальнейшему расширению и интеграции.